CSCML 2023

2023 INTERNATIONAL SYMPOSIUM ON CYBER SECURITY CRYPTOLOGY AND MACHINE LEARNING

JUNE 29-30, 2023
VIRTUAL CONFERENCE

PROFESSOR SHLOMI DOLEV
GENERAL CHAIR

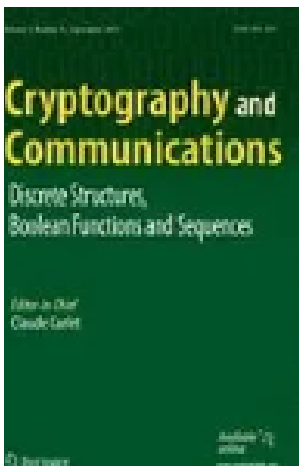
CSCML2023

SPONSORS

אוניברסיטת בן-גוריון בנגב
Ben-Gurion University of the Negev

BGU NHSA

IBM



Springer

Lecture Notes in Computer Science
LNCS LNAI LNBI

INTERNATIONAL ASSOCIATION FOR CRYPTOLOGIC RESEARCH
i c a r

7th International Symposium on Cyber Security Cryptography and Machine Learning (CSCML 2023)

TECHNICAL REPORT

Editors

Shlomi Dolev, Oded Margalit and

Yonah Alexandre Bronstein

Technical Report #23-01

June 29, 2023

The BGU-Negev Hi-Tech Faculty Startup Accelerator,

Department of Computer Science,

Ben-Gurion University, Beer Sheva, Israel

Table of Contents

Ph.D./ Masters Student Research Track
Chair: Oded Margalit

Ph.D./ Masters Student Research Track chaired by Prof. Oded Margalit

In the PhD track this year, we've heard from all the letters of the acronym CSCML:

- **C**yber-**S**ecurity: for example, a paper from Ariel University in Israel about detecting security patches in git repositories;
- **C**ryptology: for example, a work from Amrita University in India about cryptanalysis of SLIM cipher;
- **M**achine **L**earning: for example, research from Ben Gurion University in Israel about scoring the quality of physiotherapy treatment from video input.

Like in previous years, the audience of this session enjoys a "tasting menu" of the state-of-the-art in CSCML research topics, while the speakers get experts' feedback on their work.

I enjoy chairing these sessions and invite all relevant researchers from all over the world to actively attend CSCML-2024, either as an attendee, or, preferably, as a speaker.

See you at CSCML 2024

Regards,

Prof. Oded Margalit,
CSCML 2023 Ph.D./Masters Track Chair
Computer Science department, BGU
and Advanced Research Center, Trellix

Detecting Security Patches via Behavioral Data in Code Repositories

Nitzan Farhi[1], Noam Koenigstein[2], and Yuval Shavitt[3]

[1] School of Electrical Engineering, Tel-Aviv University, Tel-Aviv, Israel
nitzanfarhi@mail.tau.ac.il
[2] Department of Industrial Engineering, Tel-Aviv University, Tel-Aviv, Israel
[3] School of Electrical Engineering, Tel-Aviv University, Tel-Aviv, Israel

Abstract. The absolute majority of software today is developed collaboratively using collaborative version control tools such as Git. It is a common practice that once a vulnerability is detected and fixed, the developers behind the software issue a Common Vulnerabilities and Exposures or CVE record to alert the user community of the security hazard and urge them to integrate the security patch. However, some companies might not disclose their vulnerabilities and just update their repository. As a result, users are unaware of the vulnerability and may remain exposed.
In this paper, we present a system to automatically identify security patches using only the developer behavior in the Git repository without analyzing the code itself or the remarks that accompanied the fix (commit message). We showed we can reveal concealed security patches with an accuracy of 88.3% and F1 Score of 89.8%. This is the first time that a language-oblivious solution for this problem is presented.

Keywords:
CVE, Machine Learning, Git, GitHub, LSTM, Conv1D

Introduction

The absolute majority of software today is developed collaboratively, where developers work separately on a different part of the software, and then merge their code with the rest of the software. To allow this collaboration, source control tools, such as Git, were developed. Git is a standard protocol that supports file versioning, which is extensively used by programmers to collaborate on software development. A repository of files is maintained by GIT, allowing operations such as updating files (push/commit), merging file versions (merge), and splitting a project into two branches (fork).

A hosting service such as GitHub[4] allows users to maintain and share a repository, and also allows tracking issues such as bug reports, feature suggestions, fork repositories, maintain merge requests, etc. This makes GitHub an ecosystem of software, where interaction among developers reveal patterns of behavior [6]. For example, adding a

[4] https://github.com/

new feature to a repository may trigger many forks to the repository since other people will want to modify the feature.

Software is naturally prone to errors, and some of them can be exploited by hackers to abuse the software for malicious acts. These errors are often called vulnerabilities and are usually fixed by a software update called a security patch. Vulnerabilities can be divided into 2 classes: 0-day, a vulnerability that is not known to the public and still does not have a security patch, and N-day, one that has a security patch.

The Common Vulnerabilities and Exposures (CVE[5]) is a glossary of publicly disclosed vulnerabilities. This CVE glossary details the vulnerability date, description, references, and metrics regarding the vulnerability such as its complexity whether user interaction is required, and more. Vulnerabilities are analyzed and the Common Vulnerability Scoring System (CVSS) is employed to evaluate the threat level. The CVE score is often used for prioritizing the security of vulnerabilities.

It is a common practice that once a vulnerability is detected and fixed, the company or individuals behind the software issue a CVE to alert the user community of the security threat, hopefully urging them to integrate the security patch. However, some companies might not disclose their vulnerabilities and just update their repository. This behavior may be in order to allow certain users to use the patch sooner, as a measure of "Security by obscurity", or to avoid public embarrassment. The disadvantage of not disclosing vulnerabilities is that users will not be aware that their software needs to be updated, while hackers can notice the code changes, detect a vulnerability, and exploit it.

Recent work [11, 13] has shown the ability to analyze source code changes and commit messages using natural language processing models. These aforementioned models rely on a certain programming language or on the natural language used in the comments and commit messages. Hence, these earlier models are limited in their ability to detect security patches on a repository that uses other programming languages or where comments are written in a different natural language. For example, a model that was trained on detecting security patches in *C++* would not be able to detect security patches in a *Perl* repository, and since it is less popular, gathering a sufficient amount of data to train a model for *Perl* would be difficult.

In this paper, we propose a method that trains a deep learning (DL) model to recognize the user behavioral data around a security patch of a repository on GitHub (a window of events that will be defined in the Data Pre-processing Section), and uses CVE publications to label if it is related to a security patch. Importantly, our model is insensitive to the programming language, or the natural language used by the developers. This allows us to identify security patches in languages that are rather rare and thus do not have sufficient data to train a model. Our model was able to achieve an accuracy of 88.32% and an F1-Score of 89.75% in detecting a security patch. Hence, our approach can be effectively used for detecting undisclosed security patches.

Since GitHub behavioral data is temporal (various actions over time), we experimented with appropriate models such as Long-Short Term Memory (LSTM) [5] and 1-Dimensional Convolutional Neural Network (Conv1D), which allow calculating re-

[5] https://cve.mitre.org/

lations between features at different time points. Note that we do not require Natural Language processing or manual labeling to detect security patches.

We made the source code for our data extraction and our models publicly available at GitHub[6]. The relevant data-set is available as well[7]. We note that our published data-set covers more repositories than any previous work.

Related Work

In this section, we provide an overview of different works in the field of detecting security patches using the CVE Program.

PatchDB [12] is a dataset of security patch commits on *C/C++* projects. Each patch commit contains the additions and deletions made to the code and a commit message. This dataset was created by extracting security patch commits using the CVE Program (we use a similar extracting technique for our data collection). PatchDB was used in PatchRNN [13] where natural language processing methods were employed to detect which commit is a security patch. This is done by tokenizing the commit message, the additions, and the deletions. Later, the tokens were fed into an LSTM neural network to provide a prediction regarding a specific commit with an accuracy of 83.57%. A similar process was suggested by [7]. A vulnerability-contributing commit can be deduced from the security patch (e.g., by using the command 'git blame'). In [7] tokenizing was performed on vulnerability-contributing commits in *java* repositories and an attention-based GRU was employed in order to classify the severity of the commit.

[14] selected 4 open-source *C/C++* projects and filtered commits by using known keywords that describe a security patch. Then, security researchers were employed to manually label the commits as either commits that *introduces* a vulnerability, commits that *closes* a vulnerability, or commits that are *unrelated* to vulnerabilities. Natural Language Processing methods were used to create a deep learning model based on LSTM and Convolutional Neural network, which led to a reported accuracy of 90%.

[8] also used the CVE Program to extract security patches from git repositories. This was done to analyze and gain insights into the security patches. The authors investigated the number of changes done to the code base, the maturity of the bugs, and the number of bugs that were solved by the commits.

[11] extracted security patch commits from the CVE Program and verified that the commits are security patches by manually inspecting 200 of them. Then, the authors extracted the commit message only from GitHub (and did not use other features such as added lines, removed lines, or date). Repositories that contained C/C++ code were selected, and negative samples were acquired by taking all other commits from the same repositories. The natural language processing model that was used to detect the security patches was a hierarchical attention network with Gated Recurrent Units (GRU) [2] as its layers. [11] reached an accuracy of 92%.

VCCFinder [9] extract security patches of *C/C++* repositories similarly, but used them to understand which is the commit that introduced the vulnerability. This paper

[6] https://github.com/nitzanfarhi/SecurityPatchDetection
[7] https://nitzanfarhi.github.io/datasets

takes the first step in the direction we suggest by extracting limited meta-data: they use present "fork count", "star count", "number of commits" and "programming language" but disregard the entire history of these event data. A Support Vector Machine (SVM) model was employed to detect the vulnerability introducing commits.

[10] used meta-data of Git logs to solve a different problem: detecting repositories that are "engineered", namely repositories that are not personal nor inactive. For this they extracted Integration Frequency, Commit Frequency, Integrator Frequency, Committer Frequency, and Merge Frequency, and transformed it into a time series, which was used to detect whether a repository is "engineered". Similarly, [4] used GitHub's API to extract the number of Forks, open issues closed issues, commits, and max days without commits to detect whether a project is under maintenance or unmaintained.

While there were some previous works that used the git behavioral meta-data, these works were either limited in the amount of meta-data that was used or focused on a different type of task. This work is the first solution that uses meta-data alone, which enables a language-oblivious general solution.

Methods

Data Collection

As explained below, we collected data from multiple sources and processed them into a single data-set that will be used by different models for training. A diagram of the overall data collection process can be seen in Figure 1.

As mentioned before, Git is a collaborative version control system that allows tracking changes made to a repository. A *commit* is a change to a repository that includes the text of the code that is being changed and a commit message describing the change in natural language. To make the repository and the commits accessible to all developers, they can be sent (pushed) to GitHub, a server that mirrors the local Git information.

GraphQL[8] API allows us to iterate over the entire repository's history and gather all events. It must be noted that in this data collection, only data from "master", "main", or activate branches can be collected because inactive branches are deleted and cannot be extracted. The data was collected from the years 2015 to 2021 and features that were collected are detailed in Table 2.

We also used GitHub's GraphQL API to acquire static information about repositories, this information will allow us to create different models for different repositories (for example by programming language) and also improve accuracy, as will be detailed later. The static information includes properties of the repository such as the programming language in use, the current size of the repository, and information about the owner of the repository, e.g., if it is an individual or a company. The list of static features is detailed in Table 1.

To acquire more data, we used "GH Archive"[9], which records GitHub events and can be easily accessed via SQL. This database contains events that cannot be acquired via GraphQL since they are recorded in real-time, and branches that were deleted from

[8] https://graphql.github.io/
[9] https://www.gharchive.org/

the repository cannot be retrieved. We acquired only the repositories that had CVEs and Gathered all the events from "GraphQL" and "GH Archive", making our dataset more accurate and comprehensive to detect unlabeled vulnerabilities. Features extracted from the GH Archive can also be seen in Table 2.

To gather a dataset of existing security patches, we used the largest existing database of vulnerabilities - the CVE Program. A CVE is a unique identifier that identifies a vulnerability, the CVE format is as follows: "CVE-2021-44228" where 2021 is the current year the vulnerability was published and 44228 is a unique vulnerability ID for that year.

The CVE Program publishes all CVEs in a comma-separated values (CSV) format. Each entry in the CSV file is a CVE, and includes the CVE's description and additional links. We used the additional links to acquire security patches in the following way: the additional links' field can be separated into separate links, and if one of the links points to a commit in GitHub, we gather the tuple ⟨CVE-ID, commit-URL⟩.

Finally, we cloned the repositories from GitHub to acquire a mapping between a commit ID and its timestamp, the number of additions, deletions, and the number of files that were changed. This allows us to have timestamped labeled commit events, where a commit can be labeled as either *positive* - namely, a security patch or *negative* - not a security patch. Since all other data we acquired is also dated, we overall have a dataset of events with their date, time, and label.

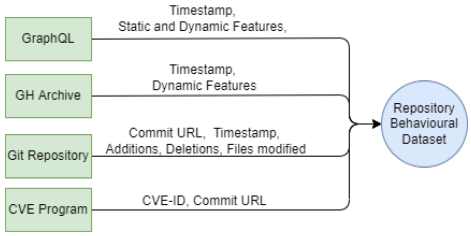


Fig. 1: The overall data collection process.

Data Pre-processing

The main preprocessing method we used is Event-base Aggregation: Time elapsed between two events is not very meaningful, for example, it is insignificant if a fork was done one hour after a commit or two. Therefore, we can discard the date and time (and preserve only the temporal features that are one hot encoded). A configurable parameter is the *window_size*, which denotes the number of events before and after the security patch commit that are concatenated together to create a window (or a vector) of features. This window of features is one of many that will be fed into the model to train it, as detailed in the next section.

Different from the time interval between events, other temporal features were found to be informative such as hour, day of the week, and month. These features, along with

Feature	Type	Explanation
isCompany	Boolean	Does the repository belong to a company or an independent person
isEmployee	Boolean	Is the repository owner an employee of a company
isHirable	Boolean	Is the repository owner can be hired
isSiteAdmin	Boolean	Is the repository owner site administrator
isSponsoringViewer	Boolean	Is the repository owner sponsored
isGitHubStar	Boolean	Is the repository owner a member of the GitHub star program
isCampusExpert	Boolean	Is the repository owner a member of the campus expert program
isDeveloperProgramMember	Boolean	Is the repository owner a member of the GitHub developer program
isVerified	Boolean	Is the repository owner verified by GitHub
isInOrganization	Boolean	Is the repository owner in an organization
createdAt	Integer	Year the repository was created (one-hot encoded)
diskUsage	Integer	Disk usage of the repository
hasIssuesEnabled	Boolean	Does the repository have issues enabled
hasWikiEnabled	Boolean	Does the repository have wiki enabled
isMirror	Boolean	Is the repository a mirror
isSecurityPolicyEnabled	Boolean	Is the security policy enabled in the repository
fundingLinks	Integer	Number of funding links in the repository
languages	List	List of programming languages used in the repository

Table 1: Repository's static features that can be extracted via GraphQL.

Feature	Description	Location
Comment Event	Comment on a commit was added	GraphQL
Stargazers Event	A stargazer was added to the repository	GraphQL
Commit Additions Event	lines were added to the repository (Integer)	GraphQL
Commit Deletions Event	lines were deleted from the repository (Integer)	GraphQL
Subscribers Event	A subscription was added to the repository	GraphQL
Create Event	A Git branch or tag was created	GH Archive
Delete Event	A Git branch or tag was deleted	GH Archive
Issue Comment Event	A comment on an issue was added	GH Archive
Commit Comment Event	A comment on a commit was added	GH Archive
Pull Request Review Comment Event	A comment on a pull request was added	GH Archive
Member Event	Membership on the repository changed	GH Archive
Public Event	A repository changed its viewing settings	GH Archive
Push Event	A commit was pushed to the repository	Both
Fork Event	The repository was forked	Both
Release Event	A new version of the repository was released	Both
Issue Event	A new issue was added	Both
Watchers Event	User marked the repository was watched	Both
Pull Request Event	A Pull request was created	Both

Table 2: Repository's features that can be extracted via GraphQL and GH Archive

event types were encoded using one-hot encoding and fed to the LSTM units which usually take numbers and not events. At this point, we gather for each repository all its security patch commits. Additionally, in order to make the dataset balanced, we also collected a random sample of non-security patch commits.

For each repository, we created a list of events with their date and time. Commit events also have additional normalized integer features: the number of lines added to a file, the number of lines deleted from a file, and the number of files modified. Furthermore, repositories that contained less than 100 events overall were discarded, since they were mostly insufficient for extracting the windows that we required.

Models

When dealing with large amounts of data, it is important to leverage the temporal information encapsulated in the data. A few types of models were examined for this cause. LSTM is an architecture that had proven its efficiency for such tasks. LSTM is based on Recurrent Neural Network (RNN) architecture, where performance decreases as a larger number of time steps are fed into the network. However, LSTM can forget some less important data, while preserving the more important parts of it. GRUs have an architecture that is similar to LSTMs but with fewer parameters, which might be an advantage for some learning tasks. Finally, Conv1D is a Convolutional Neural Network (CNN) that convolutes 1-dimensional data, such as time series. This model can be stacked to allow multi-level pattern recognition.

These model architectures accept their data as a series of vectors, each can be associated with many features. The following process, as can be seen in Figure 2 is employed to allow a time series to be given as input to LSTM, GRU, and Conv1D. For each repository, in our data-set, we gathered for each label - security and non-security patch commits, all the events that surround a commit, as a single input to the model. For example, an 11-event window (composed of a certain event, 5 events before and 5 events after) can be used to classify the window as one that contains a security patch (marked with a red column) or blue (as a window with non-security patches). As can be seen in the figure, some events might contain more than one feature (one column is composed of a few short lines). Window size is the number of events that are contained in the window, each window size is constant, and windows are fed into the models one by one.

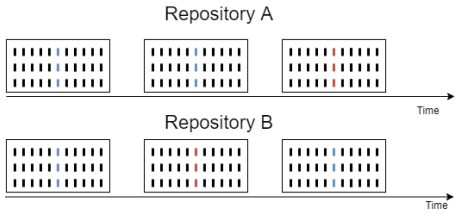


Fig. 2: Demonstration of the sliding window method.

Experimental Design

To validate the proposed methodology, the overall experimental design is as follows: We randomly divided the repositories into training (90%) and test sets (10%). We further employed k-fold cross-validation on the training data to choose hyper-parameters, where $k = 10$ as follows: On each iteration, 10% of the repositories were kept as validation, which the model did not use for training.

The distribution that achieved the best validation accuracy was used for hyper-tuning the parameters, to achieve even better validation accuracy. Finally, the obtained model was used to classify the windows that surrounds a commits of the test set, and performance metrics were calculated as described in Section Performance Evaluation.

As mentioned above, we experimented with several learning models for training:

- *Conv1D* - uses Conv1D layer, Max Pooling, and then fully connected layers (Table 3).
- *LSTM* - LSTM model, which uses LSTM neural network layers (Table 4).
- *GRU* - GRU model, which uses GRU neural network layers (Table 5).

Model: Conv1D

Type	Output Shape	Param #
Conv1D	(None, 198, 64)	8256
MaxPooling1d	(None, 99, 64)	0
Flatten	(None, 6336)	0
Dense	(None, 256)	1622272
Dropout	(None, 256)	0
Dense	(None, 64)	16448
Dropout	(None, 64)	0
Dense	(None, 64)	4160
Dropout	(None, 64)	0
Dense	(None, 1)	65

Total params: 1,651,201

Table 3: Model summary for Conv1D.

The proposed models were implemented using the Keras [3] library with the TensorFlow backend [1] in Python. Training the model was done on a 64-bit Windows 10 Machine with Intel® Core™ i9-10940X CPU @ 3.30GHz with 16.0 GB RAM and NVIDIA GeForce RTX 3080 GPU, where training the training set takes about 4 minutes, and a single prediction takes about 5 milliseconds. The optimizer used is SGD and the selected batch size is 32 and the number of epochs is 50.

Performance Evaluation

To assess the performance of the trained models, we will define the metrics that were used. When solving binary classification problems, accuracy, as defined in Equation 1,

Model: LSTM

Type	Output Shape	Param #
LSTM	(None, 199, 100)	66000
LSTM	(None, 199, 50)	30200
LSTM	(None, 199, 25)	7600
LSTM	(None, 12)	1824
Dense	(None, 1)	13

Total params: 105,637

Table 4: Model summary for LSTM.

Model: GRU

Type	Output Shape	Param #
GRU	(None, 199, 100)	49800
GRU	(None, 199, 50)	22800
GRU	(None, 199, 25)	5775
GRU	(None, 12)	1404
Dense	(None, 1)	13

Total params: 79,792

Table 5: Model summary for GRU.

is usually used as the main evaluation criterion. Other important metrics are precision (Equation 3), recall (Equation 2), and F1-Score (Equation 4), which is calculated from precision and recall. These metrics are defined as follows:

$$Accuracy = \frac{\text{TP + TN}}{\text{TP + FP + TN + FN}}, \tag{1}$$

$$Recall = \frac{\text{TP}}{\text{TP + FN}}, \tag{2}$$

$$Precision = \frac{\text{TP}}{\text{TP + FP}}, \tag{3}$$

$$F1 = 2 \cdot \frac{Precision \cdot Recall}{Precision + Recall}, \tag{4}$$

where TP, TN, FP, and FN are the count of *true-positives*, *true-negatives*, *false-positives*, and *false-negatives* respectively. Finally, we also report the Receiver operating characteristic (ROC). The ROC is a curve that shows the trade-off between the false-positive rate and the true-positive rate according to a selected threshold. The area under the ROC curve (AUC) gives a numerical grade between 0 and 1 to the classifier, where 1 is a perfect classification.

Results

As discussed in Section Experimental Design, a few models were evaluated to determine which is optimal for the task of classifying commits as security patches. We tested

the three models with varied window sizes and found that for all window values Conv1D was the best model. Figure 3 compares the three model accuracy for the optimal window size of 10. Conv1D is leading by 3 percentage points over the next model and hence, in the rest of the experiments, we will focus on this model.

To discuss the rest of the results, we remind the reader that the window size determines the number of events that were gathered before and after the labeled event, e.g., if the window size is 5, we gathered overall 10 events: 5 before and 5 after the event).

In the experiments, we checked the metrics when using different window sizes, which varied between 5 and 20. As can be seen in Figure 5, the optimal window size is 10 in both accuracy and F1-Score , when using the Conv1D model.

Table 6 shows a confusion matrix for the best case, namely where the window size is set to 10 and the used model is Conv1D. In this case, the accuracy achieved was 83.93%, F1-Score was 84.14%, the precision was 0.7550, and the recall was 0.8769.

The number of false positives (FPs) is rather high, which results in rather low precision. We suspect that some of the FPs are unreported security patches that our model identifies correctly. To validate this we manually examined 52 randomly selected FPs and found (see also Section Case Study) that 16 of them contain security patches, namely 30.8%. If we apply this to the confusion matrix of Table 6, we get an estimate of 39 cases that are actually true positives (TPs). This results with a precision of 0.852. The accuracy also improves to 88.32%, and the F1-Score to 89.75%.

Figure 4 depicts the AUC for a window size of 10, which is 0.91. Since we already discussed above that some of the FPs are actually TPs, the real AUC is higher.

Table 7 compares our results with previous works that attempted to detect security patch commits. We noted that our paper is the only one that uses only behavioral git operations, and the amount of repositories used is much larger.

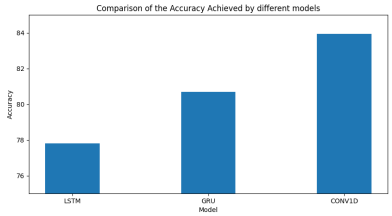


Fig. 3: Comparison of the accuracy of different models.

Other Experiments

In this section, we report experiments that had not produced sufficiently positive results.

– Time-base Aggregation- As mentioned before, we used the Event-base Aggregationto build the horizon window. Another possibility is to use Time-base Aggregationby counting the events by type during time frames (Hour, 2 Hours, Day, etc...)

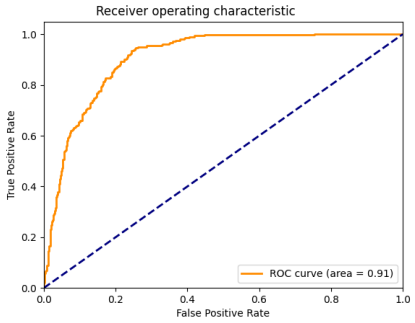


Fig. 4: ROC curve for the best model.

	Not Classified as a Security Patch	Classified as a Security Patch
Not Marked as a Security Patch	373 (TN)	128 (FP)
Marked as a Security Patch	28 (FN)	473 (TP)

Table 6: Confusion Matrix of the classification of patches in the test set (140 repositories).

	Data Used	Amount Of Repositories	Accuracy Score
[13]	Commit message and code difference	313	83.57%
[11]	Commit messages	993	92.81%
Our Paper with original labels	Git & GitHub behavioral data	1389	83.93%
Our Paper with estimated labels	Git & GitHub behavioral data	1389	88.32 %

Table 7: Results Comparison.

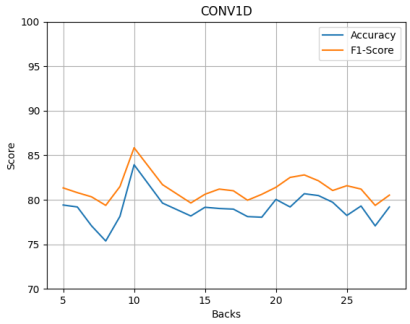


Fig. 5: Conv1D model prediction accuracy and F1-Score as a function of window size.

and summing them up to a single data row. This turned out to achieve a low accuracy rate since events can be sparse and result in many zero entries.

- *Hours / Days* - if Time-base Aggregationis used, the amount of hours/days recorded before the labeled event is the window size.
- *Resample* - if Time-base Aggregationis used, we can aggregate all gathered events differently (for example, all the events in several hours can be aggregated into one vector)

– Extracting location of repositories from GitHub - Since GitHub provides an API to the location of the owner we wanted to test if learning behavior by region or country can improve the prediction. Unfortunately, this field is free text and does not necessarily contain the actual location or is empty, which resulted in a high variety of answers and proved to be not useful.
– Gathering a window before the actual security patch: this is an attempt to predict a security patch in the future or as it occurs, as some events can indicate preparation for a security patch. However, the amount of data before the security patch is not sufficient to make such a prediction, and data after is also needed to achieve satisfactory results.

Case Study

As discussed above, some vulnerabilities are fixed in commits but are not reported as vulnerabilities. In this case, our model will classify them as vulnerabilities, but they will not be labeled as such (false-positive), we manually examined the false-positive samples and checked for vulnerability-fixing changes in them, out of 52 false-positive commits, we found 16 to contain security patches (30%). We will elaborate on two case studies.

LibTIFF

LibTIFF is a library for processing the TIFF image format. A Commit[10] fixed 2 heap-based buffer overflow vulnerabilities and was detected by our model as a security patch, although it was not assigned a CVE (but only an issue at Bugzilla[11], which we do not take labels from).

Wireshark

Wireshark is a network protocol analyzer that allows viewing network communication at several protocol layers. A Commit[12] fixed a denial of service attack by null pointer dereference. This is clear from observing the code and also detected by our model but the commit was not assigned a CVE.

[10] https://GitHub.com/vadz/libtiff/commit/5ed9fea523316c2f5cec4d393e4d5d671c2dbc33
[11] http://bugzilla.maptools.org/show_bug.cgi?id=2633
[12] https://GitHub.com/wireshark/wireshark/commit/ba179a7ef7e660d788cbaade65982ffc7249b91f

Conclusion

In this work, we proposed a methodology to gather repositories' behavioral meta-data from 3 different sources. After merging the different sources into one time series data-set, we used the CVE Program to label the data-set's commits to security patches and non-security patches. We introduced 3 types of time series models and discovered that Conv1D achieved the best classification accuracy of 88.32%. We did not use natural language processing tools, and thus our results are language insensitive (both programming language and natural language).

Our results can be used to detect unreported security patches and warn the community to patch an open security problem. Our future research goal is to explore the impact of the different features on the classification, this can be done by using more interpretable models, comparing model accuracy on specific languages, etc.

Bibliography

[1] Martin Abadi, Paul Barham, Jianmin Chen, Zhifeng Chen, Andy Davis, Jeffrey Dean, Matthieu Devin, Sanjay Ghemawat, Geoffrey Irving, Michael Isard, et al. Tensorflow: A system for large-scale machine learning. In *12th USENIX Symposium on Operating Systems Design and Implementation (OSDI'16)*, pages 265–283, 2016.

[2] Kyunghyun Cho, Bart Van Merriënboer, Caglar Gulcehre, Dzmitry Bahdanau, Fethi Bougares, Holger Schwenk, and Yoshua Bengio. Learning phrase representations using rnn encoder-decoder for statistical machine translation. *arXiv preprint arXiv:1406.1078*, 2014.

[3] François Chollet. keras. https://github.com/fchollet/keras, 2015.

[4] Jailton Coelho, Marco Tulio Valente, Luciana L Silva, and Emad Shihab. Identifying unmaintained projects in github. In *Proceedings of the 12th ACM/IEEE International Symposium on Empirical Software Engineering and Measurement*, pages 1–10, 2018.

[5] Sepp Hochreiter and Jürgen Schmidhuber. Long short-term memory. *Neural computation*, 9(8):1735–1780, 1997.

[6] Sameera Horawalavithana, Abhishek Bhattacharjee, Renhao Liu, Nazim Choudhury, Lawrence O. Hall, and Adriana Iamnitchi. Mentions of security vulnerabilities on reddit, twitter and github. In *IEEE/WIC/ACM International Conference on Web Intelligence*, pages 200–207, 2019.

[7] Triet Huynh Minh Le, David Hin, Roland Croft, and M Ali Babar. Deepcva: Automated commit-level vulnerability assessment with deep multi-task learning. In *36th IEEE/ACM International Conference on Automated Software Engineering (ASE)*, pages 717–729. IEEE, 2021.

[8] Frank Li and Vern Paxson. A large-scale empirical study of security patches. In *ACM SIGSAC Conference on Computer and Communications Security*, pages 2201–2215, 2017.

[9] Henning Perl, Sergej Dechand, Matthew Smith, Daniel Arp, Fabian Yamaguchi, Konrad Rieck, Sascha Fahl, and Yasemin Acar. VCCFinder: Finding potential vulnerabilities in open-source projects to assist code audits. In *22nd ACM SIGSAC Conference on Computer and Communications Security*, pages 426–437, 2015.

[10] Peter Pickerill, Heiko Joshua Jungen, Mirosław Ochodek, Michał Maćkowiak, and Miroslaw Staron. PHANTOM: Curating github for engineered software projects using time-series clustering. *Empirical Software Engineering*, 25(4):2897–2929, 2020.

[11] Mingxin Sun, Wenjie Wang, Hantao Feng, Hongu Sun, and Yuqing Zhang. Identify vulnerability fix commits automatically using hierarchical attention network. *EAI Endorsed Transactions on Security and Safety*, 7(23):e2, 2020.

[12] Xinda Wang, Shu Wang, Pengbin Feng, Kun Sun, and Sushil Jajodia. PatchDB: A large-scale security patch dataset. In *51st Annual IEEE/IFIP International Conference on Dependable Systems and Networks (DSN)*, pages 149–160. IEEE, 2021.

[13] Xinda Wang, Shu Wang, Pengbin Feng, Kun Sun, Sushil Jajodia, Sanae Benchaaboun, and Frank Geck. PatchRNN: A deep learning-based system for security patch identification. In *IEEE Military Communications Conference (MILCOM)*, pages 595–600. IEEE, 2021.

[14] Yaqin Zhou, Jing Kai Siow, Chenyu Wang, Shangqing Liu, and Yang Liu. SPI: Automated identification of security patches via commits. *ACM Transactions on Software Engineering and Methodology (TOSEM)*, 31(1):1–27, 2021.

OneShot - Adaptive Encrypted Traffic Classification By Retrieval

Amir Lukach[1,2], Ran Dubin[1,2], Amit Dvir[1,2], and Chen Hajaj[3,4]

[1] Department of Computer Science, Ariel University Ariel, Israel

[2] Ariel Cyber Innovation Center, Ariel University Ariel, Israel

[3] Department of Industrial Engineering and Management, Ariel University, Israel

[4] Data Science and Artificial Intelligence Research Center, Ariel University, Israel

amir.lukach@msmail.ariel.ac.il

Abstract. Internet traffic classification's research work, tackles the classification problem from different approaches and with different goals (e.g., traffic type classification, applications and malicious/benign traffic classification, etc.). Moreover, classical machine and deep learning models have been shown to be applicable in the scope of internet traffic classification. In this paper, we introduce *OneShot*, a novel approach for internet traffic classification, by using our AKNN-based method, which allows us to effectively identify new and existing classes without retraining the model, but instead only inspecting new classes once. Our basic idea is simple, yet effective, and it is based on the idea, that if a sample's distance to an existing features is larger than a defined threshold, then we classify it as a new class.

Keywords: Malware detection, Approximate Nearest Neighbors, internet traffic classification, applications identification, security management,

1 Introduction

Recently, there has been a massive change in the internet protocols, where new network protocols such as QUIC [24], HTTP/2, HTTP/3, and new privacy-concerned protocols such as TLS 1.3 and DoH [58] have been introduced. These new privacy-preserving methods challenge traditional classification methods that rely on Deep Packet Inspection (DPI), which leverages DNS and TLS/SSL Service Name Indicator (SNI), to identify encrypted network traffic. Nowadays, these fields are no longer usable, and advanced encrypted traffic flow classification algorithms are needed [50]. With the prevalence of encrypted traffic, Internet Traffic Classification (ITC) research work, is needed more than ever that tackles the classification problem from different approaches while achieving different goals (e.g., traffic type classification, applications classification, and malicious/benign traffic classification).

Classical machine and deep learning models have been shown to be applicable in the scope of internet traffic classification [8, 13, 18, 23, 26, 34, 38, 41, 47, 48, 52, 54, 57]. Recently, a growing number of works used Natural Language Processing (NLP) [41] techniques, such as transforming the flow into a language to use word embedding [8,13,18,23,34], while others have converted the network flow into an image to harness image processing's techniques and equivalent Deep-Learning's architectures [26, 38, 47, 48, 52, 54, 57].

When examining, the general flow of a supervised ML pipeline, it is usually as follows: collecting a wide range of labeled samples, extracting a set of features from each sample, training a classification model on a subset of the samples (commonly called the "training set"), and then evaluating the model on the rest of the samples (commonly called "test set"), which were not part of the learning phase.

Internet traffic is changing all the time, and new applications, malware, protocols, and services are frequently introduced. In order to make sure, that a

internet traffic classification model will efficiently classify new types of samples correctly, there are a few requirements [39]:

- Acquire a large amount of training data for the new class
- Add it to all the dataset, which was used to train the classifier initially
- Retrain the classifier on the combined data set

Note, that retraining ML's/DL's models every time they become obsolete is both resource and time-consuming, especially when the application is complex, and the evaluated datasets are large [25]. Therefore, there is a growing need to allow classification models to detect and adapt to new classes dynamically, without retraining [49]. Nearest neighbor search is one of the most well-known tools in many research areas [11, 31]. In some cases, a generic nearest neighbor search under a suitable distance or measure of similarity offers surprising quality improvements [10]. To obtain efficient algorithms, one may use Approximate Nearest Neighbors (ANN) [4] in which the returned neighbors may be an approximation of the true nearest neighbors. Usually, this means that the answer to finding the nearest neighbors to a query point is judged by the distance of the query point to the set of its true nearest neighbors. Approximate KNN (AKNN) [3] is a variation of KNN that aims to limit the training sample number that each new test point is compared with, before returning a result.

1.1 Our Contribution

In this paper, we present an encrypted network traffic classification system (i.e. *OneShot*) based on AKNN model, which enables the system to deal with new classes. OneShot is a novel approach for internet traffic classification, that tackles the disadvantage of retraining by using a vector search instead, which allows us to effectively identify new and existing classes, by a simple yet effective method. Our novel approach is based on the idea that if a sample distance to existing

features, is higher than a defined threshold, we classify it as a new class. By that, we omit the necessity to retrain the model if new classes or instances are introduced, as the vector search process is dynamic. This way, our system can extend itself using only a few samples from new data. Furthermore, we provide real-time fast classification, since it is based on simple mathematical distance calculation of approximate k's nearest neighbors.

2 Related work

In recent years, deep-learning models have become the prominent method for network traffic classification [30,36,56,57]. The deep-learning's work span over multiple scopes and domains such as classification of the operating system, browser, and application levels [40]; mobile app identification [53,54] and IoTs [43]. Some works have converted the network flow into an image to harness image processing techniques and equivalent Deep-Learning (DL) architectures [38,47,48,52,54,57], while others used NLP [8] or graph neural network [42]. In the cyber domain, works tackle the task of malware network traffic detection (benign, malicious) and classification [2, 9, 14, 15, 21, 30, 33, 38, 44, 51, 55].

Although DL's architectures started to replace the ML for traffic classification, in [35] the authors presented a comparative experiment between ML's/DL's algorithms that shows, in some cases, that ML algorithms such as RF [29] are more than enough. It is essential for an effective classification system to support continuous model updates in order to closely monitor the network traffic landscape. All the above works tackle the classification problem without discussing the challenge of retraining the model. Retraining machine learning involves updating models to accommodate the new knowledge, which is necessary to perform well, however, most of the works use datasets with only a few classes, e.g. [12,52], they use per-flow features, and do not consider scenarios where new applications are progressively added to models. Therefore, the focus of these works, is only

on the problem of creating the most accurate classifier given immutable data for both the number of classes and the data for each class. These systems are based on creating a new training set and training a new model from scratch. It may, however, be inefficient and require high computation performance, to update them with new classes' classification.

The K-Nearest Neighbors (KNN) is one kind of lazy classification algorithm without the process of classifier training. By learning-to-hash algorithms, the KNN-based classification can be mapped to the hash table searching whose execution time and memory cost are both acceptable. Qi et al., [45] presented lightweight IoT traffic classification based on KNN [28]. Ma et al. [37] proposed a method, based on the K-nearest neighbor (KNN) algorithm, which only needs a small amount of data to train a model. Moreover, the authors also presented a three-layer classification framework for encrypted network flows. Approximate KNN (AKNN) [3] is a variation of KNN that aims to limit the training sample number that each new test point is compared with, before returning a result. Many efficient AKNN implementations have been developed with diverse approaches, such as dimension reduction [32], locality-sensitive hashing [27], and compressed sensing [5], however, non of them uses the advantages of the AKNN in the field of Internet traffic classification.

3 Methodology

Our goal is to find a solution, that provides accurate results for different types of internet traffic classification challenges. In order to achieve this goal, we present a unique, yet practical system, (i.e. OneShot), that solves two challenges.

First, it allows users and security software to accurately identify new classes, without the need for retraining the model; second, it performs a fast real-time classification with comparable results, by using AKNN query, which is based on simple mathematical euclidean distance [22] calculation.

The general idea behind our method is that if the distance between current tested vector features, to the closest class is larger than a defined threshold, then we add a new class. In our case, we use Euclidean distance metric, where we query the elastic search for K closest samples, and select the class with maximum hits' score.

In the training phase, each sample is transformed to vector with a label. The *One Shot* model saves each training sample vector index and uses the label to map it to a class. The prediction phase is responsible to label each test sample, with the matching class by using the distance from the test sample, to get k samples from the elastic and than using the label of the sample with the max hits to decide on the label. The architecture's proposed solution steps' are as follows, in the training part, each dataset comprises of traffic samples (e.g., PCAP files), where each sample represent a traffic flow. From each sample we extract the required features (see Table 1), and normalize them. In order to build our vector database (at this point, we only train and than for any sample that we want to classify we use the distance to label it), we split the data to training set (70%) and test set (30%), add the training set features' to Elastic search database [19], and then perform prediction for each test sample. The prediction is done by ANN queries from the Elastic search database as seen in Figure 1. Each ANN query, selects K closest samples, and from which we choose the class with the max hits.

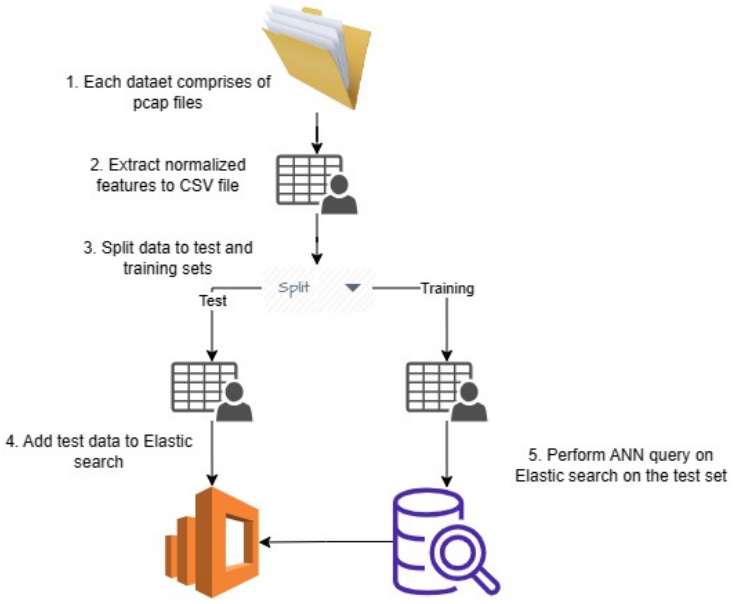


Fig. 1: One Shot Work Flow

Table 1 presents our features' vector, which contains two types of feature groups: layer 3's, and layer 4's features.

4 Experimental Design

We conducted a set of experiments to evaluate the effectiveness of the ANN-based approach for a set of classification tasks on encrypted traffic classification, from multiple known public datasets. Specifically, we looked at both malicious and benign traffic. Using the malicious traffic, we classified the malware family. For the benign traffic, we evaluated OS classification, browser classification, and the application's classification.

The goal of our evaluation was first, to assess the decrease in our classifier's performance due to using a limited number of features, and packets while clas-

Feature	Slots in vector	Description	Comment
Bits per peak	3	Summary of bits of every "peak" from dst to src	Layer 3
First packets sizes	30	First 30 packets sizes by direction of communication	Layer 3
Beaconing	20	Sum of packets size where the src is more active than the dst	Layer 4
Bandwidth	20	Min and max delta of TCP window size	Layer 4
Statistics of packets sizes	4	Min, max, mean, STD of packet sizes	Layer 3
Delta size between Pkts.	2	Mean and STD deltas of packet's size	Layer 3
Packets per second	2	Packets per second forward and backward	Layer 3
Inter-arrival time	9	Min, max and mean of bidirectional, forward and backward	Layer 4
Silence windows	1	Amount of silence windows longer than 1 second	Layer 3
Amount of ACK packets	1	Bidirectional count of TCP packets which contain ACK flag	Layer 4
Big requests	1	Count client to server messages $>$ 200b	Layer 4

Table 1: Feature sets

sifying new classes and comparing to a classical machine learning classifier (i.e., RF). Second, to show the robustness of our ANN-based classifier in classification of new classes.

4.1 Datasets

We used two common datasets for our evaluations: BOA [16], MTA [17].

The BOA dataset was presented in [16] where the authors collected the data over a period of more than two months in their lab, using a selenium web crawler for browser traffic. The dataset contains applications' traffic, such as YouTube and Facebook, labeled as browser traffic, and Dropbox and TeamViewer, labeled as non-browser traffic. The dataset contains more than 20,000 sessions. The average duration of a session was 518 seconds, and on average, each session had 520 forward packets (the average forward traffic size was 261K bytes) and 637 backward packets (the average backward traffic size was 615K bytes). Almost

all of the flows are TLS encrypted. Examples of works that used this dataset include [16, 46].

The MTA data source is a website (blog) [17] that includes many types of malware infection traffic for analysis. The website contains many types of malware, such as ransomware and exploit kits. As of 2013 to date, the blog is updated daily with relevant malware traffic, continuously adding more samples to the dataset. Using Intrusion-Detection Systems (IDS) and Antivirus software, every binary file in the PCAPs has been confirmed as malicious. Papers such as [6,30,33] have used this dataset for malware detection.

5 Results

In this section we present our experimental results, which includes multiple experiments on BOA and MTA datasets. For each dataset we first find the optimal number of packets, then calculate the minimum features' set. Moreover we present the results of our model with new samples.

5.1 AKNN Classifier's - the influence of the number of packets on BOA dataset

In the first experiment, we wanted to check the influence of the number of packets on our solution compare to RF. Therefore, we used the entire feature sets and increased the number of packets from each flow until the accuracy of our solution stopped improving. In figure 2, we show the results as a function of the number of packets, which is the motivation to use only 10 packets.

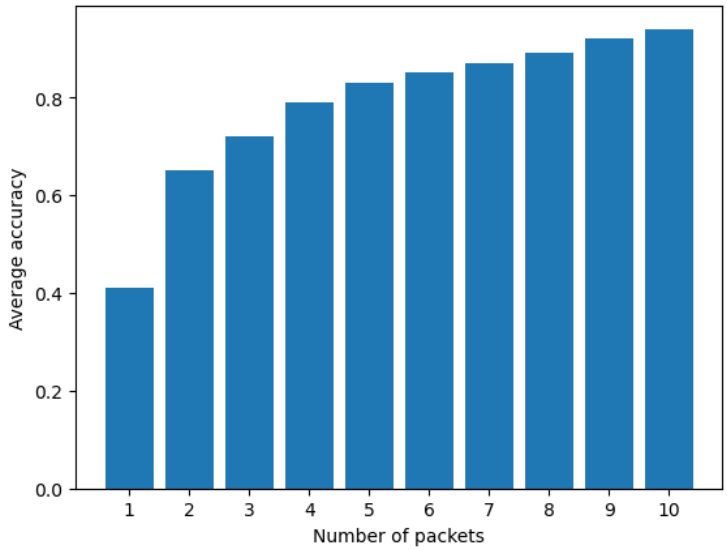


Fig. 2: One Shot Precision Plot Bar

In the BOA dataset, this occurred after 10 packets of a flow (on average around 5-6 KB). Table 2 presents the results of our AKNN model for the BOA dataset with the optimal number of processed packets (10 packets).

	Our Approach			RF		
	Prec	Rec	F1	Prec	Rec	F1
win	0.96	0.95	0.95	1	0.99	0.99
OSX	0.92	0.97	0.94	0.92	0.99	0.95
Linux	0.98	0.98	0.98	1	0.98	0.99
CR	0.91	0.90	0.91	0.96	0.98	0.97
FF	0.91	0.91	0.91	1	0.99	0.99
IE	0.97	0.97	0.97	0.97	0.98	0.97
Safari	0.95	0.97	0.96	1	1	1

Table 2: AKNN model results per Class for BOA dataset using 10 packets

From the results, we can see that RF performs slightly better than our algorithm, and this is because we use a single classifier model (ANN), and RF is an ensemble method [7].

5.2 BOA AKNN Classifier's Minimal Selected Features Maximal Performance

In this experiment we used the BOA dataset and we performed feature selection to obtain the minimal number of features, which provide maximal performance using the 10 packets from each sample. The results are depicted in Table 3. The full features' set is depicted in Table 1. As in previous section, we can see that RF performs slightly better than our algorithm.

	Prec.	**Rec.**	**F1-SC**	**RF Prec.**	**RF Rec.**	**RF F1 SC**
Win	0.96	0.95	0.95	1	0.99	0.99
OSX	0.86	0.9	0.92	0.92	0.99	0.95
Linux	0.98	0.98	0.98	1	0.98	0.99
CR	0.9	0.9	0.9	0.96	0.98	0.97
FF	0.91	0.91	0.91	1	0.99	0.99
IE	0.97	0.98	0.97	0.97	0.98	0.97
Safari	0.95	0.95	0.95	1	1	1

Table 3: AKNN model results per Class for BOA dataset using 10 packets with minimal features

5.3 MTA AKNN Classifier's Maximal Performance

In this section, similar to the first experiment, we wanted to check the influence of the number of packets on our solution. Therefore, we used the entire feature sets and increased the number of packets from each flow until the accuracy of our

solution stopped improving. We present the optimal classifier results, using 98 packets (on average around 50-60 KB) for MTA dataset in table 5. As in earlier sections, we see that RF as an ensemble method slightly outperforms our ANN classifer. Notice, that MTA consist of main class, which includes InfoStealer and Dropper, and the rest are secondary classes.

	Pr	**Rec.**	**F1-SC**	**RF Pr**	**RF Rec.**	**RF F1-SC**
Istealer	0.95	0.96	0.95	0.96	0.97	0.96
Dropper	0.93	0.92	0.92	0.95	0.95	0.95
ACC		0.96			0.97	
Dridex	0.84	0.79	0.81	0.86	0.85	0.55
Emotet	0.92	0.93	0.92	0.93	0.94	0.93
Hancitor	0.96	0.96	0.96	0.96	0.96	0.96
Icedid	0.85	0.93	0.84	0.87	0.92	0.87
Qakbot	0.92	0.94	0.93	0.92	0.94	0.93
Valak	0.92	0.94	0.93	0.92	0.94	0.94
zloader	0.86	0.88	0.87	0.87	0.88	0.87
ACC		0.93			0.95	

Table 5: AKNN model results per Class for MTA dataset using 98 packets

5.4 MTA AKNN Classifier's Minimal Selected Features Maximal Performance

In this section, we performed feature selection on MTA dataset, to obtain the minimal number of features, which provide maximal performance using 98 packets. The results are depicted in table 6. The list of features is depicted in table 7.

Class	**Pr**	**Rec.**	**F1-SC**	**RF Pr**	**RF Rec.**	**RF F1-SC**
Infostealer	0.95	0.96	0.95	0.95	0.97	0.95
Dropper	0.93	0.92	0.92	0.94	0.94	0.94
Accuracy	0.96					
Dridex	0.84	0.79	0.81	0.85	0.82	0.82
Emotet	0.92	0.93	0.93	0.93	0.93	0.93
Hancitor	0.95	0.95	0.95	0.95	0.95	0.95
Icedid	0.85	0.92	0.84	0.86	0.92	0.86
Qakbot	0.92	0.93	0.93	0.93	0.93	0.93
Valak	0.92	0.93	0.93	0.92	0.93	0.93
zloader	0.86	0.88	0.87	0.87	0.88	0.87
Accuracy	0.93					

Table 6: AKNN model minimal features' list results per Class for MTA dataset using 98 packets

Feature	**Description**
window delta 11	11th index of the max difference between packet's size sent/rcv. in the same direction
wavelet 9	9th coefficient on FFT of the packet sizes
ps 29	Packet's size of the 29th packet in the same direction
ps 6	Packet's size of the 6th packet our of first 30 packets in the same direction
wavelet 12	Using the 12th coefficient on FFT of the packet sizes in a 10 seconds window
window delta 10	10th index of the max difference between packet's size sent/received
wavelet 18	Using the 18th coefficient on FFT of the packet sizes in a 10 seconds window
bidirectional mean piat ms	average arrival time between packets in both directions
wavelet 5	Using the 5th coefficient on FFT of the packet sizes in a 10 seconds window
ps 4	Packet's size of the 4th packet our of first 30 packets in the same direction

Table 7: Feature selection for MTA

5.5 MTA AKNN One-Shot New Classes Classification

So far we tested our approach on known classes. In the following section, we present the results of our approach in the case of new classes of malware (i.e. Cobalt Strike), first seen by the model. The result is shown in table 8. From the results, we can see there is a decrease in the accuracy when testing samples from new classes, however, all the classes are classified accurately.

	F1-Score (before adding new samples)	**F1-Score**
Beacon		0.95
Infostealer	0.97	0.96
Dropper	0.92	0.91
Cobalt Strike		0.95
Dridex	0.81	0.79
Emotet	0.93	0.93
Hancitor	0.97	0.97
Icedid	0.84	0.82
Qakbot	0.93	0.93
Valak	0.93	0.93
zloader	0.87	0.84

Table 8: AKNN model for MTA's results per Class for Cobalt Strike

6 Discussions and Future Perspectives

Encrypted traffic classification is an invaluable part of cybersecurity, since network traffic encryption has become prevalent. With the growing usage of QUIC, it will not be possible to use Server Name Identification (SNI) field to identify traffic due to privacy-preserving methods. Therefore, this enhances the need for statistic-based network classification based on OSI model's network levels non-encrypted layers. In this paper, we have shown how our approach can achieve

the following goals, detect malware activity on encrypted network traffic, and classifying the malware type and name, create a simple model which can be maintained easily, detect unknown traffic, and classify it as new classes, create an alternative solution for retraining. Our plans for future work include:

1. Features selection for behavioral features needed for malware classification
2. Features selection for behavioral features needed for application classification
3. Features selection for behavioral features needed for application type classification (chat, VoIP [20], data download, VOD [1], etc.)
4. Understanding the impact of the features order, on the classification's results.
5. Optimize runtime performance.
6. Use *OneShot*'s classification on additional datasets
7. Benchmark our solution VS additional ML's/DL's methods

7 Acknowledgement

This work was supported by the Ariel Cyber Innovation Center in conjunction with the Israel National Cyber Directorate in the Prime Minister's Office.

References

1. AFOLABI, O.: What is video on demand (vod) streaming and how does it work? (2022), `https://www.makeuseof.com/what-is-video-on-demand-how-it-works/`
2. Anderson, B., McGrew, D.A.: Identifying encrypted malware traffic with contextual flow data. In: Freeman, D.M., Mitrokotsa, A., Sinha, A. (eds.) Identifying Encrypted Malware Traffic with Contextual Flow Data (2016)
3. Apache: Approximate nearest neighbors algorithm (2023). https://doi.org/https://ignite.apache.org/docs/latest/machine-learning/binary-classification/ann

4. Aumüller, M., Bernhardsson, E., Faithfull, A.: Ann-benchmarks: A benchmarking tool for approximate nearest neighbor algorithms. Information Systems **87**, 101374 (2020)
5. Aumüller, M., Bernhardsson, E., Faithfull, A.J.: Ann-benchmarks: A benchmarking tool for approximate nearest neighbor algorithms. CoRR **abs/1807.05614** (2018), http://arxiv.org/abs/1807.05614
6. Bader, O., Lichy, A., Hajaj, C., Dubin, R., Dvir, A.: Maldist: From encrypted traffic classification tomalware traffic detection and classification. In: Consumer Communications & Networking Conference (CCNC), IEEE Annual. IEEE (2022)
7. Ballings, M., Van den Poel, D., Hespeels, N., Gryp, R.: Evaluating multiple classifiers for stock price direction prediction. Expert systems with Applications **42**(20), 7046–7056 (2015)
8. Bar, R., Hajaj, C.: Simcse for encrypted traffic detection and zero-day attack detection. IEEE Access (2022)
9. Bekerman, D., Shapira, B., Rokach, L., Bar, A.: Unknown malware detection using network traffic classification. In: 2015 IEEE Conference on Communications and Network Security, CNS 2015, Florence, Italy, September 28-30, 2015. pp. 134–142. IEEE (2015)
10. Boytsov, L., Novak, D., Malkov, Y., Nyberg, E.: Off the beaten path: Let's replace term-based retrieval with k-nn search. In: Proceedings of the 25th ACM International on Conference on Information and Knowledge Management. p. 1099–1108. CIKM '16, Association for Computing Machinery, New York, NY, USA (2016). https://doi.org/10.1145/2983323.2983815, https://doi.org/10.1145/2983323.2983815
11. Bustos, B., Deussen, O., Hiller, S., Keim, D.: A graphics hardware accelerated algorithm for nearest neighbor search. In: Computational Science–ICCS 2006: 6th International Conference, Reading, UK, May 28-31, 2006, Proceedings, Part IV 6. pp. 196–199. Springer (2006)
12. Chen, T.: All versus one: an empirical comparison on retrained and incremental machine learning for modeling performance of adaptable software. In: 2019 IEEE/ACM 14th International Symposium on Software Engineering for Adaptive and Self-Managing Systems (SEAMS). pp. 157–168. IEEE (2019)
13. Corizzo, R., Zdravevski, E., Russell, M., Vagliano, A., Japkowicz, N.: Feature extraction based on word embedding models for intrusion detection in network traffic.

Journal of Surveillance, Security and Safety **1**(2), 140–150 (2020)

14. D., A., K.A., V.K., S., S.C., P., V.: Malware traffic classification using principal component analysis and artificial neural network for extreme surveillance. Computer Communications **147**, 50–57 (2019). https://doi.org/https://doi.org/10.1016/j.comcom.2019.08.003, https://www.sciencedirect.com/science/article/pii/S0140366419306693
15. Demontis, A., Melis, M., Biggio, B., Maiorca, D., Arp, D., Rieck, K., Corona, I., Giacinto, G., Roli, F.: Yes, machine learning can be more secure! a case study on android malware detection. IEEE Transactions on Dependable and Secure Computing (2017)
16. Dubin, R., Dvir, A., Pele, O., Muehlstein, J., Zion, Y., Bahumi, M., Kirshenboim, I.: Analyzing https encrypted traffic to identify user's operating system, browser and application. In: IEEE Consumer Communications and Networking Conference. IEEE (Jun 2017)
17. Duncan, B.: Malware traffic analysis (2021), https://www.malware-traffic-analysis.net/
18. Dvir, A., Marnerides, A.K., Dubin, R., Golan, N., Hajaj, C.: Encrypted video traffic clustering demystified. Comput. Secur. **96**, 101917 (2020)
19. elastic: What is elastic search? (2023), https://www.elastic.co/what-is/elasticsearch
20. fcc: Voice over internet protocol (2023), https://www.fcc.gov/general/voice-over-internet-protocol-voip
21. Gibert, D., Mateu, C., Planes, J.: The rise of machine learning for detection and classification of malware: Research developments, trends and challenges. J. Netw. Comput. Appl. **153**, 102526 (2020)
22. Gokte, S.A.: Most popular distance metrics used in knn and when to use them (2023), https://www.kdnuggets.com/2020/11/most-popular-distance-metrics-knn.html
23. Goodman, E.L., Zimmerman, C., Hudson, C.: Packet2vec: Utilizing word2vec for feature extraction in packet data. arXiv preprint arXiv:2004.14477 (2020)
24. Google: Quic, a multiplexed transport over udp (2023). https://doi.org/https://www.chromium.org/quic/
25. Horchulhack, P., Viegas, E.K., Santin, A.O.: Toward feasible machine learning model updates in network-based intrusion detection. Computer Networks **202**,

108618 (2022). https://doi.org/https://doi.org/10.1016/j.comnet.2021.108618, https://www.sciencedirect.com/science/article/pii/S1389128621005120

26. Horowicz, E., Shapira, T., Shavitt, Y.: A few shots traffic classification with mini-flowpic augmentations. In: Proceedings of the 22nd ACM Internet Measurement Conference. pp. 647–654 (2022)
27. Huang, Q., Feng, J., Zhang, Y., Fang, Q., Ng, W.: Query-aware locality-sensitive hashing for approximate nearest neighbor search. Proceedings of the VLDB Endowment **9**(1), 1–12 (2015)
28. IBM: K-nearest neighbors algorithm (2023). https://doi.org/https://www.ibm.com/topics/knn
29. IBM: What is random forest? (2023), https://www.ibm.com/topics/random-forest
30. Kim, D., Han, J., Lee, J., Roh, H., Lee, W.: Poster: Feasibility of malware traffic analysis through tls-encrypted flow visualization. In: ICNP 2020, Madrid, Spain, October 13-16. pp. 1–2. IEEE (2020)
31. Korn, F., Sidiropoulos, N., Faloutsos, C., Siegel, E., Protopapas, Z.: Fast nearest neighbor search in medical image databases. Tech. rep. (1998)
32. Kramer, O.: Dimensionality reduction with unsupervised nearest neighbors, vol. 51. Springer (2013)
33. Letteri, I., Penna, G.D., Vita, L.D., Grifa, M.T.: Mta-kdd'19: A dataset for malware traffic detection. In: Loreti, M., Spalazzi, L. (eds.) CEUR Workshop Proceeding, Italy, February 4-7. vol. 2597, pp. 153–165 (2020)
34. Li, J., Zhang, H., Wei, Z.: The weighted word2vec paragraph vectors for anomaly detection over http traffic. IEEE Access **8**, 141787–141798 (2020)
35. Lichy, A., Bader, O., Dubin, R., Dvir, A., Hajaj, C.: When a rf beats a cnn and gru, togetherâ€"a comparison of deep learning and classical machine learning approaches for encrypted malware traffic classification. Computers & Security **124**, 103000 (2023). https://doi.org/https://doi.org/10.1016/j.cose.2022.103000, https://www.sciencedirect.com/science/article/pii/S0167404822003923
36. Liu, C., He, L., Xiong, G., Cao, Z., Li, Z.: Fs-net: A flow sequence network for encrypted traffic classification. In: INFOCOM Paris, France, April 29 - May 2. pp. 1171–1179 (2019)
37. cheng Ma, C., Du, X., Cao, L.: Improved knn algorithm for fine-grained classification of encrypted network flow (2020). https://doi.org/https://www.mdpi.com/2079-9292/9/2/324

38. Marín, G., Casas, P., Capdehourat, G.: Deepmal - deep learning models for malware traffic detection and classification. CoRR **abs/2003.04079** (2020)
39. Moschitti, A.: Updating neural networks to recognize new categories, with minimal retraining, https://www.amazon.science/blog/updating-neural-networks-to-recognize-new-categories-with-minimal-retraining
40. Muehlstein, J., Zion, Y., Bahumi, M., Kirshenboim, I., Dubin, R., Dvir, A., Pele, O.: Analyzing HTTPS encrypted traffic to identify user's operating system, browser and application. In: CCNC, Las Vegas, NV, USA, January 8-11. pp. 1–6 (2017)
41. Oracle: Quic transport protocol rfc9000 (2023), https://www.oracle.com/il-en/artificial-intelligence/what-is-natural-language-processing/
42. Pang, B., Fu, Y., Ren, S., Wang, Y., Liao, Q., Jia, Y.: Cgnn: Traffic classification with graph neural network (2021). https://doi.org/10.48550/ARXIV.2110.09726, https://arxiv.org/abs/2110.09726
43. Pinheiro, A.J., de M. Bezerra, J., Burgardt, C.A., Campelo, D.R.: Identifying iot devices and events based on packet length from encrypted traffic. Computer Communications **144**, 8–17 (2019). https://doi.org/https://doi.org/10.1016/j.comcom.2019.05.012, https://www.sciencedirect.com/science/article/pii/S0140366419300052
44. de la Puerta, J.G., Pastor-López, I., Sanz, B., Bringas, P.G.: Network traffic analysis for android malware detection. In: HAIS. Lecture Notes in Computer Science, vol. 11734, pp. 468–479 (2019)
45. Qi, H., Wang, J., Li, W., Wang, Y., Qiu, T.: A blockchain-driven iiot traffic classification service for edge computing. IEEE Internet of Things Journal **8**(4), 2124–2134 (2021). https://doi.org/10.1109/JIOT.2020.3035431
46. Rezaei, S., Liu, X.: How to achieve high classification accuracy with just a few labels: A semi-supervised approach using sampled packets. CoRR (2018)
47. Rezaei, S., Liu, X.: Deep learning for encrypted traffic classification: An overview. IEEE Commun. Mag. **57**(5), 76–81 (2019)
48. Salman, O., Elhajj, I.H., Kayssi, A.I., Chehab, A.: Data representation for CNN based internet traffic classification: a comparative study. Multim. Tools Appl. **80**(11), 16951–16977 (2021)
49. Schulz, J., Veal, C., Buck, A., Anderson, D., Keller, J., Popescu, M., Scott, G., Ho, D.K., Wilkin, T.: Extending deep learning to new classes without retraining. In:

Detection and Sensing of Mines, Explosive Objects, and Obscured Targets XXV. vol. 11418, pp. 13–26. SPIE (2020)

50. Serpanos, D., Wolf, T.: Transport layer systems (2011), https://www.sciencedirect.com/topics/computer-science/flow-classification
51. Shabtai, A., Tenenboim-Chekina, L., Mimran, D., Rokach, L., Shapira, B., Elovici, Y.: Mobile malware detection through analysis of deviations in application network behavior. Comput. Secur. **43**, 1–18 (2014)
52. Shapira, T., Shavitt, Y.: Flowpic: A generic representation for encrypted traffic classification and applications identification. IEEE Trans. Netw. Serv. Manag. **18**(2), 1218–1232 (2021)
53. Taylor, V.F., Spolaor, R., Conti, M., Martinovic, I.: Robust smartphone app identification via encrypted network traffic analysis. IEEE Transactions on Information Forensics and Security **13**(1), 63–78 (Jan 2018)
54. Wang, P., Chen, X., Ye, F., Sun, Z.: A survey of techniques for mobile service encrypted traffic classification using deep learning. IEEE Access **7**, 54024–54033 (2019)
55. Wang, S., Yan, Q., Chen, Z., Yang, B., Zhao, C., Conti, M.: Detecting android malware leveraging text semantics of network flows. IEEE Trans. Inf. Forensics Secur. **13**(5), 1096–1109 (2018)
56. Wang, W., Zhu, M., Wang, J., Zeng, X., Yang, Z.: End-to-end encrypted traffic classification with one-dimensional convolution neural networks. In: ISI, Beijing, China, July 22-24. pp. 43–48. IEEE (2017)
57. Wang, W., Zhu, M., Zeng, X., Ye, X., Sheng, Y.: Malware traffic classification using convolutional neural network for representation learning. In: ICOIN. pp. 712–717 (2017)
58. Wikipedia: Dns over https (2023), https://en.wikipedia.org/wiki/DNS_over_HTTPS

MULTIFACTOR SEQUENTIAL DISENTANGLEMENT VIA STRUCTURED KOOPMAN AUTOENCODERS

Nimrod Berman*, **Ilan Naiman***, **Omri Azencot**
Department of Computer Science
Ben-Gurion University of the Negev
{bermann,naimani}@post.bgu.ac.il, azencot@cs.bgu.ac.il

ABSTRACT

Disentangling complex data to its latent factors of variation is a fundamental task in representation learning. Existing work on sequential disentanglement mostly provides two factor representations, i.e., it separates the data to time-varying and time-invariant factors. In contrast, we consider *multifactor* disentanglement in which multiple (more than two) semantic disentangled components are generated. Key to our approach is a strong inductive bias where we assume that the underlying dynamics can be represented linearly in the latent space. Under this assumption, it becomes natural to exploit the recently introduced Koopman autoencoder models. However, disentangled representations are not guaranteed in Koopman approaches, and thus we propose a novel spectral loss term which leads to structured Koopman matrices and disentanglement. Overall, we propose a simple and easy to code new deep model that is fully unsupervised and it supports multifactor disentanglement. We showcase new disentangling abilities such as swapping of individual static factors between characters, and an incremental swap of disentangled factors from the source to the target. Moreover, we evaluate our method extensively on two factor standard benchmark tasks where we significantly improve over competing unsupervised approaches, and we perform competitively in comparison to weakly- and self-supervised state-of-the-art approaches. The code is available at GitHub.

1 INTRODUCTION

Representation learning deals with the study of encoding complex and typically high-dimensional data in a meaningful way for various downstream tasks (Goodfellow et al., 2016). Deciding whether a certain representation is better than others is often task- and domain-dependent. However, disentangling data to its underlying explanatory factors is viewed by many as a fundamental challenge in representation learning that may lead to preferred encodings (Bengio et al., 2013). Recently, several works considered two factor disentanglement of sequential data in which time-varying features and time-invariant features are encoded in two separate sub-spaces. In this work, we contribute to the latter line of work by proposing a simple and efficient unsupervised deep learning model that performs *multifactor* disentanglement of sequential data. Namely, our method disentangles sequential data to more than two semantic components.

One of the main challenges in disentanglement learning is the limited access to labeled samples, particularly in real-world scenarios. Thus, prior work on sequential disentanglement focused on unsupervised models which uncover the time-varying and time-invariant features with no available labels (Hsu et al., 2017; Li & Mandt, 2018). Specifically, two feature vectors are produced, representing the dynamic and static components in the data, e.g., the motion of a character and its identity, respectively. Subsequent works introduce two factor self-supervised models which incorporate supervisory signals and a mutual information loss (Zhu et al., 2020) or data augmentation and a contrastive penalty (Bai et al., 2021), and thus improve the disentanglement abilities of prior baseline models. Yamada et al. (2020) proposed a probabilistic model with a ladder module, allowing certain multifactor disentanglement capabilities. Still, to the best of our knowledge, the majority of existing work do not explore the problem of unsupervised multifactor sequential disentanglement.

*joint first authors

In the case of static images, multiple disentanglement approaches have been proposed (Kulkarni et al., 2015; Higgins et al., 2017; Kim & Mnih, 2018; Chen et al., 2018; 2016; Burgess et al., 2018; Kumar et al., 2017; Bouchacourt et al., 2018). In addition, there are several approaches that support disentanglement of the image to multiple distinct factors. For instance, Li et al. (2020) design an architecture which learns the shape, pose, texture and background of natural images, allowing to generate new images based on combinations of disentangled factors. In (Xiang et al., 2021), the authors introduce a weakly-supervised framework where N factors can be disentangled, given $N-1$ labels. In comparison, our approach is fully unsupervised, deals with sequential data and the number of distinct components is determined by a hyperparameter.

Recently, Locatello et al. (2019) showed that unsupervised disentanglement is impossible without inductive biases on models and datasets. While exploiting the underlying temporal structure had been shown as a strong inductive bias in existing disentanglement approaches, we argue in this work that a stronger assumption should be considered. Specifically, based on Koopman theory (Koopman, 1931) and practice (Budišić et al., 2012; Brunton et al., 2021), we assume that there exists a learnable representation where the dynamics of input sequences becomes *linear*. Namely, the temporal change between subsequent latent feature vectors can be encoded with a matrix that approximates the *Koopman operator*. Indeed, the same assumption was shown to be effective in challenging scenarios such as fluid flows (Rowley et al., 2009) as well as other application domains (Rustamov et al., 2013; Kutz et al., 2016). However, it has been barely explored in the context of disentangled representations.

In this paper, we design an autoencoder network (Hinton & Zemel, 1993) that is similar to previous Koopman methods (Takeishi et al., 2017; Morton et al., 2018), and which facilitates the learning of linear temporal representations. However, while the dynamics is encoded in a Koopman operator, disentanglement is *not* guaranteed. To promote disentanglement, we make the following key observation: eigenvectors of the approximate Koopman operator represent time-invariant and time-variant factors. Motivated by this understanding, we propose a novel spectral penalty term which splits the operator's spectrum to separate and clearly-defined sets of static and dynamic eigenvectors. Importantly, our framework naturally supports multifactor disentanglement: every eigenvector represents a unique disentangled factor, and it is considered static or dynamic based on its eigenvalue.

Contributions. Our main contributions can be summarized as follows.

1. We introduce a strong inductive bias for disentanglement tasks, namely, the dynamics of input sequences can be encapsulated in a matrix. This assumption is backed by the rich Koopman theory and practice.
2. We propose a new unsupervised Koopman autoencoder learning model with a novel spectral penalty on the eigenvalues of the Koopman operator. Our approach allows straightforward multifactor disentanglement via the eigendecomposition of the Koopman operator.
3. We extensively evaluate our method on new multifactor disentanglement tasks, and on several two factor benchmark tasks, and we compare our work to state-of-the-art unsupervised and weakly-supervised techniques. The results show that our approach outperforms baseline methods in various quantitative metrics and computational resources aspects.

2 Related Work

Sequential Disentanglement. Most existing work on sequential disentanglement is based on the dynamical variational autoencoder (VAE) architecture (Girin et al., 2020). Initial attempts focused on probabilistic models that separate between static and dynamic factors, where in (Hsu et al., 2017) the joint distribution is conditioned on the mean, and in (Li & Mandt, 2018) conditioning is defined on past features. Subsequent works proposed self-supervised approaches that depend on auxiliary tasks and supervisory signals (Zhu et al., 2020), or on additional data and contrastive penalty terms (Bai et al., 2021). In Han et al. (2021a), the authors replace the common Kullback–Leibler divergence with the Wasserstein distance between distributions. Some approaches tailored to video disentanglement use generative adversarial network (GAN) architectures (Villegas et al., 2017; Tulyakov et al., 2018) and a recurrent model with adversarial loss (Denton & Birodkar, 2017). Finally, Yamada et al. (2020) proposed a variational autoencoder model including a ladder module (Zhao et al., 2017), which allows to disentangle multiple factors. The authors demonstrated qualitative results of multifactor latent traversal between various two static features and three dynamic features on the Sprites dataset.

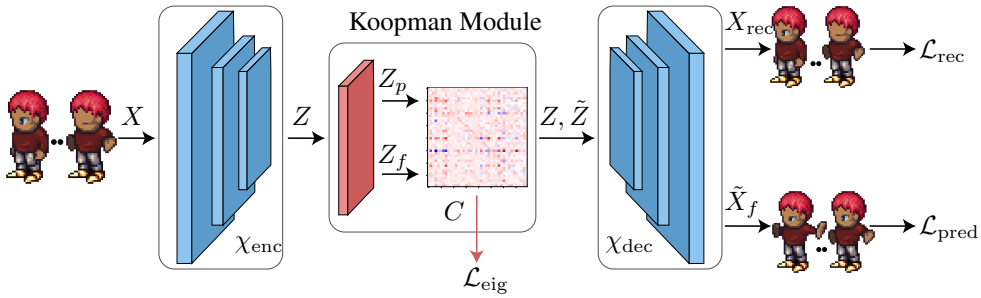


Figure 1: Our architecture is based on a Koopman autoencoder network which includes encoder χ_{enc}, decoder χ_{dec}, and a Koopman module that computes the Koopman operator C via least squares solves. We augment this model with a novel spectral penalty term $\mathcal{L}_{\text{eig}}$ which facilitates the learning of spectrally structured C matrices, and thus supporting multifactor disentanglement by construction.

Dynamics Learning. Over the past few years, an increasing interest was geared towards learning and representing dynamical systems using deep learning techniques. Two factor disentanglement methods based on Kalman filter (Fraccaro et al., 2017), and state-space models (Miladinović et al., 2019) focus on ordinary differential equation systems. Other methods utilize the mutual information between past and future to estimate predictive information Clark et al. (2019); Bai et al. (2020). Mostly related to our approach are Koopman autoencoders (Lusch et al., 2018; Yeung et al., 2019; Otto & Rowley, 2019; Li et al., 2019; Azencot et al., 2020; Han et al., 2021b), related to classical learning methods, e.g., Azencot et al. (2019); Cohen et al. (2021). Specifically, in (Takeishi et al., 2017; Morton et al., 2018; Iwata & Kawahara, 2020) the Koopman operator is learned via a least squares solve per batch, allowing to train a single neural model on multiple initial conditions. We base our architecture on the latter works, and we augment it with a novel spectral loss term which promotes disentanglement. Recently, an intricate model for video disentanglement was proposed in (Comas et al., 2021). While the authors employ Koopman techniques in that work, it is only partially related to our work since they explicitly model pose and appearance components, whereas our approach can model an arbitrary number of disentangled factors. In addition, their architecture is based on the attention network (Bahdanau et al., 2014), where the Koopman module is mostly related to prediction. In comparison, in our work the Koopman module is directly responsible for unsupervised disentanglement of sequential data.

Koopman Spectral Analysis. Our method is based on learning Koopman operators with structured spectra. Spectral analysis of Koopman operators is an active research topic (Mezić, 2013; Arbabi & Mezic, 2017; Mezic, 2017; Das & Giannakis, 2019; Naiman & Azencot, 2023). We explore Koopman eigenfunctions associated with the eigenvalue 1. These eigenfunctions are related to global stability (Mauroy & Mezić, 2016), and to orbits of the system (Mauroy & Mezić, 2013; Azencot et al., 2013; 2014). Other attempts focused on computing eigenfunctions for a known spectrum (Mohr & Mezić, 2014). Recently, pruning weights of neural networks using eigenfunctions with eigenvalue 1 was introduced in (Redman et al., 2021). However, to the best of our knowledge, our work is among a few to propose a deep learning model for generating spectrally-structured Koopman operators.

3 Koopman Autoencoder Models

We recall the Koopman autoencoder (KAE) architecture introduced in (Takeishi et al., 2017) as it is the basis of our model. The KAE model consists of an encoder and decoder modules, similarly to standard autoencoders, and in between, there is a Koopman module. The general idea behind this architecture is that the encoder and decoder are responsible to generate effective representations and their reconstructions, driven by the Koopman layer which penalizes for nonlinear encodings.

We denote by $X \in \mathbb{R}^{b\times(t+1)\times m}$ a batch of sequence data $\{x_{ij}\} \subset \mathbb{R}^m$ where $i \in \{1,\dots,b\}$ and $j \in \{1,\dots,t+1\}$ represent the batch sample and time indices, respectively. The tensor X is encoded to its latent representation $Z \in \mathbb{R}^{b\times(t+1)\times k}$ via $Z = \chi_{\text{enc}}(X)$. The Koopman layer splits the latent variables to past Z_p and future Z_f observations, and then, it finds the best linear map C such that

$Z_p \cdot C \approx Z_f$. Formally, $Z_p = (z_{ij}) \in \mathbb{R}^{b \cdot t \times k}$ for $j \in \{1, \ldots, t\}$ and any i, and $Z_f = (z_{ij}) \in \mathbb{R}^{b \cdot t \times k}$ for $j \in \{2, \ldots, t+1\}$ and any i, i.e., Z_p holds the first t latent variables per sample, and Z_f holds the last t variables. Then, $C = \arg\min_{\tilde{C}} |Z_p \cdot \tilde{C} - Z_f|_F^2 = Z_p^+ Z_f$, where A^+ denotes the pseudo-inverse of the matrix A. Importantly, the matrix C is computed per Z during both training and inference, and in particular, C is not parameterized by network weights. Additionally, the pseudo-inverse computation supports backpropagation, and thus it can be used during training (Ionescu et al., 2015). Lastly, the latent samples are reconstructed with the decoder $X_{\text{rec}} = \chi_{\text{dec}}(Z)$.

The above architecture employs reconstruction and prediction loss terms: the reconstruction loss promotes an autoencoder learning, and the prediction loss aims to capture the dynamics in C. We use the notation $\mathcal{L}_{\text{MSE}}(X, Y) = \frac{1}{b \cdot t} \sum_{i,j} |Y(i,j) - X(i,j)|_2^2$ for the average distance between tensors $X, Y \in \mathbb{R}^{b \times t \times k}$ for $i \in \{1, \ldots, b\}$ and $j \in \{1, \ldots, t\}$. Then, the losses are given by

$$\mathcal{L}_{\text{rec}}(X_{\text{rec}}, X) = \mathcal{L}_{\text{MSE}}(X_{\text{rec}}, X) \,, \tag{1}$$

$$\mathcal{L}_{\text{pred}}(\tilde{Z}_f, Z_f, \tilde{X}_f, X_f) = \mathcal{L}_{\text{MSE}}(\tilde{Z}_f, Z_f) + \mathcal{L}_{\text{MSE}}(\tilde{X}_f, X_f) \,, \tag{2}$$

where $\tilde{Z}_f := Z_p \cdot C$, $\tilde{X}_f := \chi_{\text{dec}}(\tilde{Z}_f)$, and X_f are the inputs corresponding to Z_f latent variables. The network loss is taken to be $\mathcal{L} = \lambda_{\text{rec}} \mathcal{L}_{\text{rec}} + \lambda_{\text{pred}} \mathcal{L}_{\text{pred}}$, where $\lambda_{\text{rec}}, \lambda_{\text{pred}} \in \mathbb{R}^+$ balance between the reconstruction and prediction contributions. We show in Fig. 1 an illustration of the Koopman autoencoder architecture using the notations above.

4 Multifactor Disentangling Koopman Autoencoders

How disentanglement can be achieved given the Koopman autoencoder architecture? For comparison, other disentanglement approaches typically represent the disentangled factors explicitly. In contrast the batch dynamics in KAE models is encoded in the approximate Koopman operator matrix C, where C propagates latent variables through time while carrying the static as well as dynamic information. Thus, the time-varying and time-invariant factors are still entangled in the Koopman matrix. We now show that KAE theoretically enables disentanglement under the following analysis.

Koopman disentanglement. In general, one of the key advantages of Koopman theory and practice is the linearity of the Koopman operator, allowing to exploit tools from linear analysis. Specifically, our approach depends heavily on the spectral analysis of the Koopman operator (Mezić, 2005). In what follows, we perform our analysis directly on C, and we refer the reader to App. A and the references therein for a detailed treatment of the full Koopman operator. The eigendecomposition of C consists of a set of left eigenvectors $\{\phi_i \in \mathbb{C}^k\}$ and a set of eigenvalues $\{\lambda_i \in \mathbb{C}\}$ such that

$$\phi_i^T C = \lambda_i \phi_i^T \,, \quad i = 1, \ldots, k \,. \tag{3}$$

The eigenvectors can be viewed as approximate Koopman eigenfunctions, and thus the eigenvectors hold fundamental information related to the underlying dynamics. For instance, the eigenvectors describe the temporal change in latent variables. Formally,

$$z_j^T C = \sum_{i=1}^{k} \langle z_j^T, \phi_i^T \rangle \phi_i^T C = \sum_i \bar{z}_j^i \lambda_i \phi_i^T \approx z_{j+1}^T \,, \quad j = 1, \ldots, t \,, \tag{4}$$

where $\bar{z}_j^i := \langle z_j^T, \phi_i^T \rangle$ is the projection of z_j^T on the eigenvector ϕ_i^T. The approximation follows from C being the best (and not necessarily exact) linear fit between past and future features. Moreover, it follows that predicting step $j + r$ from j is achieved simply by applying powers of the Koopman matrix on z_j^T, i.e., $z_j^T C^r = \sum_i \bar{z}_j^i \lambda_i^r \phi_i^T \approx z_{j+r}^T$.

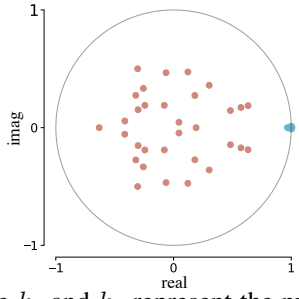
Our approach to multifactor disentanglement is based on the following key observation: *eigenvectors of the matrix C whose eigenvalue is 1 represent time-invariant factors*. For instance, assume C has a single eigenvector ϕ_1 with $\lambda_1 = 1$ and $\lambda_i \neq 1$ for $i \neq 1$, then it follows from Eq. 4 that

$$z_j^T C^r = \bar{z}_j^1 \phi_1^T + \sum_{i=2}^{k} \bar{z}_j^i \lambda_i^r \phi_i^T \,. \tag{5}$$

Essentially, the contribution of ϕ_1 is not affected by the dynamics and it remains constant, and thus the first addend remains constant throughout time, and it is related to static features of the dynamics.

In contrast, every element in the sum in Eq. 5 is scaled by its respective λ_i^r, and thus the sum changes throughout time, and these eigenvectors are related to dynamic features. We conclude that the KAE architecture virtually allows disentanglement via eigendecomposition of the Koopman matrix where the static factors are eigenvectors with eigenvalue 1, and the rest are dynamic factors.

Multifactor Koopman Disentanglement. Unfortunately, the vanilla KAE model is not suitable for disentanglement as the learned Koopman matrices can generally have arbitrary spectra, with multiple static factors or no static components at all. Moreover, KAE does not allow to explicitly balance the number of static vs. dynamic factors. To alleviate the shortcomings of KAE, we propose to augment the Koopman autoencoder with a spectral loss term $\mathcal{L}_{\text{eig}}$ which explicitly manipulates the structure of the Koopman spectrum, and its separation to static and dynamic factors. Formally,



$$\mathcal{L}_{\text{stat}} = \frac{1}{k_s}\sum_{i}^{k_s} |\lambda_i - (1+\imath 0)|^2 \,, \tag{6}$$

$$\mathcal{L}_{\text{dyn}} = \frac{1}{k_d}\sum_{i}^{k_d} \xi(|\lambda_i|,\, \epsilon) \,, \tag{7}$$

$$\mathcal{L}_{\text{eig}} = \mathcal{L}_{\text{stat}} + \mathcal{L}_{\text{dyn}} \,, \tag{8}$$

where k_s and k_d represent the number of static and dynamic components, respectively, and thus $k = k_s + k_d$. The term $\mathcal{L}_{\text{stat}}$ measures the average distance of every static eigenvalue from the complex value 1. The role of $\mathcal{L}_{\text{dyn}}$ is to encourage separation between the static and dynamic factors. In practice, this is achieved with a threshold function ξ which takes the modulus of λ_i and a user parameter $\epsilon \in (0, 1)$, and it returns $|\lambda_i|$ if $|\lambda_i| > \epsilon$, and zero otherwise. Thus, $\mathcal{L}_{\text{dyn}}$ penalizes dynamic factors whose modulus is outside an ϵ-ball. The inset figure shows an example spectrum we obtain using our loss penalties, where blue and red denote static and dynamic factors, respectively.

Method Summary. Given a batch $X \in \mathbb{R}^{b\times t\times m}$, we feed it to the encoder. Our encoder is similar to the one used in C-DSVAE (Bai et al., 2021) having five convolutional layers, followed by a uni-directional LSTM module. The output of the encoder is denoted by $Z \in \mathbb{R}^{b\times t\times k}$, and it is passed to the Koopman module. Then, Z is split to past Z_p and future Z_f observations, allowing to compute the approximate Koopman operator via $C = Z_p^+ Z_f$. In addition, we compute $\tilde{Z}_f := Z_p \cdot C$ which will be used to compute $\mathcal{L}_{\text{pred}}$. After the Koopman module, we apply the decoder whose structure mimics the encoder but in reverse having an LSTM component and de-convolutional layers. Additional details on the encoder and decoder are detailed in Tab. 5. We decode Z to obtain the reconstructed signal X_{rec}, and we decode $\tilde{Z}_f$ to approximate the future recovered signals $\tilde{X}_f$. The total loss is given by $\mathcal{L} = \lambda_{\text{rec}}\mathcal{L}_{\text{rec}} + \lambda_{\text{pred}}\mathcal{L}_{\text{pred}} + \lambda_{\text{eig}}\mathcal{L}_{\text{eig}}$, where the balance weights $\lambda_{\text{rec}}, \lambda_{\text{pred}}$ and λ_{eig} scale the loss penalty terms and the exact values are given in Tab. 6. To compute $\mathcal{L}_{\text{eig}}$, we identify the static and dynamic subspaces. This is done by simply sorting the eigenvalues based on their modulus, and taking the last k_s eigenvectors, whereas the rest k_d are dynamic factors. Identifying multiple factors is more involved and can be obtained by manual inspection or via an automatic procedure using a pre-trained classifier, see App. B.5.

Multifactor Static and Dynamic Swap. Similar to previous methods our approach allows to swap between e.g., the static factors of two different input samples. In addition, our framework naturally supports multifactor swap as we describe next. For simplicity, we first consider the swap of a single factor (e.g., hair color in Sprites (Reed et al., 2015)) for the given latent codes of two samples, $z_j(u)$ and $z_j(v), j = 1, \ldots, t+1$. Denote by ϕ_1 the eigenvector of the factor we wish to swap, then a single swap is obtained by switching the Koopman projection coefficients of ϕ_1, i.e.,

$$\hat{z}_j(u) = \bar{z}_j^1(v)\phi_1 + \sum_{i=2}^{k} \bar{z}_j^i(u)\phi_i \,, \quad \hat{z}_j(v) = \bar{z}_j^1(u)\phi_1 + \sum_{i=2}^{k} \bar{z}_j^i(v)\phi_i \,, \tag{9}$$

where $\hat{z}_j(u)$ denotes the new code of $z_j(u)$ using the swapped factor from the v sample, and similarly for $\hat{z}_j(v)$. If several factors are to be swapped, then $\hat{z}_j(u) = \sum_{i\in I} \bar{z}_j^i(v)\phi_i + \sum_{i\in I^c} \bar{z}_j^i(u)\phi_i$, where I denotes the set of eigenvector indices we swap, and I^c is the complement set. The above formulation

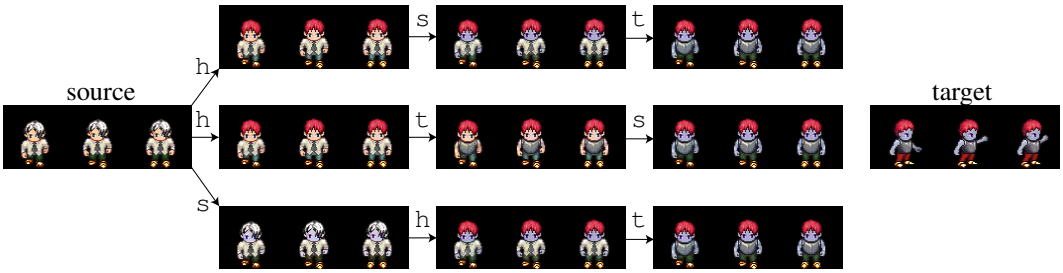


Figure 2: In the factorial swap experiment we modify individual static factors of the source character to match those of the target. The top row shows the gradual change of the hair, skin, and top colors.

is equivalent to the simpler tensor notation $\bar{Z}^s[u,\ :,\ I^c] = \bar{Z}[u,\ :,\ I^c]$ and $\bar{Z}^s[u,\ :,\ I] = \bar{Z}[v,\ :,\ I]$, where $\bar{Z} \in \mathbb{C}^{n\times t\times k}$ is the Koopman projection coefficients of a batch with n samples, and $\bar{Z}^s$ represents the swapped coefficients. Thus, the swapped latent code is given by $\hat{Z} = \bar{Z}^s \cdot \Phi$, where $\Phi = (\phi_i)$ is the matrix of eigenvectors organized in columns. A more detailed description of the swaps and how to obtain the disentangled subspaces representations is provided in App. B.

5 Results

We evaluate our model on several two- and multi-factor disentanglement tasks. For every dataset, we train our model, and for evaluation, we additionally train a vanilla classifier on the label sequences. In all experiments, we apply our model on mini-batches, extracting the latent codes Z and the Koopman matrix C. Disentanglement tests use the eigendecomposition of C, where we identify the subspaces corresponding to the dynamic and static factors, denoted by $I_{\texttt{dyn}}$ and $I_{\texttt{stat}}$, respectively. We may label other subspaces such as $I_{\texttt{h}}$ to note they correspond to e.g., hair color change in Sprites. To identify the subspace corresponding to a specific factor we perform manual or automatic approaches (App. B). Importantly, subspace's dimension of a single factor may be larger than one. We provide further details regarding the network architectures, hyperparameters, datasets, data pre-processing, and a comparison of computational resources (App. B). Additional results are provided in App. C.

5.1 Multifactor Disentanglement

We will demonstrate that our method disentangles sequential data to multiple distinct factors, and thus it extends the toolbox introduced in competitive sequential disentanglement approaches which only supports two factor disentanglement. Specifically, while prior techniques separate to static and dynamic factors, we show that our model identifies several semantic static factors, allowing a finer control over the factored items for downstream tasks. We perform qualitative and quantitative tasks on the Sprites (Reed et al., 2015) and MUG (Aifanti et al., 2010) datasets to show those advantages.

Factorial swap. This experiment demonstrates that our method is capable of swapping individual content components between sprite characters. We extract a batch with 32 samples, and we identify by manual inspection the subspaces responsible for hair color, skin color, and top color, labeled by $I_{\texttt{h}}, I_{\texttt{s}}, I_{\texttt{t}}$. We select two samples from the test batch, shown as the source and target in Fig. 2. To swap individual static factors between the source and target, we follow Eq. 9. Specifically, we gradually change the static features of the source to be those of the target. For example, the top row in Fig. 2 shows the source being modified to have the hair color, followed by skin color, and then top color of the target, from left to right. In practice, this is achieved via setting $\bar{Z}^{\texttt{h}} = \bar{Z}^{\texttt{hs}} = \bar{Z}^{\texttt{hst}} = \bar{Z}_{\text{src}}$ and assigning $\bar{Z}^{\texttt{h}}[:,\ I_{\texttt{h}}] = \bar{Z}_{\text{tgt}}[:,\ I_{\texttt{h}}]$, $\bar{Z}^{\texttt{hs}}[:,\ I_{\texttt{hs}}] = \bar{Z}_{\text{tgt}}[:,\ I_{\texttt{hs}}]$, and $\bar{Z}^{\texttt{hst}}[:,\ I_{\texttt{hst}}] = \bar{Z}_{\text{tgt}}[:,\ I_{\texttt{hst}}]$, where $\bar{Z}_{\text{src}}, \bar{Z}_{\text{tgt}} \in \mathbb{C}^{8\times 40}$ are the Koopman projection values of the source and target, respectively. The set $I_{\texttt{hs}} := I_{\texttt{h}} \cup I_{\texttt{s}}$, and similarly for $I_{\texttt{hst}}$. The tensor $\bar{Z}^{\texttt{h}}$ represents the new character obtained by borrowing the hair color of the target, and similarly for $\bar{Z}^{\texttt{hs}}$ and $\bar{Z}^{\texttt{hst}}$. In total, we demonstrate in Fig. 2 the changes: $\texttt{h}\rightarrow\texttt{s}\rightarrow\texttt{t}$ (top), $\texttt{h}\rightarrow\texttt{t}\rightarrow\texttt{s}$ (middle), and $\texttt{s}\rightarrow\texttt{h}\rightarrow\texttt{t}$ (bottom). We additionally show in Fig. 12 an example of individual swaps including all possible combinations. Our results display good multifactor separation and transfer of individual static factors between different characters.

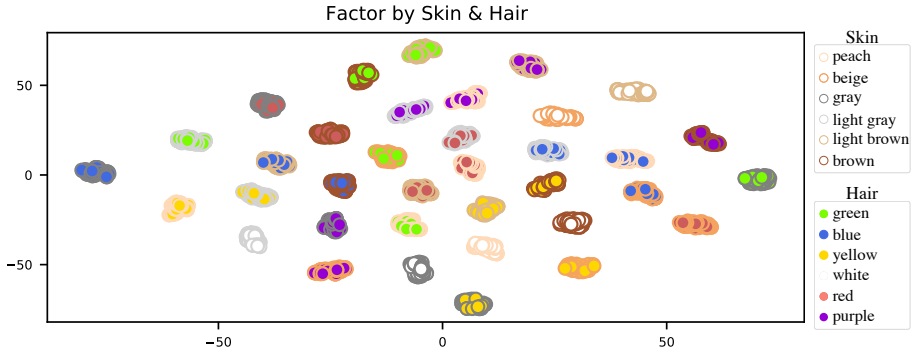


Figure 3: We show the t−SNE plot of the 4D Koopman static subspace which encodes the skin and hair colors. The embedding perfectly clusters all (skin, hair) color combinations.

To quantitatively assess the performance of our approach in the factorial swap task, we consider the following experiment. We iterate over test batches of size 256, and for every batch we *automatically* identify its hair color and skin color subspaces, $I_{\mathrm{h}}, I_{\mathrm{s}}$. Then, we compute a random sampling of Z denoted by J, and separately swap the hair color and the skin color. In practice, this boils down to $\bar{Z}^{\mathrm{h}} = \bar{Z}^{\mathrm{s}} = \bar{Z}$ and setting $\bar{Z}^{\mathrm{h}}[:, :, I_{\mathrm{h}}] = \bar{Z}[J, :, I_{\mathrm{h}}]$ and similarly, $\bar{Z}^{\mathrm{s}}[:, :, I_{\mathrm{s}}] = \bar{Z}[J, :, I_{\mathrm{s}}]$. The new latent codes are reconstructed and fed to the pre-trained classifier, and we compare the predicted labels to the true labels of $Z[J]$. The results are reported in Tab. 1 where we list the accuracy measures for every factor. For most non-swapped factors, we obtain an accuracy score close to random guess, e.g., the skin accuracy in the hair swap is 16.25% which is very close to $1/6$. Moreover, the swapped factors yield high accuracy scores marked in bold, validating the successful swap of individual factors.

Table 1: Accuracy measures of factorial swap experiments.

Test	action	skin	top	pants	hair
hair swap	10.51%	16.25%	16.33%	35.51%	**90.59%**
skin swap	10.55%	**73.01%**	16.29%	30.55%	17.70%

Latent Embedding. We now explore the effect of our model on the latent representation of samples. To this end, we consider a batch X of sprites where the motion, skin and hair colors are arbitrary, and the top and pants colors are fixed, for a total of 324 examples. Following the above experiment, we automatically identify the subspaces responsible for changing the hair and skin color, $I_{\mathrm{h}}, I_{\mathrm{s}}$. To explore the distribution of the latent code, we visualize the Koopman projection coefficients of the 4-dimensional subspace $I_{\mathrm{hs}} = I_{\mathrm{h}} \cup I_{\mathrm{s}}$ given by $\bar{Z}[:, :, I_{\mathrm{hs}}] \in \mathbb{C}^{324\times 8\times 4}$. We plot in Fig. 3 the 2D embedding obtained using t−SNE (Van der Maaten & Hinton, 2008). To distinguish between skin and hair labels, we paint the face of every 2D point based on its true hair label, and we paint the point's edge with the true skin color. The plot resembles a grid-like pattern, showing a *perfect* separation to all 36 unique combinations of (skin, hair) colors. We conclude that the Koopman subspace I_{hs} indeed disentangles the samples based on either their skin or hair.

Incremental Swap. In this test we explore multifactor features of time-varying Koopman subspaces on the MUG dataset. Given a source image u, we gradually modify its dynamic factors to be those of the target v. In practice, we compute $\bar{Z}[u, :, I_{\mathrm{q}}] = \bar{Z}[v, :, I_{\mathrm{q}}]$, where $I_q \subset I_{\mathrm{dyn}}$ is an index set from I_{dyn} such that $q \in \{1, 2, 3\}$ and $I_1 \subset I_2 \subset I_3 \subset I_{\mathrm{dyn}}$. Specifically, $|I_1| = 4, |I_2| = 6, |I_3| = 32$. Fig. 4 shows the incremental swap results of two examples changing from disgust to happiness (left), and happiness to anger (right). The three rows below the source row are the reconstructions of the gradual swap denoted by $\tilde{X}(I_{\mathrm{q}}) := \chi_{\mathrm{dec}}(\bar{Z}[u, :, I_q] \cdot \Phi)$. Our results demonstrate in both cases a non-trivial gradual change from the source expression to the target, as more dynamic features are swapped. For instance, the left source is mapped to a smiling character over all time samples in $\tilde{X}(I_2)$, and then it is fixed to better match the happiness trajectory source in $\tilde{X}(I_3)$.

Figure 4: Our method allows to swap the dynamic features incrementally, and thus it achieves a relatively smooth transition between the source and target expressions.

5.2 Two Factor Disentanglement of Image Data

We perform two factor disentanglement on Sprites and MUG datasets, and we compare with state-of-the-art methods. Evaluation is performed by fixing the time-varying features of a test batch while randomly sampling its time-invariant features. Then, a pre-trained classifier generates predicted labels for the new samples while comparing them to the true labels. We use metrics such as accuracy (Acc), inception score (IS), intra-entropy $H(y|x)$ and inter-entropy $H(y)$ (Bai et al., 2021). We extract batches of size 256, and we identify their static and dynamic subspaces automatically. In contrast to most existing work, our approach is not based on a variational autoencoder model, and thus the sampling process in our approach is performed differently. Specifically, for every test sequence, we randomly sample static features by generating a new latent code based on a random sampling in the convex hull of the batch. That is, we generate random coefficients $\{\alpha_i\}$ for every sample in the batch such that they form a partition of unity and $\alpha_i \in [0, 1]$. Then, we swap the static features of the batch with those of the new samples, $\bar{Z}[:, :, I_{\texttt{stat}}] = \sum_i \alpha_i \bar{Z}[i, :, I_{\texttt{stat}}]$. We perform 300 epochs of random sampling, and we report the average results in Tab. 2, 3. Notably, our method outperforms previous SOTA methods on the Sprites dataset across all metrics. On the MUG dataset, we achieve competitive accuracy results and better results on IS and $H(y|x)$ metrics. In comparison to unsupervised methods MoCoGAN, DSVAE and R-WAE, our results are the best on all metrics.

5.3 Two Factor Disentanglement of Audio Data

We additionally evaluate our model on a different data modality, utilizing a benchmark downstream speaker verification task (Hsu et al., 2017) on the TIMIT dataset (Garofolo et al., 1992). In this task, we aim to distinguish between speakers, independently of the text they read. We compute for each test sample its latent representation Z, and its dynamic and static sub-representations $Z_{\texttt{dyn}}, Z_{\texttt{stat}}$, respectively. In an ideal two factor disentanglement, we expect $Z_{\texttt{stat}}$ to encode the speaker identity, whereas $Z_{\texttt{dyn}}$ should be agnostic to this data. To quantify the disentanglement we employ the Equal Error Rate (EER) test. Namely, we compute the cosine similarity between all pairs of latent sub-representations in $Z_{\texttt{stat}}$. The pair is assumed to encode the same speaker if their cosine similarity is higher than a threshold $\epsilon \in [0, 1]$, and the pair has different speakers otherwise. The threshold ϵ needs to be calibrated to receive the EER (Chenafa et al., 2008). If $Z_{\texttt{stat}}$ indeed holds the speaker identity,

Table 2: Disentanglement metrics on Sprites.

Method	Acc↑	IS↑	$H(y\|x)$↓	$H(y)$↑
MoCoGAN	92.89%	8.461	0.090	2.192
DSVAE	90.73%	8.384	0.072	2.192
R-WAE	98.98%	8.516	0.055	2.197
S3VAE	99.49%	8.637	0.041	2.197
C-DSVAE	99.99%	8.871	0.014	2.197
Ours	**100%**	**8.999**	**1.6e−7**	**2.197**

Table 3: Disentanglement metrics on MUG.

Method	Acc↑	IS↑	$H(y\|x)$↓	$H(y)$↑
MoCoGAN	63.12%	4.332	0.183	1.721
DSVAE	54.29%	3.608	0.374	1.657
R-WAE	71.25%	5.149	0.131	1.771
S3VAE	70.51%	5.136	0.135	1.760
C-DSVAE	**81.16%**	5.341	0.092	**1.775**
Ours	77.45%	**5.569**	**0.052**	1.769

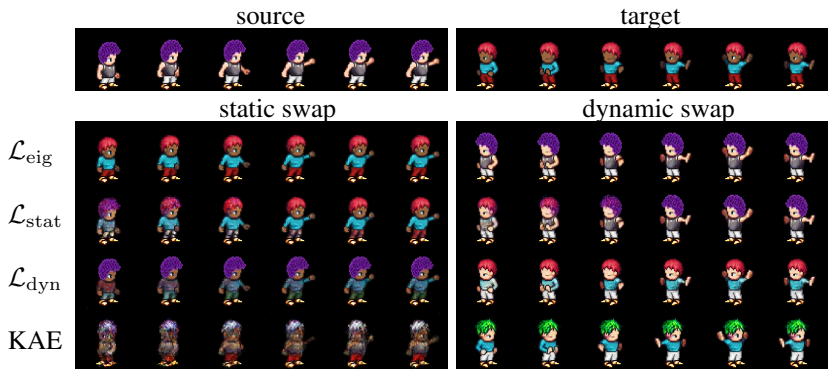


Figure 5: Our ablation study shows that the full model $\mathcal{L}_{\text{eig}}$ disentangles data well, whereas models using only $\mathcal{L}_{\text{stat}}$ loss or only $\mathcal{L}_{\text{dyn}}$ loss or no $\mathcal{L}_{\text{eig}}$ loss at all struggle with swapping static features.

then its EER score should be low. The same test is also repeated on Z_{dyn} for which we expect high EER scores as it should not contain speaker information.We report the results in Tab. 4. Our method achieves the third best overall EER on the static and dynamic tests. However, S3VAE and C-DSVAE either use significantly more data or self-supervision signals. We label by C-DSVAE* and C-DSVAE† the approach C-DSVAE without content and dynamic augmentation, respectively. When comparing to unsupervised approaches that do not use additional data (FHVAE, DSVAE, and R-WAE), we achieve the best results with a margin of 0.27% and 3.37% static and dynamic, respectively.

Table 4: Disentanglement metrics on TIMIT.

Method	FHVAE	DSVAE	R-WAE	S3VAE	C-DSVAE*	C-DSVAE†	C-DSVAE	Ours
Static EER↓	5.06%	5.65%	4.73%	5.02%	5.09%	4.31%	**4.03%**	4.46%
Dynamic EER↑	22.77%	19.20%	23.41%	25.51%	24.30%	31.09%	**31.81%**	26.78%

5.4 Ablation Study

We train different models to evaluate the effect of our loss term on the KAE architecture: full model with $\mathcal{L}_{\text{eig}}$, KAE + $\mathcal{L}_{\text{stat}}$, KAE + $\mathcal{L}_{\text{dyn}}$, and baseline KAE without $\mathcal{L}_{\text{eig}}$. All other parameters are left fixed. In Fig. 5, we show a qualitative example of static and dynamic swaps between the source and the target. Each of the bottom four rows in the plot is associated with a different model. The full model ($\mathcal{L}_{\text{eig}}$) yields clean disentanglement results on both swaps. In contrast, the static features are not perfectly swapped when removing the dynamic penalty ($\mathcal{L}_{\text{stat}}$). Moreover, the model without static loss ($\mathcal{L}_{\text{dyn}}$) does not swap the static features at all. Finally, the baseline KAE model generates somewhat random samples. We note that in all cases (even for the KAE model), the motion is swapped relatively well which can be attributed to the good encoding of the dynamics via the Koopman matrix.

6 Discussion

We have proposed a novel approach for multifactor disentanglement of sequential data, extending existing two factor methods. Our model is based on a strong inductive bias where we assumed that the underlying dynamics can be encoded linearly. The latter assumption calls for exploiting recent Koopman autoencoders which we further enhance with a novel spectral loss term, leading to an effective disentangling model. Throughout an extensive evaluation, we have shown new disentanglement sequential tasks such as factorial swap and incremental swap. In addition, our approach achieves state-of-the-art results on two factor tasks in comparison to baseline unsupervised approaches, and it performs similarly to self-supervised and weakly-supervised techniques.

There are multiple directions for future research. First, our approach is complementary to most existing VAE approaches, and thus merging features of our method with variational sampling, mutual information and contrastive losses could be fruitful. Second, theoretical aspects such as disentanglement guarantees could be potentially shown in our framework using Koopman theory.

ACKNOWLEDGEMENTS

This research was partially supported by the Lynn and William Frankel Center of the Computer Science Department, Ben-Gurion University of the Negev, an ISF grant 668/21, an ISF equipment grant, and by the Israeli Council for Higher Education (CHE) via the Data Science Research Center, Ben-Gurion University of the Negev, Israel.

REFERENCES

Niki Aifanti, Christos Papachristou, and Anastasios Delopoulos. The MUG facial expression database. In *11th International Workshop on Image Analysis for Multimedia Interactive Services WIAMIS 10*, pp. 1–4, 2010.

Hassan Arbabi and Igor Mezic. Ergodic theory, dynamic mode decomposition, and computation of spectral properties of the Koopman operator. *SIAM Journal on Applied Dynamical Systems*, 16(4): 2096–2126, 2017.

Omri Azencot, Mirela Ben-Chen, Frédéric Chazal, and Maks Ovsjanikov. An operator approach to tangent vector field processing. In *Computer Graphics Forum*, volume 32, pp. 73–82. Wiley Online Library, 2013.

Omri Azencot, Steffen Weißmann, Maks Ovsjanikov, Max Wardetzky, and Mirela Ben-Chen. Functional fluids on surfaces. In *Computer Graphics Forum*, volume 33, pp. 237–246. Wiley Online Library, 2014.

Omri Azencot, Wotao Yin, and Andrea Bertozzi. Consistent dynamic mode decomposition. *SIAM Journal on Applied Dynamical Systems*, 18(3):1565–1585, 2019.

Omri Azencot, N Benjamin Erichson, Vanessa Lin, and Michael Mahoney. Forecasting sequential data using consistent Koopman autoencoders. In *International Conference on Machine Learning*, pp. 475–485. PMLR, 2020.

Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio. Neural machine translation by jointly learning to align and translate. *arXiv preprint arXiv:1409.0473*, 2014.

Junwen Bai, Weiran Wang, Yingbo Zhou, and Caiming Xiong. Representation learning for sequence data with deep autoencoding predictive components. *arXiv preprint arXiv:2010.03135*, 2020.

Junwen Bai, Weiran Wang, and Carla P Gomes. Contrastively disentangled sequential variational autoencoder. *Advances in Neural Information Processing Systems*, 34, 2021.

Yoshua Bengio, Aaron Courville, and Pascal Vincent. Representation learning: A review and new perspectives. *IEEE transactions on pattern analysis and machine intelligence*, 35(8):1798–1828, 2013.

Diane Bouchacourt, Ryota Tomioka, and Sebastian Nowozin. Multi-level variational autoencoder: Learning disentangled representations from grouped observations. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 32, 2018.

Steven L Brunton, Marko Budišić, Eurika Kaiser, and J Nathan Kutz. Modern Koopman theory for dynamical systems. *arXiv preprint arXiv:2102.12086*, 2021.

Marko Budišić, Ryan Mohr, and Igor Mezić. Applied Koopmanism. *Chaos: An Interdisciplinary Journal of Nonlinear Science*, 22(4):047510, 2012.

Christopher P Burgess, Irina Higgins, Arka Pal, Loic Matthey, Nick Watters, Guillaume Desjardins, and Alexander Lerchner. Understanding disentangling in beta-vae. *arXiv preprint arXiv:1804.03599*, 2018.

Ricky TQ Chen, Xuechen Li, Roger B Grosse, and David K Duvenaud. Isolating sources of disentanglement in variational autoencoders. *Advances in neural information processing systems*, 31, 2018.

Xi Chen, Yan Duan, Rein Houthooft, John Schulman, Ilya Sutskever, and Pieter Abbeel. Infogan: Interpretable representation learning by information maximizing generative adversarial nets. *Advances in neural information processing systems*, 29, 2016.

Mohamed Chenafa, Dan Istrate, Valeriu Vrabie, and Michel Herbin. Biometric system based on voice recognition using multiclassifiers. In *European Workshop on Biometrics and Identity Management*, pp. 206–215. Springer, 2008.

David Clark, Jesse Livezey, and Kristofer Bouchard. Unsupervised discovery of temporal structure in noisy data with dynamical components analysis. *Advances in Neural Information Processing Systems*, 32, 2019.

Ido Cohen, Omri Azencot, Pavel Lifshits, and Guy Gilboa. Modes of homogeneous gradient flows. *SIAM Journal on Imaging Sciences*, 14(3):913–945, 2021.

Armand Comas, Sandesh Ghimire, Haolin Li, Mario Sznaier, and Octavia Camps. Self-supervised decomposition, disentanglement and prediction of video sequences while interpreting dynamics: A Koopman perspective. *arXiv preprint arXiv:2110.00547*, 2021.

Suddhasattwa Das and Dimitrios Giannakis. Delay-coordinate maps and the spectra of Koopman operators. *Journal of Statistical Physics*, 175(6):1107–1145, 2019.

Emily L Denton and Vighnesh Birodkar. Unsupervised learning of disentangled representations from video. *Advances in neural information processing systems*, 30, 2017.

Tanja Eisner, Bálint Farkas, Markus Haase, and Rainer Nagel. *Operator theoretic aspects of ergodic theory*, volume 272. Springer, 2015.

Marco Fraccaro, Simon Kamronn, Ulrich Paquet, and Ole Winther. A disentangled recognition and nonlinear dynamics model for unsupervised learning. *Advances in neural information processing systems*, 30, 2017.

J. Garofolo, Lori Lamel, W. Fisher, Jonathan Fiscus, D. Pallett, N. Dahlgren, and V. Zue. TIMIT acoustic-phonetic continuous speech corpus. *Linguistic Data Consortium*, 11 1992.

Laurent Girin, Simon Leglaive, Xiaoyu Bie, Julien Diard, Thomas Hueber, and Xavier Alameda-Pineda. Dynamical Variational Autoencoders: A comprehensive review. *arXiv preprint arXiv:2008.12595*, 2020.

Ian Goodfellow, Yoshua Bengio, and Aaron Courville. *Deep learning*. MIT press, 2016.

Jun Han, Martin Renqiang Min, Ligong Han, Li Erran Li, and Xuan Zhang. Disentangled Recurrent Wasserstein Autoencoder. *arXiv preprint arXiv:2101.07496*, 2021a.

Minghao Han, Jacob Euler-Rolle, and Robert K Katzschmann. DeSKO: Stability-assured robust control with a deep stochastic Koopman operator. In *International Conference on Learning Representations*, 2021b.

Irina Higgins, Loic Matthey, Arka Pal, Christopher Burgess, Xavier Glorot, Matthew Botvinick, Shakir Mohamed, and Alexander Lerchner. beta-vae: Learning basic visual concepts with a constrained variational framework. In *International Conference on Learning Representations, ICLR*, 2017.

Geoffrey E Hinton and Richard Zemel. Autoencoders, minimum description length and Helmholtz free energy. *Advances in neural information processing systems*, 6, 1993.

Wei-Ning Hsu, Yu Zhang, and James Glass. Unsupervised learning of disentangled and interpretable representations from sequential data. *Advances in neural information processing systems*, 30, 2017.

Catalin Ionescu, Orestis Vantzos, and Cristian Sminchisescu. Matrix backpropagation for deep networks with structured layers. In *Proceedings of the IEEE international conference on computer vision*, pp. 2965–2973, 2015.

Tomoharu Iwata and Yoshinobu Kawahara. Neural dynamic mode decomposition for end-to-end modeling of nonlinear dynamics. *arXiv preprint arXiv:2012.06191*, 2020.

Hyunjik Kim and Andriy Mnih. Disentangling by factorising. In *International Conference on Machine Learning*, pp. 2649–2658. PMLR, 2018.

Diederik P Kingma and Jimmy Ba. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2014.

Bernard O Koopman. Hamiltonian systems and transformation in Hilbert space. *Proceedings of the national academy of sciences of the united states of america*, 17(5):315, 1931.

Tejas D Kulkarni, William F Whitney, Pushmeet Kohli, and Josh Tenenbaum. Deep convolutional inverse graphics network. *Advances in neural information processing systems*, 28, 2015.

Abhishek Kumar, Prasanna Sattigeri, and Avinash Balakrishnan. Variational inference of disentangled latent concepts from unlabeled observations. *arXiv preprint arXiv:1711.00848*, 2017.

J Nathan Kutz, Steven L Brunton, Bingni W Brunton, and Joshua L Proctor. *Dynamic mode decomposition: data-driven modeling of complex systems*. SIAM, 2016.

Yueheng Lan and Igor Mezić. Linearization in the large of nonlinear systems and Koopman operator spectrum. *Physica D: Nonlinear Phenomena*, 242(1):42–53, 2013.

Yingzhen Li and Stephan Mandt. Disentangled sequential autoencoder. *arXiv preprint arXiv:1803.02991*, 2018.

Yuheng Li, Krishna Kumar Singh, Utkarsh Ojha, and Yong Jae Lee. MixNMatch: Multifactor disentanglement and encoding for conditional image generation. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 8039–8048, 2020.

Yunzhu Li, Hao He, Jiajun Wu, Dina Katabi, and Antonio Torralba. Learning compositional Koopman operators for model-based control. *arXiv preprint arXiv:1910.08264*, 2019.

Francesco Locatello, Stefan Bauer, Mario Lucic, Gunnar Raetsch, Sylvain Gelly, Bernhard Schölkopf, and Olivier Bachem. Challenging common assumptions in the unsupervised learning of disentangled representations. In *international conference on machine learning*, pp. 4114–4124. PMLR, 2019.

Bethany Lusch, J Nathan Kutz, and Steven L Brunton. Deep learning for universal linear embeddings of nonlinear dynamics. *Nature communications*, 9(1):1–10, 2018.

Alexandre Mauroy and Igor Mezić. A spectral operator-theoretic framework for global stability. In *52nd IEEE Conference on Decision and Control*, pp. 5234–5239. IEEE, 2013.

Alexandre Mauroy and Igor Mezić. Global stability analysis using the eigenfunctions of the Koopman operator. *IEEE Transactions on Automatic Control*, 61(11):3356–3369, 2016.

Igor Mezić. Spectral properties of dynamical systems, model reduction and decompositions. *Nonlinear Dynamics*, 41(1):309–325, 2005.

Igor Mezić. Analysis of fluid flows via spectral properties of the Koopman operator. *Annual Review of Fluid Mechanics*, 45:357–378, 2013.

Igor Mezic. Koopman operator spectrum and data analysis. *arXiv preprint arXiv:1702.07597*, 2017.

Đorđe Miladinović, Muhammad Waleed Gondal, Bernhard Schölkopf, Joachim M Buhmann, and Stefan Bauer. Disentangled state space representations. *arXiv preprint arXiv:1906.03255*, 2019.

Ryan Mohr and Igor Mezić. Construction of eigenfunctions for scalar-type operators via Laplace averages with connections to the Koopman operator. *arXiv preprint arXiv:1403.6559*, 2014.

Jeremy Morton, Antony Jameson, Mykel J Kochenderfer, and Freddie Witherden. Deep dynamical modeling and control of unsteady fluid flows. *Advances in Neural Information Processing Systems*, 31, 2018.

Ilan Naiman and Omri Azencot. An operator theoretic approach for analyzing sequence neural networks. In *Proceedings of the AAAI conference on artificial intelligence*, 2023.

Samuel E Otto and Clarence W Rowley. Linearly recurrent autoencoder networks for learning dynamics. *SIAM Journal on Applied Dynamical Systems*, 18(1):558–593, 2019.

Adam Paszke, Sam Gross, Francisco Massa, Adam Lerer, James Bradbury, Gregory Chanan, Trevor Killeen, Zeming Lin, Natalia Gimelshein, Luca Antiga, Alban Desmaison, Andreas Kopf, Edward Yang, Zachary DeVito, Martin Raison, Alykhan Tejani, Sasank Chilamkurthy, Benoit Steiner, Lu Fang, Junjie Bai, and Soumith Chintala. PyTorch: An imperative style, high-performance deep learning library. In H. Wallach, H. Larochelle, A. Beygelzimer, F. d'Alché-Buc, E. Fox, and R. Garnett (eds.), *Advances in Neural Information Processing Systems 32*, pp. 8024–8035. Curran Associates, Inc., 2019.

William T Redman, Maria Fonoberova, Ryan Mohr, Yannis Kevrekidis, and Igor Mezic. An operator theoretic view on pruning deep neural networks. In *International Conference on Learning Representations*, 2021.

Scott E Reed, Yi Zhang, Yuting Zhang, and Honglak Lee. Deep visual analogy-making. *Advances in neural information processing systems*, 28, 2015.

Clarence W Rowley, Igor Mezić, Shervin Bagheri, Philipp Schlatter, and Dan S Henningson. Spectral analysis of nonlinear flows. *Journal of fluid mechanics*, 641:115–127, 2009.

Raif M Rustamov, Maks Ovsjanikov, Omri Azencot, Mirela Ben-Chen, Frédéric Chazal, and Leonidas Guibas. Map-based exploration of intrinsic shape differences and variability. *ACM Transactions on Graphics (TOG)*, 32(4):1–12, 2013.

Peter J Schmid. Dynamic mode decomposition of numerical and experimental data. *Journal of fluid mechanics*, 656:5–28, 2010.

Naoya Takeishi, Yoshinobu Kawahara, and Takehisa Yairi. Learning Koopman invariant subspaces for dynamic mode decomposition. *Advances in Neural Information Processing Systems*, 30, 2017.

Sergey Tulyakov, Ming-Yu Liu, Xiaodong Yang, and Jan Kautz. Mocogan: Decomposing motion and content for video generation. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 1526–1535, 2018.

Laurens Van der Maaten and Geoffrey Hinton. Visualizing data using t-SNE. *Journal of machine learning research*, 9(11), 2008.

Ruben Villegas, Jimei Yang, Seunghoon Hong, Xunyu Lin, and Honglak Lee. Decomposing motion and content for natural video sequence prediction. *arXiv preprint arXiv:1706.08033*, 2017.

Stephen Wiggins, Stephen Wiggins, and Martin Golubitsky. *Introduction to applied nonlinear dynamical systems and chaos*, volume 2. Springer, 2003.

Sitao Xiang, Yuming Gu, Pengda Xiang, Menglei Chai, Hao Li, Yajie Zhao, and Mingming He. DisUnknown: Distilling unknown factors for disentanglement learning. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pp. 14810–14819, 2021.

Masanori Yamada, Heecheol Kim, Kosuke Miyoshi, Tomoharu Iwata, and Hiroshi Yamakawa. Disentangled representations for sequence data using information bottleneck principle. In *Asian Conference on Machine Learning*, pp. 305–320. PMLR, 2020.

Enoch Yeung, Soumya Kundu, and Nathan Hodas. Learning deep neural network representations for Koopman operators of nonlinear dynamical systems. In *2019 American Control Conference (ACC)*, pp. 4832–4839. IEEE, 2019.

Shengjia Zhao, Jiaming Song, and Stefano Ermon. Learning hierarchical features from generative models. *arXiv preprint arXiv:1702.08396*, 2017.

Yizhe Zhu, Martin Renqiang Min, Asim Kadav, and Hans Peter Graf. S3VAE: Self-supervised sequential VAE for representation disentanglement and data generation. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 6538–6547, 2020.

A KOOPMAN THEORY

We briefly introduce the key ingredients of Koopman theory (Koopman, 1931) which are related to our work. Consider a dynamical system $\varphi : \mathcal{M} \to \mathcal{M}$ over the domain $\mathcal{M}$ given via the update rule

$$x_{t+1} = \varphi(x_t) \ ,$$

where $x_t \in \mathcal{M} \subset \mathbb{R}^m$, and $t \in \mathbb{N}$ is the time index. Koopman theory proposes an alternative representation of the dynamical system φ by a *linear* yet infinite-dimensional Koopman operator $\mathcal{K}_\varphi$. Formally,

$$\mathcal{K}_\varphi f(x_t) = f \circ \varphi(x_t) \ ,$$

where $f : \mathcal{M} \to \mathbb{C}$ is an observable complex-valued function, and $f \circ \varphi$ denotes composition of transformations. Due to the linearity of $\mathcal{K}_\varphi$, we can discuss its eigendecomposition, when it exists. Specifically, let $\lambda_j \in \mathbb{C}, \phi_j : \mathcal{M} \to \mathbb{C}$ be a pair of eigenvalue and eigenfunction respectively of $\mathcal{K}_\varphi$, i.e., it holds that

$$\mathcal{K}_\varphi \phi_j = \lambda_j \phi_j \quad \text{for any } j \ .$$

From a theoretical viewpoint, there is no loss of information to represent the dynamics with φ or with $\mathcal{K}_\varphi$ (Eisner et al., 2015). Namely, one can recover the dynamics φ from a given $\mathcal{K}_\varphi$ operator. Moreover, the Hartman-Grobman Theorem states that the linearization around hyperbolic fixed points is conjugate to the full, nonlinear system (Wiggins et al., 2003). The latter result was further extended to the entirety of the basin (Lan & Mezić, 2013). In practice, various tools were recently developed to approximate the infinite-dimensional Koopman operator using a finite-dimensional Koopman matrix. In particular, the Dynamic Mode Decomposition (DMD) (Schmid, 2010) is a popular technique for approximating dynamical systems and their modes. DMD was shown to be intimately related to Koopman mode decomposition in (Rowley et al., 2009), which deals with the extraction of Koopman eigenvalues and eigenfunctions in a data-driven setting. Thus, the above discussion establishes the link between our work and Koopman theory since in practice, our Koopman module is similar in spirit to DMD. Moreover, it justifies our use of the Koopman matrix to encode the dynamics as well as disentangle it.

B EXPERIMENTAL SETUP: ARCHITECTURE, DATASETS, HYPERPARAMETERS, AND MORE

B.1 DATASETS

Sprites. Reed et al. (2015) introduced a dataset of animated cartoon characters. Each character is composed of static and dynamic attributes. The static attributes include the color of skin, tops, pants and hair; each contains six possible variants. The dynamic attributes include three different motions: walking, casting spells and slashing, where each motion admits three different orientations: left, right, and forward. In total there are nine motions a character can perform and 1296 unique characters. A sequence is composed of eight RGB image frames of size of 64×64. We use 9000 samples for training and 2664 samples for testing.

MUG. Aifanti et al. (2010) share a facial expression dataset which contains image sequences of 52 subjects. Each subject performs six facial expressions: anger, fear, disgust, happiness, sadness and surprise. Each video in the dataset consists of 50 to 160 frames. To create sequences of length 15 as described in previous work (Bai et al., 2021), we randomly sample 15 frames from the original sequence. Then, we crop the faces using Haar Cascades face detection, and we resize to 64×64 resulting in sequences $x \in \mathbb{R}^{15\times3\times64\times64}$ for a total of 3429 samples. Finally, we split the dataset such that 75% of it is used for the train set, and 25% for the test set.

TIMIT. Garofolo et al. (1992) made TIMIT available which contains 16kHz audio recordings of American English speakers reading short texts. In total, the dataset has 6300 utterances (5.4 hours) aggregated from 630 speakers reading 10 phonetically rich sentences each. For each batch of samples the data pre-processing procedure goes as follows: First, we take the maximum raw audio length in the batch, and we zero pad all samples to match that length. Second, we calculate for each sample its log spectrogram with 201 frequency features calculated by a window of 10ms, using Short Time Fourier Transform (STFT). Thus, each batch has its own t (time steps) length, with an average length after padding of $t = 450$. The resulting sequences are of dimension $x \in \mathbb{R}^{t\times201}$.

B.2 DISENTANGLEMENT METRICS

Accuracy (Acc) measures how well a model preserves the fixed features while sampling the others. We compute it using a pre-trained classifier $\mathfrak{C}$ (also called judge) which is trained on the same train set and tested on the same test set as our model. The classifier outputs the probability measures per feature of the dataset. For instance, $\mathfrak{C}$ outputs one label for the pose and additional labels for each of the static factors (hair, skin, top and pants) for the Sprites dataset.

Inception Score (IS) measures the performance of a generator. The score is calculated by first applying the judge $\mathfrak{C}$ on every generated sequence $x_{1:t}$, yielding the conditional predicted label distribution $p(y|x_{1:t})$. Then, given the marginal predicted label distribution $p(y)$ we compute the Kullback—Leibler (KL) divergence $\mathrm{KL}\left(p(y|x_{1:t}) \,||\, p(y)\right)$. The inception score is given by:

$$\mathrm{IS} = \exp(\mathbb{E}_x \left[\mathrm{KL}(p(y|x_{1:T})) \,||\, p(y)\right]) \,.$$

Intra-Entropy $H(y|x)$ measures the conditional predicted label entropy of all generated sequences. To obtain the predicted labels we use the judge $\mathfrak{C}$, and we compute $\frac{1}{b}\sum_{i=1}^{b} H(p(y|x^i_{1:t}))$ where b is the number of generated sequences. Lower intra-entropy score reflects higher confidence of the classifier $\mathfrak{C}$.

Inter-Entropy $H(y)$ measures the marginal predicted label entropy of all generated sequences. We can compute $H(p(y))$ using the judge's output on the predicted labels $\{y\}$. Higher inter-entropy score reflects higher diversity among the generated sequences.

Equal Error Rate (EER) is used in the speaker verification task on the TIMIT dataset. It is the value of false positive rate or false negative rate of a model over the speaker verification task, when the rates are equal.

B.3 ARCHITECTURE AND HYPERPARAMETERS

Our models are implemented in the PyTorch (Paszke et al., 2019) framework. We used Adam optimizer (Kingma & Ba, 2014) and a learning rate of 0.001 for all models, with no weight decay. Regarding hyper-parameters, in our experiments, k is tuned between 40 and 200 and $\lambda_{\mathrm{rec}}, \lambda_{\mathrm{pred}}$ and λ_{eig} are tuned over $\{1, 3, 5, 10, 15, 20\}$. k_s is tuned between 4 and 20, and the ε threshold for the dynamic loss is tuned over $\{0.4, 0.5, 0.55, 0.6, 0.65\}$. The hyper-parameters are chosen through standard grid search.

B.3.1 ENCODER AND DECODER

Sprites and MUG. Our encoder and decoder follow the same general structure as in Bai et al. (2021). First we have the same convolutional encoder as in C-DSVAE. Then we have a uni-directional LSTM. The architecture is described in detail in Tab. 5, where Conv2D and Conv2DT denote a 2D convolution layer and its transpose, and BN2D is a 2D batch normalization layer. Additionally, the hyperparameters are listed in Tab. 6, where b is the batch size, k is the size of Koopman matrix, h is the dimension of the LSTM hidden state, and #epochs is the number of epochs we used for training. The balance weights $\lambda_{\mathrm{rec}}, \lambda_{\mathrm{pred}}$ and λ_{eig} scale the loss penalty terms of the Koopman layer, $\mathcal{L}_{\mathrm{rec}}, \mathcal{L}_{\mathrm{pred}}$ and $\mathcal{L}_{\mathrm{eig}}$, respectively. Finally, k_s is the amount of static factors, and ϵ is the dynamic threshold, see Eqs. 6 and 7 in the main text.

Table 5: Architecture details.

Encoder	Decoder
$64 \times 64 \times 3$ image	Z
Conv2D$(3, 32, 4, 2, 1)$ $\rightarrow$ BN2D(32) $\rightarrow$ LeakyReLU	LSTM(k, h)
Conv2D$(32, 64, 4, 2, 1)$ $\rightarrow$ BN2D(64) $\rightarrow$ LeakyReLU	Conv2DT$(h, 256, 4, 1, 0)$ $\rightarrow$ BN2D(256) $\rightarrow$ LeakyReLU
Conv2D$(64, 128, 4, 2, 1)$ $\rightarrow$ BN2D(128) $\rightarrow$ LeakyReLU	Conv2DT$(256, 128, 4, 1, 0)$ $\rightarrow$ BN2D(128) $\rightarrow$ LeakyReLU
Conv2D$(128, 256, 4, 2, 1)$ $\rightarrow$ BN2D(256) $\rightarrow$ LeakyReLU	Conv2DT$(128, 64, 4, 1, 0)$ $\rightarrow$ BN2D(64) $\rightarrow$ LeakyReLU
Conv2D$(256, k, 4, 2, 1)$ $\rightarrow$ BN2D(k) $\rightarrow$ LeakyReLU	Conv2DT$(64, 32, 4, 1, 0)$ $\rightarrow$ BN2D(32) $\rightarrow$ LeakyReLU
LSTM(k, k)	Conv2DT$(32, 3, 4, 1, 0)$ $\rightarrow$ Sigmoid

Table 6: Hyperparameter details.

Dataset	b	k	h	#epochs	λ_{rec}	λ_{pred}	λ_{eig}	k_s	ϵ
Sprites	32	40	40	800	15	1	1	8	0.5
MUG	16	40	100	1000	20	1	1	5	0.5
TIMIT	30	165	-	400	15	3	1	15	0

TIMIT. We design a neural network related to DSVAE architecture, but we use a uni-directional LSTM module instead of a bi-directional layer. The encoder LSTM input dimension is 201 which is the spectrogram features dimension and its output dimension is k. The decoder LSTM input dimension is k and its output dimension is 201. The hyperparameter values are detailed in Tab. 6.

B.3.2 KOOPMAN LAYER

The Koopman layer in our architecture is responsible for calculating the Koopman matrix C, and it is associated with the accompanying losses $\mathcal{L}_{\text{rec}}, \mathcal{L}_{\text{pred}}, \mathcal{L}_{\text{eig}}$. It may happen that the latent codes provided to the Koopman module are very similar, leading to numerically unstable computations. To alleviate this issue, we consider two possibilities. One, use blur filter on the image before inserting it to the encoder (used for the Sprites datasets). Two, add small random uniform noise sampled from $[0, 1]$ to the latent code Z, i.e., $Z + 0.005N$, where N denotes the noise (used on TIMIT). Both options yield more diverse latent encodings, which in turn, stabilize the computation of C and the training procedure. Finally, we note that our spectral penalty terms $\mathcal{L}_{\text{stat}}$ and $\mathcal{L}_{\text{dyn}}$ which compose $\mathcal{L}_{\text{eig}}$ are stable for a large regime of hyperparameter ranges.

B.3.3 ADDITIONAL DYNAMIC LOSS OPTIONS

The proposed form of $\mathcal{L}_{\text{dyn}}$ in Eq. 7 constrains the dynamic factor modulus to an ϵ-ball to promote separation between the static factors located on the point $1 + \imath 0$ and the dynamic factors. However, there are settings for which $\mathcal{L}_{\text{dyn}}$ may be not optimal. For instance, a dataset may contain measure-preserving dynamic factors, e.g., as in the motion of a pendulum. Another example includes growing dynamic factors, e.g., as in a ball moving from the center of the frame towards the boundaries of the frame. If one has additional knowledge regarding the underlying dynamics, one may adapt $\mathcal{L}_{\text{dyn}}$ accordingly. We consider the following options:

1. Set $\epsilon = 1$ while adding the dynamic loss term to $\mathcal{L}_{\text{eig}}$. In this case, $\mathcal{L}_{\text{dyn}}$ penalizes dynamic factors that are inside a δ-ball around the point $1 + 0\imath$. This option addresses measure-preserving dynamic oscillations in the data.
2. Set $\epsilon = 1 + \eta,\ \eta > 0$ while adding the dynamic loss term to $\mathcal{L}_{\text{eig}}$. In this case, $\mathcal{L}_{\text{dyn}}$ penalizes dynamic factors that are inside a δ-ball around the point $1 + 0\imath$. This option addresses growing dynamic factors.

B.4 DISENTANGLEMENT PROCESS USING MULTIFACTOR DISENTANGLING KOOPMAN AUTOENCODERS

In what follows, we detail the process of extracting the multifactor latent representation of a sample, and in addition, we will demonstrate a general swap of a factor between two arbitrary samples. We let $X \in \mathbb{R}^{b \times t \times m}$ be our input batch and $x \in \mathbb{R}^m$ be a single sample that we want to calculate its multifactor disentangled latent representation. The disentanglement process of x into its multiple latent factors representations using our model contains the following steps:

1. We insert X into the model encoder and get the encoder output $Z \in \mathbb{R}^{b \times t \times k}$.
2. We compute the Koopman matrix C for the batch X using the Koopman layer as described in the main text.
3. We compute the eigendecomposition of C to get the eigenvectors matrix $V \in \mathbb{C}^{k \times k}$. In addition, we calculate $U = V^{-1}$. Now we calculate $\bar{z}^T := z^T V$ for every $z \in \mathbb{R}^k$. $\bar{z}$ stores the coefficients in the Koopman space and they are the disentangled latent representation in our method. Notice that $z^T = z^T V U = \bar{z}^T U$

4. We identify the indices that correspond to each latent factor. It may be that several indices represent one factor. We use the identification method of subspaces described in B.5 to extract the indices set. Let I_k be some latent factor index set. Then, the latent representation of factor I_k for the input x is $\bar{z}[I_k]$. For instance, I_k can be the hair color factor. If we want to take a group of factors, we can aggregate a few factors together $I = I_s \cup I_t \cup I_h \cup I_p$, where $I_s =$ skin indices, $I_t =$ top indices, $I_h =$ hair indices, $I_p =$ pants indices. In practice I encodes a character identity on the Sprites dataset.

To conclude, these four steps describe the process of disentangling arbitrary factors in our setup. To demonstrate a swap, let us assume we use the Sprites dataset. Let x_1, x_2 be two samples in X and let us assume we want to swap their hair and skin attributes. We will use steps 1, 2 and 3 to extract x_1, x_2 multifactor latent representation $\bar{z}_1, \bar{z}_2$. Then, using Step 4, we will identify and extract $I_K = I_s \cup I_h$, were $I_h =$ hair indices and $I_s =$ skin indices. Now, we want to swap the latent representations of the hair and skin factors between the sample. To do so, we simply preform $\bar{z}_1[I_K] = \bar{z}_2[I_K]$ and vice versa $\bar{z}_2[I_K] = \bar{z}_1[I_K]$ in parallel.

To get back to the pixel space, we need to repeat our steps backward. First we need to compute the new z_i after the swap. We will do it using the V matrix we calculated in step 3. We compute $\tilde{z}_1^T = \bar{z}_1^T V, \tilde{z}_2^T = \bar{z}_2^T V$. Finally, we can insert $\text{Re}(\tilde{z}_1), \text{Re}(\tilde{z}_2)$ as inputs to the model decoder and get the desired swapped new samples $\tilde{x}_1, \tilde{x}_2$. Last note, if z is some input for the model decoder then z must be real-valued, however, z is typically complex-valued since $V, U \in \mathbb{C}^{k \times k}$. Thus, we keep the real part of z, and we eliminate its imaginary component.

In what follows, we show that $\text{Re}(z^T VU) = z^T$, and thus feeding the real part to the decoder as mentioned above is well justified. Moreover, a similar proof holds for swapped latent vectors, i.e., $\text{Im}(\tilde{z}) = 0$. Finally, we validated that standard numerical packages such as Numpy and pyTorch satisfy this property up to machine precision.

Theorem 1. *If $C \in \mathbb{R}^{k \times k}$ is full rank, then Re$(z^T VU) = z^T$ for any $z \in \mathbb{R}^k$, where V is the matrix of eigenvectors of C, and $U = V^{-1}$.*

Proof. It follows that

$$z^T VU = \sum_{j=1}^{k} \langle z, v_j \rangle u_j \ ,$$

where v_j is the j-th column of V, and u_j is the j-row of U. To prove that $\text{Im}(z^T VU) = 0$, it is sufficient to show that if v_1 and v_2 are complex conjugate pair of vectors from V, i.e., $v_{i_1} = \overline{v_{i_2}}$, then $\langle z, v_1 \rangle u_1$ is the complex conjugate of $\langle z, v_2 \rangle u_2$. First, we have that

$$a_1 = \langle z, v_1 \rangle = \sum_{i}^{k} z[i] v_1[i] = \sum_{i}^{k} z[i] \overline{v_2[i]} = \langle z, \overline{v_2} \rangle = \overline{a_2} \ ,$$

where the third equality holds since $v_1 = \overline{v_2}$, and the last equality holds since z is real-valued. The proof is complete if we show that $u_1 = \overline{u_2}$, since then we have $\langle z, v_1 \rangle u_1 = \overline{\langle z, v_2 \rangle u_2}$. To verify that complex conjugate column pairs transform to complex conjugate row pairs, we assume w.l.o.g that the matrix V can be organized such that nearby columns are complex conjugates, i.e., $v_1 = \overline{v_2}, v_3 = \overline{v_4}$, and so on. Let P be the permutation matrix that exchanges the columns of V to their complex conjugates, i.e., it switches the i-th column with the $(i+1)$-th column, where i is odd. Then $VP = \overline{V}$. It follows that

$$(VP)^{-1} = P^T V^{-1} = P^T U = \overline{U} \ ,$$

namely, the i-th row is the complex conjugate of the $(i+1)$-th row, where i is an odd number. □

B.5 Identification of Subspaces

There are two scenarios in which we need to identify semantic Koopman subspaces in the eigenvectors of the Koopman matrix C:

1. separate between static and dynamic information (*two factor separation*).
2. identify individual factors, e.g., hair color in sprites (*multifactor separation*).

Table 7: Accuracy measures of factorial swap experiments, see Tab. 1.

Test	action	skin	top	pants	hair
hair swap	$11.35\% \pm 0.65\%$	$17.40\% \pm 0.79\%$	$17.07\% \pm 0.77\%$	$36.29\% \pm 0.88\%$	$\mathbf{90.20\% \pm 0.52\%}$
skin swap	$11.35\% \pm 0.65\%$	$\mathbf{72.72\% \pm 0.68\%}$	$17.23\% \pm 0.89\%$	$31.22\% \pm 0.84\%$	$16.92\% \pm 0.77\%$

Two factor separation. To distinguish between time-invariant and time-varying factors, we sort the eigenvalues based on their distance from the complex value $1 + \imath 0$. Then, the subspace of static features $I_{\texttt{stat}}$ is defined as the eigenvalues' indices of the first k_s elements in the sorted array. Then, the dynamic features subspace $I_{\texttt{dyn}}$ holds the remaining indices, i.e., $I_{\texttt{dyn}} = I \setminus I_{\texttt{stat}}$, where I is the set of all indices, and $S_1 \setminus S_2$ generates the set difference of the sets S_1 and S_2.

Multifactor separation. The identification of individual features such as the hair color or skin color in Sprites is less straightforward, unfortunately. Essentially, the key difficulty lies in that the Koopman matrix may encode an individual factor using a subspace whose dimension is unknown a priori. In addition, the subspace related to e.g., hair color may depend on the particular batch sample. For instance, we observed cases where the hair color subspace was of dimension $1, 2$ and 3 for three different batches. Nevertheless, manual inspection of $I_{\texttt{stat}}$ typically reveals the role of the eigenfunctions, and it can be achieved efficiently as $k_s \leq 15$ in our experiments. Still, we opt for an automatic approach, and thus we propose the following simple procedure. We consider the power set of $I_{\texttt{stat}}$, denoted by $\mathcal{I}_{\texttt{stat}}$. Let J be an element of $\mathcal{I}_{\texttt{stat}}$, then we swap the content of the batch with respect to J, and check the accuracy of the factor in question (e.g., hair color) using the pre-trained classifier. The subspace J which corresponds to a single factor change is the one for which the accuracy of the factor decreases the most with respect to the original samples. In practice, we noticed that often the subspace of a factor is composed of subsequent eigenvectors in the sorting described for the two factor separation. Thus, many subsets J of the power set $\mathcal{I}_{\texttt{stat}}$ can be ignored. We leave further exploration of this aspect for future work.

B.6 Speaker Verification Experiment Details

The speaker verification task in Sec. 5.3 is performed as follows. We use the test set of TIMIT which contains 24 unique speakers, with eight different sentences per speaker. In total there are 192 audio samples. We compute the latent representation Z for this data, and its Koopman matrix C. Using the eigendecomposition of C, we identify the static and dynamic subspaces $I_{\texttt{stat}}$ and $I_{\texttt{dyn}}$. We denote by $Z_{\texttt{stat}}, Z_{\texttt{dyn}}$ the latent codes obtained when projecting Z to $I_{\texttt{stat}}, I_{\texttt{dyn}}$, respectively. Formally, this is computed via $Z_{\texttt{stat}} = Z \cdot \Phi^{-1}[:,\ I_{\texttt{stat}}] \cdot \Phi[I_{\texttt{stat}}]$, and similarly for the dynamic features. To perform the speaker verification task we calculate the identity representation code for the batch given by

$$\hat{Z}_{\texttt{stat}} = \frac{1}{t}\sum_{j=1}^{t} Z_{\texttt{stat}}[:,\ j,\ :]\ , \quad \hat{Z}_{\texttt{dyn}} = \frac{1}{t}\sum_{j=1}^{t} Z_{\texttt{stat}}[:,\ j,\ :]\ , \quad \hat{Z}_{\texttt{dyn}}, \hat{Z}_{\texttt{dyn}} \in \mathbb{R}^{192\times 165}\ .$$

The EER calculations are performed separately for $\hat{Z}_{\texttt{stat}}$ and $\hat{Z}_{\texttt{dyn}}$ for all of their $192^2 = 18336$ pair combinations.

C Additional Results

C.1 Mean and Standard Deviation Measures

We report the mean and standard deviation measures computed over 300 runs for the results reported in Tab. 1, 2, 3 in the main text. The results are detailed in Tab. 7, 8. The low standard deviation highlights the robustness of our method to various seed numbers, and the overall stability of our trained models.

C.2 Data Generation

We present qualitative results of our model's unconditional generation capabilities. To this end, we randomly sample static and dynamic features by producing a new latent code based on a random

Table 8: Disentanglement metrics on Sprites and MUG, see Tabs. 2 3.

Method	Acc↑	IS↑	$H(y\|x)$↓	$H(y)$↑
Sprites	$\mathbf{100\% \pm 0\%}$	$\mathbf{8.999 \pm 2.3e{-6}}$	$\mathbf{1.6e{-7} \pm 2.2e{-7}}$	$\mathbf{2.197 \pm 0}$
MUG	$77.45\% \pm 0.62\%$	$\mathbf{5.569 \pm 0.026}$	$\mathbf{0.052 \pm 0.004}$	1.769 ± 0

sampling in the convex hull of two randomly chosen samples from the batch. That is, for every sample in the batch we generate random coefficients $\{\alpha_j \in [0, 1]\}$ which form a partition of unity $\sum_{j\in J} \alpha_j = 1$, where J denotes the sample indices, and $|J| = b = 2$ is the number of samples in the combination. Then, we swap the static or dynamic features of the source (src) sample using the convex combination, $\bar{Z}[\text{src}, :, I_{\texttt{stat}}] = \sum_{j\in J} \alpha_j \bar{Z}[j, :, I_{\texttt{stat}}]$, $\bar{Z}[\text{src}, :, I_{\texttt{dyn}}] = \sum_{j\in J} \alpha_j \bar{Z}[j, :, I_{\texttt{dyn}}]$, respectively. The reconstruction of the latent codes for which static or dynamic factors are swapped are shown on the right panels in Figs. 6, 7, 13, 14 respectively. Our results on both Sprites and MUG datasets demonstrate a non-trivial generation of factors while preserving the dynamic/static factors shown on the left panels.

C.3 Two Factors and Multifactor Swaps

We present several qualitative results of two factor swapping between static and dynamic factors of two given samples. In Figs. 8 and 9, each odd indexed row $i \in \{1, 3, 5, 7\}$ shows the source sequence on the left and the target sequence to the right. Even indexed rows $j \in \{2, 4, 6, 8\}$ represent the reconstructed samples after the swap where on the left we show the static swap, and on the right the dynamic swap. Notably, all examples show clean swaps while preserving non-swapped features.

Additionally, we extend the result in Fig. 2 to show an example in which we swap all multifactor combinations. Specifically, we show in Fig. 12 several multifactor swap from the source sequence (top row) to the target sequence (bottom row). The text to the left of every sequence in between denotes the swapped factor(s). For instance, the second row with the text `p` shows how the pants color of the target is swapped to the source character. Similarly, the row with the text `s+t+h` is related to the swap of the skin, top, and hair colors.

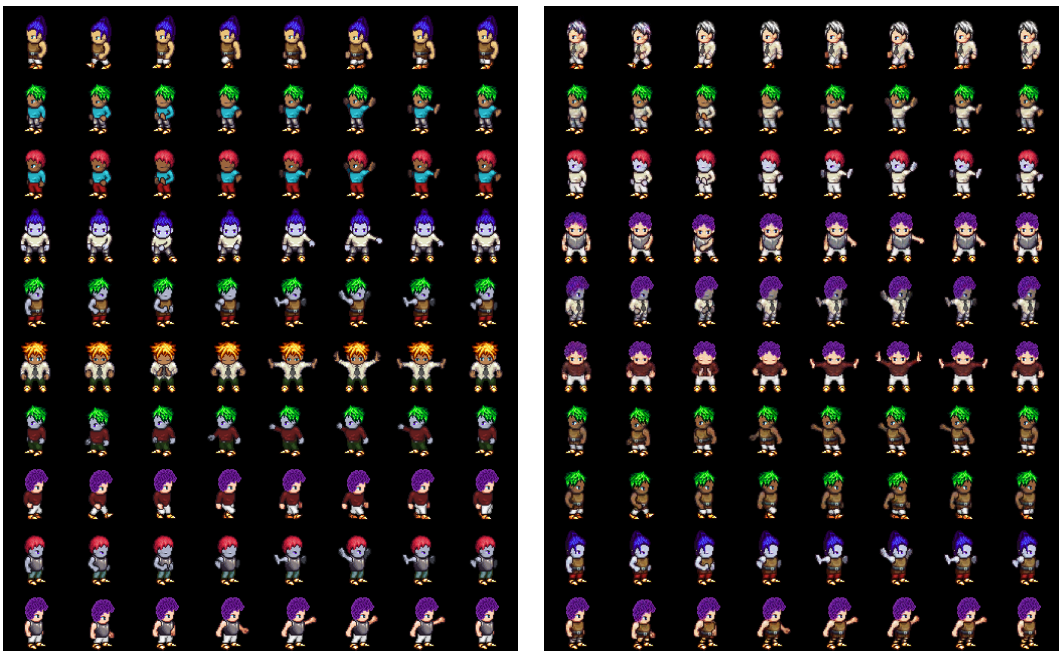


Figure 6: Unconditional generation of Sprite characters. The left panel shows the source sequences, and the right panel demonstrates the sampled characters where time-varying features are preserved.

Figure 7: Unconditional generation for the MUG dataset. The left panel shows the source sequences, and the right panel demonstrates the sampled identities where time-varying features are preserved.

C.4 STATIC INCREMENTAL SWAP ON MUG

Similarly to Fig. 4 in the main text, we now show an incremental swap example on the MUG dataset where the static features are swapped gradually, see Fig. 10. The multifactor subspaces used in this experiment are of sizes $|I_1| = 1, |I_2| = 2, |I_3| = 5$ where $I_1 \subset I_2 \subset I_3 \subset I_{\texttt{stat}}$. We observe a non-trivial gradual change from the source sequence (top row) to the target sequence (bottom row). In each incremental step, more static features are changing towards the target samples. Specifically, the skin color, hair color and density, ears structure, nose structure, chicks structure, chicks texture, lips and more other physical characteristics change gradually to better match the physical appearance of the target. Additionally, we observe that the source expression of the source is not altered during the transformation, highlighting the disentanglement capabilities of our approach.

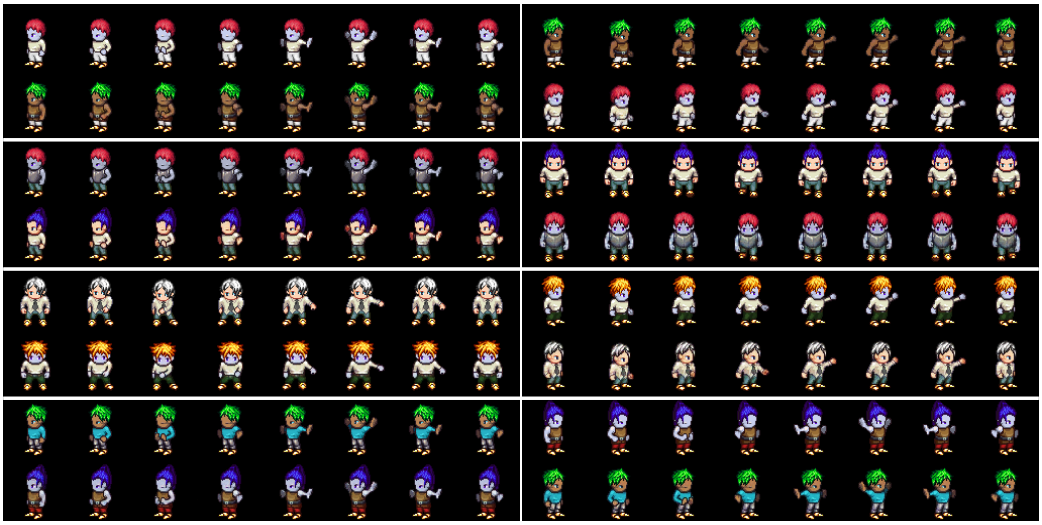


Figure 8: Several static and dynamic swap results on the Sprites dataset.

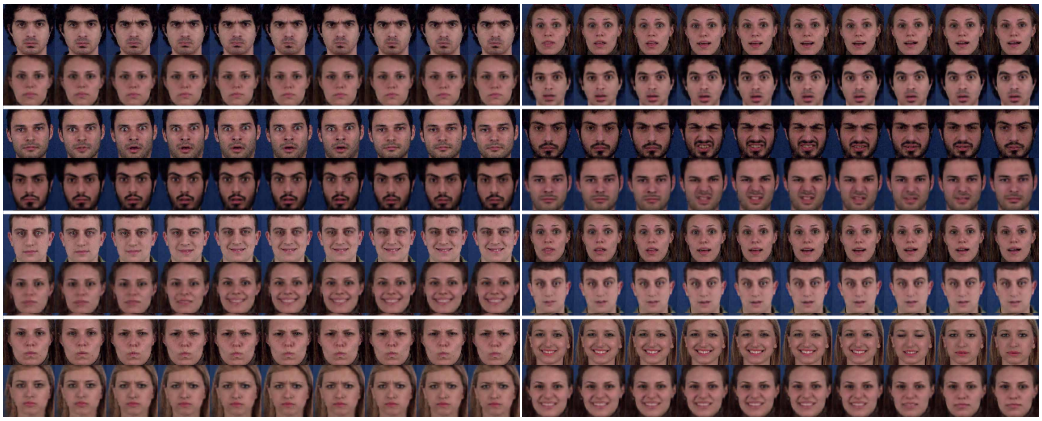


Figure 9: Several static and dynamic swap results on the MUG dataset.

C.5 KOOPMAN MATRIX SPECTRUM ABLATION STUDY

We would like to explore the impact of our spectral loss on the spectrum and the eigenvalues scattering of the Koopman matrix C. To this end, we train four different models: full model with $\mathcal{L}_{\text{eig}}$, KAE + $\mathcal{L}_{\text{stat}}$, KAE + $\mathcal{L}_{\text{dyn}}$, and baseline KAE without $\mathcal{L}_{\text{eig}}$. We show in Fig. 11 the obtained spectra for the various models, where eigenvalues associated with static factors are marked in blue, and the dynamic components are highlighted in red. Our model shows a clear separation between the static and dynamic factors, allowing to easily disentangle the data in practice. In contrast, the models KAE and $\mathcal{L}_{\texttt{stat}}$ yield spectra in which the static and dynamic components are very close to each other, leading to challenging disentanglement. Finally, the model $\mathcal{L}_{\text{dyn}}$ shows separation in its spectrum, however, some of the static factors drift away from the eigenvalue 1.

C.6 COMPUTATIONAL RESOURCES COMPARISON

We compare our method in terms of network memory footprint and the amount of data used for the Sprites dataset. We show in Tab. 9 the comparison of our method with respect to the other methods. All other approaches use significantly more parameters than our method, which uses 2 million weights. In addition, S3VAE and C-DSVAE utilize additional information during training. S3VAE exploits supervisory signals to an unknown extent as the details do not appear in the paper, and the code is proprietary. C-DSVAE uses data augmentation of size sixteen times the train set, that is, for content augmentation they generated eight times more train data, and the same amount for the motion augmentation. In comparison, our method and DSVAE do not use any additional data on top of the train set.

The time complexity analysis of our method is governed by the complexities of the encoder, decoder, the Koopman layer and the loss function. The encoder and decoder can be chosen freely and are

Figure 10: An incremental swap result of the static features on the MUG dataset.

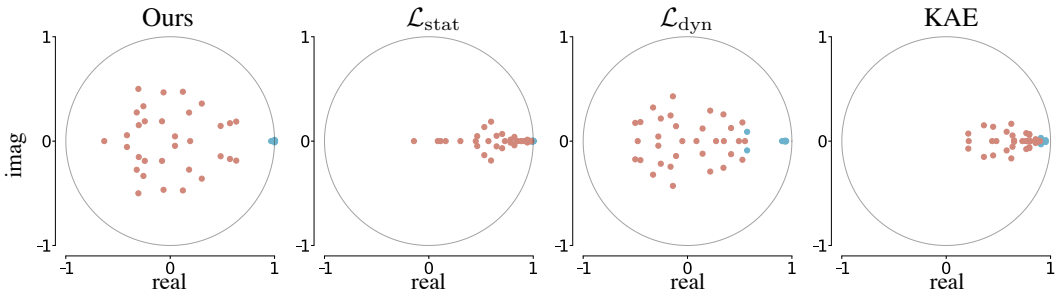


Figure 11: The Koopman matrix spectrum of different models.

typically similar to prior work (Hsu et al., 2017; Li & Mandt, 2018; Zhu et al., 2020; Bai et al., 2021), and thus we focus our analysis on the Koopman layer and the loss function. The dominant operation in the Koopman layer in terms of complexity is the computation of the pseudo-inverse of Z_p (please see Section 3). Computing the pseudo-inverse of a matrix is implemented in high-level deep learning frameworks such as pyTorch via SVD. The textbook complexity of SVD is $\mathcal{O}(\min(mn^2, m^2n))$ for an $m \times n$ matrix. In addition, computing the loss function involves eigendecomposition. The theoretic complexity of eigendecomposition is equivalent to that of matrix multiplication, which in our case is $\mathcal{O}(k^{2.376})$, where the Koopman operator is of size $k \times k$. In comparison, the matrices Z_p for which we compute pseudo-inverse are of size $b \cdot t \times k$, and typically $k < b \cdot t$. Thus, the pseudo-inverse operation governs the complexity of the algorithm. The development of efficient SVD algorithms for the GPU is an ongoing research topic in itself. As far as we know, there is some parallelization in torch SVD computation, mainly affecting the decomposition of *large* matrices. The Koopman matrices we use are typically small (e.g., 100×100), and thus the effective computation time is short.

Table 9: Computational resources comparison.

Method	DSVAE	R-WAE	S3VAE	C-DSVAE	Ours
Type	unsupervised	(weakly) unsupervised	self-supervised	self-supervised	unsupervised
Params	21M	121M	11M	11M	2M
Data	-	labels	supervisory signals	data augmentation ($\times$16)	-

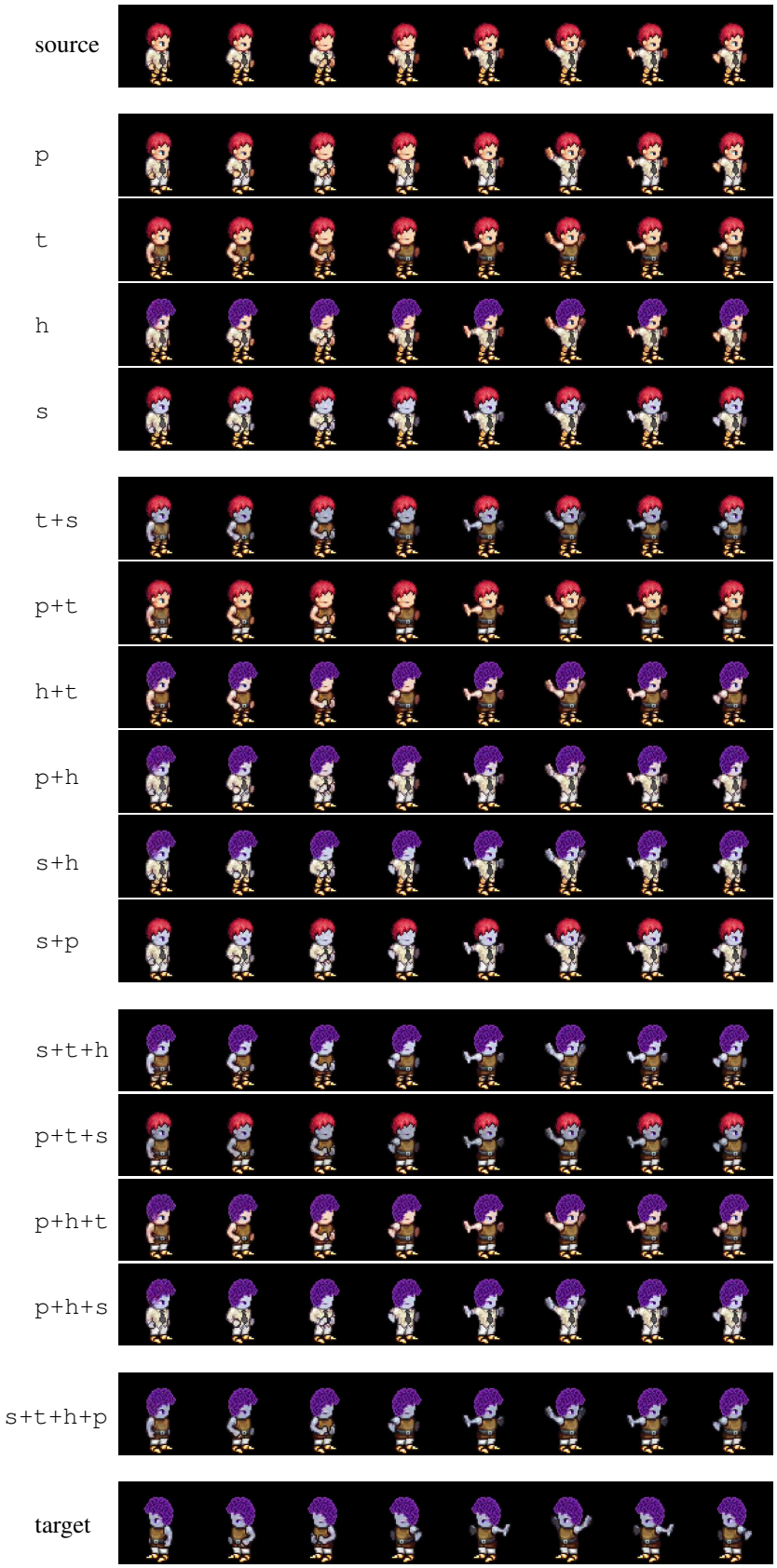


Figure 12: Multifactor swap of individual static factors and their combinations on the Sprites dataset.

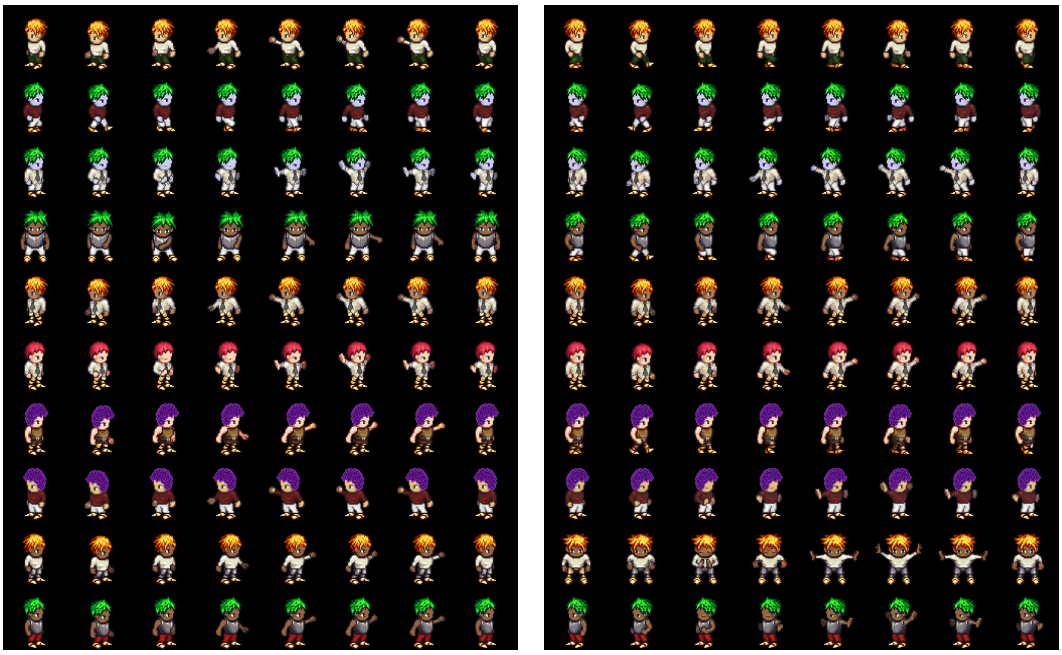


Figure 13: Unconditional generation of Sprite characters where the static factors are kept fixed, and the dynamic features are randomly sampled.

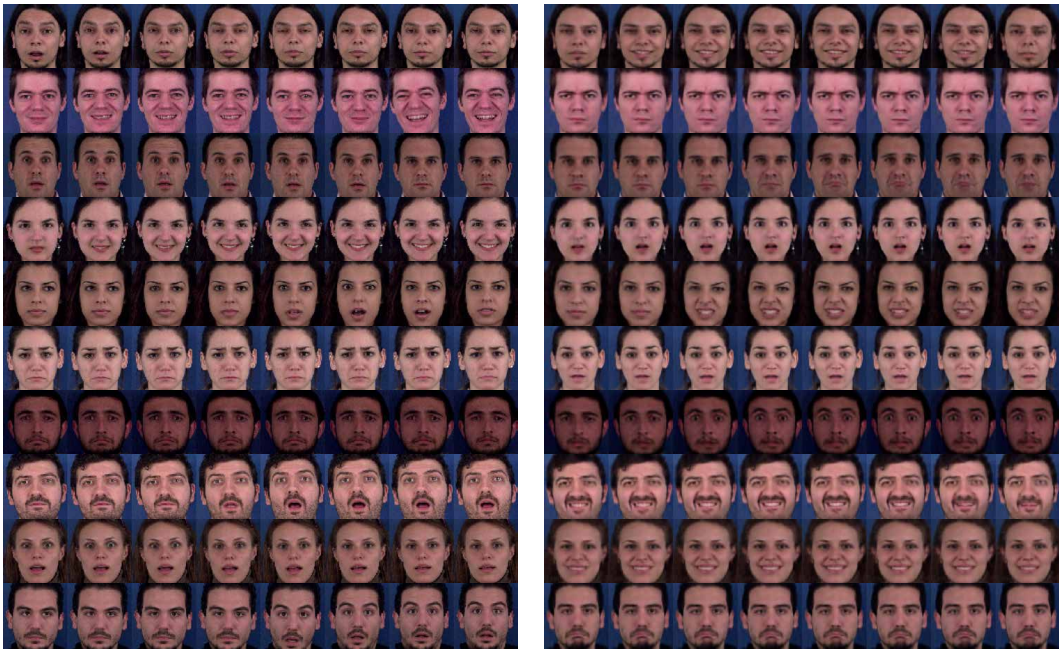


Figure 14: Unconditional generation of MUG images where the static factors are kept fixed, and the dynamic features are randomly sampled.

Mediated Protocols for Oblivious Transfer and Polynomial Evaluation

Aviad Ben Arie Tamir Tassa

Abstract

A secure multiparty computation (MPC) allows several parties to jointly compute a function over their inputs while keeping their inputs private. As a basic setting, the protocol involves only parties that hold inputs. In *mediated* (or *server-aided*) MPC, the protocol involves in addition to those parties external mediators/servers that perform the needed computation, without learning information on the inputs and outputs. In this study we propose mediated protocols for several fundamental MPC functionalities. Those protocols involve a set of mediators to whom the parties distribute secret shares in their private inputs. The mediators then proceed to compute the needed functionality on the received secret shares, while remaining completely oblivious to all underlying values. At the end, the mediators send secret shares to the relevant parties who can then reconstruct the sought-after outputs. We begin with a protocol called DSP (Distributed Scalar Product) for computing scalar products of private vectors. We then build upon DSP in designing various protocols for Oblivious Transfer (OT): k-out-of-n OT, Priced OT, and Generalized OT. We also use DSP in designing protocols for Oblivious Polynomial Evaluation (OPE) and Oblivious Multivariate Polynomial Evaluation (OMPE). In all of these problems there are two parties, Alice and Bob, that hold private vectors and they wish to compute their scalar product. However, in each of these problems Bob is restricted to submit a vector of a specified form. Hence, a crucial ingredient in our protocols is a sub-protocol in which the mediators validate that Bob's vector complies with the relevant restrictions, without learning anything else on that vector. As our protocols are based on Shamir's secret sharing scheme, they offer information-theoretic security, under the assumption of honest majority among the mediators, and they are very efficient.

1 Introduction

Distributed computing deals with settings in which a computation is carried out by several parties over input data that is distributed among them. One of the main challenges in distributed computing is the privacy of the interacting parties who, in some application scenarios, may wish to keep their own part of the input a secret. *Secure multi-party computation* (MPC) [23] is a central field of study in cryptography that aims at providing privacy-preserving solutions to distributed computing. In the basic setting of MPC, there are n mutually distrustful parties, $P_1, \ldots, P_n$, that hold private inputs, $x_1, \ldots, x_n$, and they wish to compute some joint function on their inputs, $f(x_1, \ldots, x_n)$. (The function can be sometimes multi-valued and issue different outputs to different designated parties.) Ideally, during the computation process, no party should gain any information on other parties' inputs, beyond what can be inferred from their own input and the output. Such perfect privacy is achievable for any function f that can be realized by a Boolean or an arithmetic circuit, by invoking generic solutions such as Yao's garbled circuit construction [23]. Such generic solutions are practical only for rather simple functions. When dealing with more involved functions, generic solutions are usually impractical, and more specialized solutions, which are tailored to the function of interest, should be developed.

Typically in distributed computing, and in particular in MPC, the only parties that are involved in the protocol are the functionality-relevant parties, namely, the parties that hold the inputs or those who need to receive the outputs. However, some studies considered a model of computation that is called *the mediated model* [1, 2, 8, 9, 10, 18, 21], or *the client-server model* [4, 6, 12, 16]. In that model there exist, apart from the functionality-relevant parties, also external *mediators*, $M_1, \ldots, M_D$, $D \geq 1$, to whom the parties outsource some of the needed computations. The mediators perform the computations while remaining oblivious to the private inputs and outputs. It turns out that the mediated model offers significant advantages: it may facilitate achieving the needed privacy goals; it does not require the parties to communicate with each other (a critical advantage in cases where the parties cannot efficiently communicate among themselves); in some settings it reduces communication costs; it allows the parties, that may run on computationally-bounded devices, to outsource costly computations to dedicated servers; and in some application scenarios it enables an economically-realistic collaboration model between them [18].

In this work we focus on basic MPC problems that involve two ($n = 2$) parties, Alice and Bob, and propose mediated protocols for their solution. In each of the studied problems, Alice's and Bob's private inputs may be encoded as vectors in a vector space over a finite field $\mathbb{Z}_p$; specifically, $\mathbf{a} = (a_1, \ldots, a_N) \in \mathbb{Z}_p^N$ is Alice's private vector and $\mathbf{b} = (b_1, \ldots, b_N) \in \mathbb{Z}_p^N$ is Bob's, for some integer N. Alice and Bob delegate to a set of $D > 2$ mediators, $M_1, \ldots, M_D$, secret shares in their private vectors. Subsequently, the mediators proceed to perform a multi-party computation on the received secret shares in order to validate the legality of the inputs, if the problem at hand dictates rules by which the input vectors must abide. If the inputs were validated, the mediators proceed to compute secret shares in the required output and then they send those shares to Alice and/or Bob who may then use those shares in order to reconstruct the required output.

We begin by discussing the generic problem of scalar product, in which the required output is the scalar product of the two private input vectors, $\mathbf{a} \cdot \mathbf{b}$. Our distributed scalar product protocol is then used in the subsequent problems that we consider. The first is the problem of oblivious transfer (OT) [15], which is a fundamental building block in MPC [13] and in many application scenarios such as Private Information Retrieval (PIR) [5]. We consider several variants of OT: k-out-of-N OT, priced OT, and generalized OT. Then we deal with the problem of oblivious polynomial evaluation (OPE); here, Alice holds a private uni- or multi-variate polynomial $f(\cdot)$ and Bob holds a private value x. The goal is to let Bob have $f(x)$ so that Alice learns nothing on x while Bob learns nothing on f beyond what is implied by x and $f(x)$.

Mediation helps in significantly simplifying computations that may be much more involved in the case where Alice and Bob have no one else to rely on. In addition, mediation enables carrying out all of the MPC problems that we consider here even when Alice and Bob do not know each other and thus cannot communicate among themselves. In fact, Alice can complete her part in the protocol before even Bob does his. For example, if Alice is a server that hosts a database, then her vector $\mathbf{a}$ could hold decryption keys for the items in her database; the other party, Bob, can be any client that wishes to retrieve one of the items in that database, while keeping Alice oblivious of his choice. Alice and Bob can use our OT protocols for that purpose. But as they do not need to communicate among themselves, but only with the mediators, Bob may perform his retrieval long time after Alice had already uploaded all information relating to her database.

The paper is structures as follows. Section 2 provides the relevant cryptographic preliminaries. In Section 3 we describe our scalar product protocol. Section 4 is devoted to the various OT protocols. In Section 5 we present the OPE protocols. We analyze the communication complexity of all our protocols in Section 6 and report experimental results in Section 7.

2 Preliminaries

Here we provide all necessary background on secret sharing (Section 2.1) and describe our assumptions on the mediators (Section 2.2).

2.1 Secret sharing

The main idea in our protocols for solving the various MPC problems discussed herein is to use secret sharing. Alice and Bob distribute among the D mediators shares in each entry of their private vectors, using t-out-of-D Shamir's secret sharing scheme [17], with

$$t = \lfloor (D+1)/2 \rfloor \,. \tag{1}$$

(Hereinafter we shall refer to such sharing as (t, D)-sharing.) Namely, Alice generates for each entry a_n, $n \in [N] := \{1, \ldots, N\}$, a polynomial $f_n^A(x) = a_n + \sum_{i=1}^{t-1} \alpha_i x^i$, where α_i are secret random field elements, and then she sends to M_d the value $[a_n]_d := f_n^A(d)$, $d \in [D] := \{1, \ldots, D\}$. Bob acts similarly with each entry b_n in his private vector, using another secret generating polynomial $f_n^B(\cdot)$ of degree $t-1$.

The mediators then execute some distributed computation on the shares that they had received in order to arrive at secret shares in the needed output. At the end, they distribute to Alice and/or Bob shares in the desired output from which Alice and/or Bob may reconstruct that output.

The underlying field $\mathbb{Z}_p$ is selected so that p is larger than all values in the underlying computation. If we assume that all entries in the two private vectors are bounded by some $T > 0$, then selecting $p > \max\{NT^2, D\}$ is sufficient.

Our protocols rely on the affinity of secret sharing. Namely, if s_1 and s_2 are two secrets that are independently (t, D)-shared among $M_1, \ldots, M_d$, and a, b, c are three public field elements, then the mediators can compute shares

in $as_1 + bs_2 + c$. Specifically, if $[s_i]_d$ is M_d's share in s_i, $i = 1, 2$, $d \in [D]$, then $\{a[s_1]_d + b[s_2]_d + c : d \in [D]\}$ is a proper (t, D)-sharing of $as_1 + bs_2 + c$.

Unlike affine combinations, the computation of products of shared values in more involved. Namely, given (t, D)-sharing in s_1 and in s_2, the mediators need to engage in an MPC protocol in order to get (t, D)-sharing in $s_1 \cdot s_2$ [7]. However, in order to compute only a single multiplication of shared values it is possible to take a simpler approach: each mediator multiplies locally the shares he holds in the two secrets. It is easy to see that the resulting set of shares, $\{[s_1]_d \cdot [s_2]_d : d \in [D]\}$, is a proper $(2t - 1, D)$-sharing of $s_1 \cdot s_2$. Indeed, assume that $f_i(x) = s_i + \sum_{j=1}^{t-1} c_j x^j$ is the random secret-sharing polynomial that was used to generate shares in s_i, $i = 1, 2$. In that case, the share given to M_d in s_i is $[s_i]_d = f_i(d)$. Define $F(x) = f_1(x) \cdot f_2(x)$. $F(x)$ is a random polynomial of degree $2t - 2$ where $F(0) = s_1 \cdot s_2$. Therefore, $\{[s_1]_d \cdot [s_2]_d : d \in [D]\}$ are the shares in $s_1 \cdot s_2$ as generated by the polynomial F. Since our selection of the threshold t, Eq. (1), ensures that $2t - 1 \leq D$, we thus obtained a proper $(2t - 1, D)$-sharing in the product $s_1 \cdot s_2$.

2.2 Honest majority and security of our protocols

The mediators are assumed to be semi-honest, in the sense that they follow the prescribed protocol, but try to extract from their view in the protocol information on the private inputs. We also assume them to have an *honest majority*, in the sense that if some of them are corrupted by a malicious adversary, the number of corrupted mediators is strictly smaller than the number of the remaining honest mediators. Under such an assumption, the selection of the threshold $t = \lfloor (D + 1)/2 \rfloor$ in the secret sharing scheme guarantees that the mediators will never be able to learn any information on the shared inputs, so that Alice's and Bob's privacy is fully preserved.

3 Distributed Scalar Product

Here we deal with the following MPC problem.

Definition 1. *(DSP) Assume that Alice has a private vector* $\mathbf{a} = (a_1, \ldots, a_N) \in \mathbb{Z}_p^N$*, and Bob has a private vector* $\mathbf{b} = (b_1, \ldots, b_N) \in \mathbb{Z}_p^N$*. They wish to compute their scalar product* $\mathbf{a} \cdot \mathbf{b}$ *without revealing any other information on their private vectors.*

Protocol 1 solves that problem. In the first loop (Lines 1-3), Alice and Bob distribute to the mediators (t, D)-shares in each entry of their vectors. Then, each mediator M_d computes a $(2t-1, D)$-share in $a_n \cdot b_n$ for each $n \in [N]$, and subsequently he computes a $(2t-1, D)$-share in the scalar product into s_d (Line 5). He then sends that share to Alice and Bob (Line 6) who subsequently use those shares to reconstruct the needed scalar product (Line 7).

The protocol is unconditionally secure under the assumption of an honest majority since any subset of less than t mediators cannot extract any information on the private vectors.

Protocol 1: Distributed Scalar Product

Parameters: p - field size, N - the dimension of the vectors, D - number of mediators, $t = \lfloor (D+1)/2 \rfloor$.

Inputs: Alice has a private vector $\mathbf{a} = (a_1, \ldots, a_N) \in \mathbb{Z}_p^N$, Bob has a private vector $\mathbf{b} = (b_1, \ldots, b_N) \in \mathbb{Z}_p^N$.

1 **forall** $n \in [N]$ **do**
2 | Alice sends to M_d, $d \in [D]$, a (t, D)-share in a_n, denoted $[a_n]_d$.
3 | Bob sends to M_d, $d \in [D]$, a (t, D)-share in b_n, denoted $[b_n]_d$.
4 **forall** $d \in [D]$ **do**
5 | M_d computes $s_d \leftarrow \sum_{n \in [N]} ([a_n]_d \cdot [b_n]_d)$.
6 | M_d sends s_d to Alice and Bob.
7 Alice and Bob use any $2t-1$ shares out of $\{s_1, \ldots, s_D\}$ to reconstruct $\mathbf{a} \cdot \mathbf{b}$.

Output: Alice and Bob get $\mathbf{a} \cdot \mathbf{b}$.

4 Distributed Oblivious Transfer

In this section we consider several variants of the Oblivious Transfer (OT) protocol. We begin with the basic variant of k-out-of-N OT in Section 4.1. We then discuss Priced OT (Section 4.2) and Generalized OT (Section 4.3).

4.1 k-out-of-N Oblivious Transfer

The problem that we consider here is the following:

Definition 2. *(OT_k^N) Assume that Alice has a set of N messages, $m_1, \ldots, m_N \in \mathbb{Z}_p$. Bob wishes to learn k of those messages, say $m_{j_1}, \ldots, m_{j_k}$, for some $j_1, \ldots, j_k \in [N]$. A k-out-of-N Oblivious Transfer (OT_k^N) protocol allows Bob to learn $m_{j_1}, \ldots, m_{j_k}$, and nothing beyond those messages, while preventing Alice from learning anything about Bob's selection.*

We begin by considering the case $k = 1$ and then we address the general case. The OT_1^N problem can be reduced to DSP (Section 3) if Alice sets $\mathbf{a} := (m_1, \ldots, m_N)$ and Bob sets $\mathbf{b} := \mathbf{e}_j$ (the unit vector that consists of $N - 1$ zeros and a single 1 in the jth entry, where j is the index of the message that Bob wishes to retrieve). However, the DSP protocol cannot be executed naïvely, since Bob may cheat and send to the mediators shares in a vector that is not a unit vector and, consequently, he may obtain some linear combination of the messages, and not just a single message as dictated by the OT definition. Such an abuse of the protocol may enable Bob in some cases to learn more than just one message. For example, if Bob happens to know that m_1 belongs to some one-dimensional subspace of $\mathbb{Z}_p^N$ while m_2 belongs to another one-dimensional subspace of $\mathbb{Z}_p^N$, then by choosing to learn the linear combination $m_1 + m_2$ he will be able to infer both m_1 and m_2. To that end, the DSP protocol can be executed only after the mediator apply some preliminary validation protocol:

Definition 3. *(DVV) Assume that the mediators $M_1, \ldots, M_D$ hold (t, D)-shares in a vector $\mathbf{v} \in \mathbb{Z}_p^N$. Let W be a subset of $\mathbb{Z}_p^N$. A Distributed Vector Validation (DVV) protocol is a protocol that the mediators may execute on their shares that outputs 1 if $\mathbf{v} \in W$ and 0 otherwise, and reveals no further information on $\mathbf{v}$ in the case where $\mathbf{v} \in W$.*

In our case $W = \{\mathbf{e}_j : j \in [N]\}$. The mediators can validate that $\mathbf{b} \in W$ by verifying the following two conditions: (1) $b_n \cdot (b_n - 1) = 0$ for all $n \in [N]$; and (2) $\sum_{n \in [N]} b_n = 1$. Indeed, the first condition implies that all entries in $\mathbf{b}$ are either 0 or 1, while the second condition ascertains that exactly one of the entries equals 1. Note that if the two conditions are verified, then the mediators may infer that Bob's vector is legal, but nothing more than that, as desired. Namely, if Bob is honest then his privacy is fully protected. However, if Bob is dishonest and distributed shares in a vector $\mathbf{b} \notin W$, then the above described DVV protocol will reveal some additional information on $\mathbf{b}$; however, that is acceptable since by acting dishonestly Bob looses his right for privacy.

Protocol 2 implements those ideas. After Alice and Bob set their vectors and distribute shares in them to the mediators (Lines 1-5), the mediators validate Bob's vector for compliance with conditions 1 (Lines 6-10) and 2 (Lines 11-15). If Bob's vector was validated, they compute $(2t-1, D)$-shares in the scalar product and send them to Bob so that he can recover the scalar product that equals his message of choice (Lines 16-19).

For a general $k > 1$, it is possible to solve OT_k^N by running Protocol 2 k times, with one exception: Alice needs to distribute shares in her vector only once (Lines 1 and 4 in Protocol 2). We proceed to describe an alternative solution, Protocol 3, and then we compare the efficiency of the two solutions.

Protocol 3 multiplies Alice's vector $\mathbf{a} := (m_1, \ldots, m_N)$ with the vector $\mathbf{b} = \sum_{i=1}^{k} \mathbf{e}_{j_i}$ where $1 \leq j_1 < \ldots < j_k \leq m$ are the indices of the k messages that Bob wishes to retrieve. But instead of computing their scalar product, $\sum_{n=1}^{N} a_n b_n$, the protocol computes shares in the products $a_n b_n$ for all $n \in [N]$ and sends them to Bob. Bob then uses the shares of $a_n b_n$ only for $n \in \{j_1, \ldots, j_k\}$ in order to recover the requested messages.

Here, the DVV sub-protocol consists of verifying two conditions: that $b_n \cdot (b_n - 1) = 0$ for all $n \in [N]$, and that $\sum_{n \in [N]} b_n = k$. The first condition implies that all entries in $\mathbf{b}$ are either 0 or 1, while the second condition ascertains that exactly k of the entries equal 1.

After Alice and Bob set their vectors and distribute shares in them to the mediators (Lines 1-5), the mediators validate Bob's vector for compliance with conditions 1 (Lines 6-10) and 2 (Lines 11-15). If Bob's vector was validated, they compute $(2t-1, D)$-shares in each of the N products between the components of the two vectors and send them to Bob (Lines 16-19) for him to recover the requested k messages (Lines 20-21).

Protocol 2: 1-out-of-N Oblivious Transfer

Parameters: p - field size, N - number of messages, D - number of mediators, $t = \lfloor (D+1)/2 \rfloor$.

Inputs: Alice has $U = \{m_1, \ldots, m_N\}$; Bob has a selection index $j \in [N]$.

Alice sets $\mathbf{a} = (m_1, \ldots, m_N)$.
Bob sets $\mathbf{b} = \mathbf{e}_j$.
forall $n \in [N]$ **do**
    Alice sends to M_d, $d \in [D]$, a (t, D)-share in a_n, denoted $[a_n]_d$.
    Bob sends to M_d, $d \in [D]$, a (t, D)-share in b_n, denoted $[b_n]_d$.
forall $1 \leq n \leq N$ **do**
    Each M_d, $d \in [D]$, sets $[c_n]_d = [b_n]_d \cdot ([b_n]_d - 1)$.
    The mediators use any $2t-1$ shares out of $\{[c_n]_1, \ldots, [c_n]_D\}$ to compute $\omega := b_n \cdot (b_n - 1)$.
    if $\omega \neq 0$ **then**
        Abort
forall $d \in [D]$ **do**
    M_d computes $c_d \leftarrow \sum_{n \in [N]} [b_n]_d$.
The mediators use any t shares out of $\{c_1, \ldots, c_D\}$ to compute $\omega := \sum_{n \in [N]} b_n$.
if $\omega > 1$ **then**
    Abort
forall $d \in [D]$ **do**
    M_d computes $s_d \leftarrow \sum_{n \in [N]} ([a_n]_d \cdot [b_n]_d)$.
    M_d sends s_d to Bob.
Bob uses any $2t-1$ shares out of $\{s_1, \ldots, S_D\}$ to reconstruct $\mathbf{a} \cdot \mathbf{b} = m_j$.

Output: Bob gets m_j.

4.2 Priced Oblivious Transfer

Consider a setting of OT in which each of Alice's messages has a weight and the retrieval policy allows Bob to learn any subset of messages in which the sum of weights does not exceed some given threshold. For example, if Alice holds a database of movies and each movie has a price tag, then if Bob had prepaid some amount, Alice wishes to guarantee that he retrieves movies of

Protocol 3: k-out-of-N Oblivious Transfer

Parameters: p - field size, N - number of messages, D - number of mediators, $t = \lfloor (D+1)/2 \rfloor$.

Inputs: Alice has $U = \{m_1, \ldots, m_N\}$; Bob has selection indices $1 \leq j_1 < \ldots < j_k \leq N$.

Alice sets $\mathbf{a} = (m_1, \ldots, m_N)$.

Bob sets $\mathbf{b} = (b_1, \ldots, b_N)$, where $b_n = 1$ for $n \in \{j_1, \ldots, j_k\}$ and $b_n = 0$ otherwise.

forall $n \in [N]$ **do**

  Alice sends to M_d, $d \in [D]$, a (t, D)-share in a_n, denoted $[a_n]_d$.

  Bob sends to M_d, $d \in [D]$, a (t, D)-share in b_n, denoted $[b_n]_d$.

forall $1 \leq n \leq N$ **do**

  Each M_d, $d \in [D]$, sets $[c_n]_d = [b_n]_d \cdot ([b_n]_d - 1)$.

  The mediators use any $2t-1$ shares out of $\{[c_n]_1, \ldots, [c_n]_D\}$ to compute $\omega := b_n \cdot (b_n - 1)$.

  if $\omega \neq 0$ **then**

    Abort

forall $d \in [D]$ **do**

  M_d computes $c_d \leftarrow \sum_{n \in [N]} [b_n]_d$.

The mediators use any t shares out of $\{c_1, \ldots, c_D\}$ to compute $\omega := \sum_{n \in [N]} b_n$.

if $\omega > k$ **then**

  Abort

forall $d \in [D]$ **do**

  forall $n \in [N]$ **do**

    M_d computes $[c_n]_d \leftarrow [a_n]_d \cdot [b_n]_d$.

    M_d sends $[c_n]_d$ to Bob.

forall $n \in \{j_1, \ldots, j_k\}$ **do**

  Bob uses any $2t-1$ shares out of $\{[c_n]_1, \ldots, [c_n]_D\}$ to reconstruct $c_n = a_n \cdot b_n = m_n$.

Output: Bob gets $m_{j_1}, \ldots, m_{j_k}$.

aggregated cost that does not exceed what he had paid, while Bob wishes to prevent Alice from knowing what movies he chose to watch.

Definition 4. *Let $U = \{m_1, \ldots, m_N\}$ be the set of messages that Alice has. Assume that each massage m_n has a weight $w_n \geq 0$, $n \in [N]$, and let $T > 0$ be*

some given threshold. Then a Priced OT protocol allows Bob to retrieve any subset $B \subseteq U$ for which $\sum_{m_n \in B} w_n \leq T$. Bob cannot learn any information on the messages in $U \setminus B$, while Alice has to remain oblivious of Bob's choice.

We assume that the weights $w_1, \ldots, w_N$ are publicly known, since they represent information that is supposed to be known to all. The threshold T, on the other hand, represents the amount that Bob had paid and, therefore, it is private and should remain so.

Protocol 4 executes Priced OT. It coincides with Protocol 3 except for the second part of the DVV sub-protocol. If in Protocol 3 the mediators obliviously verified that $\sum_{n \in [N]} b_n \leq k$ (Lines 11-15 there), then here it is necessary to obliviously verify that $\sum_{m_n \in B} w_n = \sum_{n \in [N]} w_n b_n \leq T$. To enable that verification, the protocol starts by publishing the vector of weights (Line 1). Then, both Alice and Bob distribute to the mediators (t, D)-shares in T (Lines 2-3) and then the mediators verify that the two underlying thresholds equal, without recovering that threshold (Lines 4-7). Those steps are necessary in order to ensure both Alice and Bob that the correct threshold is used in the DVV sub-protocol. (Namely, Bob is ascertained that Alice did not provide a too low value of T while Alice is ascertained that Bob did not provide a too high value of T).

The core of the protocol is the execution of the OT_k^N protocol - Protocol 3 (Line 8). That protocol is executed as is except for the replacement of Lines 11-15 there with Sub-protocol 5. The sub-protocol begins with the mediators computing (t, D)-shares in the difference $e := T - \sum_{n \in [N]} w_n b_n$ (Lines 1-2). Then, any subset of t mediators can recover e (Line 3). Finally, if $e < 0$ the protocol aborts (Line 4), while otherwise it proceeds towards completing the transfer.

Sub-protocol 5 reveals to the mediators the difference $e = T - \sum_{n \in [N]} w_n b_n$. Such information leakage can be reduced if Bob adds to his list of retrieved messages additional redundant messages that he will ignore later on. That way, the difference can be made a nonnegative number smaller than $\overline{w} := \max_{n \in [N]} w_n$. Such a difference reveals no meaningful information neither on Bob's threshold nor on his selections. However, if desired, it is possible to eliminate even that information leakage: assume that $\overline{w} < 2^\ell$; then Alice may add ℓ phantom messages $\hat{m}_i$, $0 \leq i < \ell$, with the weights 2^i, and then Bob will add to his list of requested messages also the subset of phantom messages of which the sum of weights equals exactly e. That way, the mediators will always recover in Line 3 in Sub-protocol 5 the value 0.

4.2.1 The case of secret weights

Even though the weights of messages are typically public, it is possible to modify the protocol so that also the weights remain hidden from the mediators. To do that, instead of publishing the vector of weights $\mathbf{w}$ (as done in Line 1 of Protocol 4), Alice would distribute to the mediators (t, D)-shares in them. Let $[w_n]_d$ denote M_d's share in w_n, $d \in [D]$, $n \in [N]$. Then, in Sub-protocol 5, Line 2 will be replaced with $[e]_d \leftarrow [T]_d - \sum_{n\in[N]}[w_n]_d[b_n]_d$. As discussed in Section 2.1, the set $\{[e]_1, \ldots, [e]_D\}$ would be then a set of $(2t-1, D)$-shares in e. Hence, in Line 3, $2t-1$ mediators will collaborate in recovering e. No further changes are required.

Protocol 4: Priced Oblivious Transfer

Parameters: p - field size, N - number of messages, D - number of mediators, $t = \lfloor (D+1)/2 \rfloor$.

Inputs: Alice has $U = \{m_1, \ldots, m_N\}$, and corresponding weights $w_n \geq 0$, $n \in [N]$; Bob has a set of selection indices $j_1, \ldots, j_k \in [N]$; Alice and Bob have $T \geq 0$.

1 Alice publishes the vector of weights $\mathbf{w} = (w_1, \ldots, w_N)$.
2 Alice sends to M_d, $d \in [D]$, a (t, D)-share in T, denoted $[T]_d$.
3 Bob sends to M_d, $d \in [D]$, a (t, D)-share in T, denoted $[T']_d$.
4 **forall** $d \in [D]$ **do**
5 | M_d computes $[e]_d \leftarrow [T]_d - [T']_d$.
6 The mediators use any t shares out of $\{[e]_1, \ldots, [e]_D\}$ to reconstruct $e = T - T'$.
7 **if** $e \neq 0$ **then Abort**.
8 Alice, Bob and the mediators execute Protocol 3 in which Lines 11-15 are replaced with Sub-protocol 5.

Output: Bob gets $\{m_{j_1}, \ldots, m_{j_k}\}$ iff $\sum_{i=1}^{k} w_{j_i} \leq T$.

4.3 Generalized Oblivious Transfer

Ishai and Kushilevitz [11] presented an extension of OT called *Generalized Oblivious Transfer* (GOT). As in the the basic version of OT, Definition 2, we consider a setting with two parties, Alice and Bob. Alice has a set of N messages, $m_1, \ldots, m_N$, that can be viewed as elements in a finite field $\mathbb{Z}_p$. Bob wishes to learn a subset of those messages, according to some retrieval

Sub-protocol 5: Priced OT: verifying that $\sum_{i=1}^{k} w_{j_i} \leq T$.

forall $d \in [D]$ **do**
  M_d computes $[e]_d \leftarrow [T]_d - \sum_{n\in[N]} w_n[b_n]_d$.
The mediators use any t shares out of $\{[e]_1, \ldots, [e]_D\}$ to $e = T - \sum_{n\in[N]} w_n b_n$.
if $e < 0$ **then Abort**.

policy. In OT_k^N, the policy restricted Bob to learn any subset of at most k messages. In GOT the policy is extended as described below.

Definition 5. *Let $U = \{m_1, \ldots, m_N\}$ be the set of messages that Alice has. An access structure is a collection of subsets of U, $\mathcal{A} \subseteq 2^U$, which is monotone decreasing in the sense that if $B \in \mathcal{A}$ and $B' \subset B$ then also $B' \in \mathcal{A}$. The basis of $\mathcal{A}$, denoted $\mathcal{A}^0$, is the collection of all maximal subsets in $\mathcal{A}$; namely, $B \in \mathcal{A}^0$ if $B \in \mathcal{A}$ and for every $B \subsetneq B' \subseteq U$, $B' \notin \mathcal{A}$.*

Bob is allowed to retrieve any subset of messages $B \subset U$ provided that $B \in \mathcal{A}$. As before, Bob cannot learn any information on the complement set of messages, $U \setminus B$, while Alice must not learn any information on the subset B that Bob chose to retrieve.

The distributed GOT protocol that we present here, Protocol 6, is based on the GOT protocol that was presented in [19], and it invokes the OT_k^N protocol, Protocol 3. Protocol 6 is designed for the case of uniform bases, namely, the case where all subsets in $\mathcal{A}^0$ have the same size, denoted k. The case of non-uniform bases can be reduced to the case of uniform bases as described in [19]. We refer the reader to [19] for a detailed description of the simple reduction.

The protocol is based on secret sharing, as we proceed to explain. Let us define the *monotone increasing closure* of $\mathcal{A}^0$ as follows:

$$\Gamma = \Gamma(\mathcal{A}^0) = \{C \subseteq U : \exists B \in \mathcal{A}^0, B \subseteq C\}. \tag{2}$$

The collection Γ is monotone increasing, in the sense that if $B \in \Gamma$ and $B \subset B' \subseteq U$, then also $B' \in \Gamma$. Let Σ be a secret sharing scheme that realizes Γ, in the following sense. It is a secret sharing scheme in which the set of participants is the set U of N messages, and the access structure is Γ. Given a secret s, the scheme Σ assigns a share s_n to each message m_n,

$n \in [N]$, so that the shares of any subset in Γ reveal s while the shares of any other subset reveal no information on s.

Protocol 6 starts with Alice selecting a secret $s \in \mathbb{Z}_p$ (Line 1). Then she computes corresponding shares in s according to the access structure $\Gamma(\mathcal{A}^0)$ (Line 2). Namely, if $B = \{m_{j_1}, \ldots, m_{j_k}\} \in \mathcal{A}^0$ is a set of messages that Bob is allowed to retrieve, the corresponding set of shares, $\{s_{j_1}, \ldots, s_{j_k}\}$, can be used to reconstruct s; otherwise, those shares reveal no information on s.

Next, Alice generates random masks x_n for all her messages and uses them to perturb m_n into m'_n, $n \in [N]$. Then, she packs those perturbed messages together with the corresponding secret shares into m''_n, $n \in [N]$ (Lines 3-6). She then proceeds to distribute to the mediators (t, D)-shares in the secret s as well as in the masks x_n, $n \in [N]$ (Lines 7-9).

The core of the protocol is in Line 10: here, Alice and Bob engage in the distributed OT_k^N protocol where Bob chooses to learn m''_n for all $n \in \{j_1, \ldots, j_k\}$. Note that this protocol takes places over the field $\mathbb{Z}_q$ whose size is twice that of $\mathbb{Z}_p$, since Bob retrieves here a pairing of the (perturbed) messages and the corresponding secret shares.

In the reminder of the protocol Bob retrieves the masks that would enable him to extract the messages m_n from m''_n for all $n \in \{j_1, \ldots, j_k\}$. First, he decouples s_n from m'_n for all k messages he chose (Lines 11-13). Using the secret shares s_n, $n \in \{j_1, \ldots, j_k\}$, he reconstructs $s^B := s$ according to the reconstruction function of the secret sharing scheme Σ (Line 14). He then distributes to the mediators (t, D)-shares in s^B (Line 15). The mediators proceed to verify that $s^B = s^A = s$ without actually recovering s (Lines 16-17). If the difference $e = s^A - s^B$ is zero, then Bob had retrieved a subset of k messages that he was allowed to retrieve. In that case, the mediators send him shares in all random masks; Bob reconstructs only the relevant random masks and then recovers the k messages of his choice (Lines 19-21). If, however, $e \neq 0$ then Bob failed to prove that he attempted retrieving an allowed subset of messages; in that case the protocol aborts (Line 23).

We note that Protocol 6 offers *almost* perfect privacy. Bob may attempt guessing the value of $s \in \mathbb{Z}_p$. If he succeeds (in negligible probability of $1/p$) he may be able to learn any subset of k messages. However, if he fails, the mediators would infer that he attempted cheating and could refuse to engage in further attempts.

Few comments are in order before we proceed:

1. For the sake of simplicity, we assumed that the access structure $\Gamma(\mathcal{A}^0)$, Eq. (2), is ideal in the sense that there exists a secret sharing scheme Σ that

realizes it, in which all secret shares $s_1, \ldots, s_N$ are taken from the same field $\mathbb{Z}_p$ as the secret s. In cases where $\Gamma(\mathcal{A}^0)$ is not ideal, or in cases where $\Gamma(\mathcal{A}^0)$ is ideal, but the selected secret sharing scheme Σ is not ideal[1], then the shares $s_1, \ldots, s_N$ cannot be taken from $\mathbb{Z}_p$. Assume that in such a case all shares can be embedded in $\mathbb{Z}_r$ for some prime $r \geq p$. Then Protocol 6 works exactly as described, where q is a prime greater than $p \cdot r$.

2. Note that Alice performs secret sharing on s in two different places in Protocol 6 and in two entirely different ways. In Line 2, Alice secret-shares s among the set of participants $U = \{m_1, \ldots, m_N\}$ where the access structure is Γ; the secret sharing scheme here is Σ. Later on, in Line 7, Alice secret-shares the same value s among the set of participants $\{M_1, \ldots, M_D\}$, i.e., the mediators, where the access structure is a simple t-out-of-D threshold access structure and t is as defined in Eq. (1); the secret sharing scheme here is the standard Shamir threshold secret sharing scheme [17]. The purpose of the first secret sharing is to ensure that Bob can retrieve only subsets of k messages from $\mathcal{A}^0$. The purpose of the second secret sharing scheme is to enable the mediators to verify that the value of s that Alice used equals the value of s that Bob sends to them, without actually knowing s. (In a simpler implementation, Alice could have sent the value of s to the mediators, without secret sharing. But then if Bob is able to corrupt a single mediator, he could get from that mediator the value of s and then Bob would be able to learn any subset of k messages. That is something that we prevent in Protocol 6 which is secure under the assumption that the majority of mediators are honest.)

3. As noted already in Section **??**, Alice and Bob do not need to be active at the same time. For example, in the case of Generalized OT, Alice can complete her part — Lines 1-9 in Protocol 6 and the relevant part of Protocol 3 — and then go offline; only when the need arises, Bob can initiate the completion of Protocol 3 and the completion of Protocol 6 (Lines 11-23). The same holds also for Priced OT and OT_k^N. When Alice is a server that serves many "Bob" clients, Alice may complete her part and then let the mediators attend to the request of any future client Bob.

[1]It is possible that a non-ideal secret sharing scheme could be simpler and easier to implement than an equivalent ideal secret sharing scheme that realizes the same access structure.

Protocol 6: Generalized Oblivious Transfer

Parameters: p, q - two primes where $q > p^2$, N - number of messages, D - number of mediators, $t = \lfloor (D+1)/2 \rfloor$.

Inputs: Alice has $U = \{m_1, \ldots, m_N\} \subset \mathbb{Z}_p$ and an access structure $\mathcal{A}$ on U, with a k-uniform basis $\mathcal{A}^0$; Bob has indices $1 \leq j_1 < \ldots < j_k \leq N$, where $B := \{m_{j_1}, \ldots, m_{j_k}\} \in \mathcal{A}^0$.

Alice selects uniformly at random a secret $s \in \mathbb{Z}_p$.
Alice computes shares $\{s_1, \ldots, s_N\} \subset \mathbb{Z}_p$ in s using a secret sharing scheme Σ that realizes the access structure $\Gamma(\mathcal{A}^0)$ on U.
forall $n \in [N]$ **do**
    Alice selects independently at random $x_n \in \mathbb{Z}_p$.
    Alice defines $m'_n = m_n + x_n \bmod p$.
    Alice sets $m''_n = m'_n + p \cdot s_n \in \mathbb{Z}_q$.
Alice distributes to the mediators (t, D)-shares in $s^A := s$; M_d's share is denoted $[s^A]_d$, $d \in [D]$.
forall $n \in [N]$ **do**
    Alice sends to M_d, $d \in [D]$, a (t, D)-share in x_n, denoted $[x_n]_d$.
The parties execute Protocol 3 so Bob gets $B := \{m''_{j_1}, \ldots, m''_{j_k}\}$.
forall $n \in \{j_1, \ldots, j_k\}$ **do**
    Bob computes $m'_n = m''_n \bmod p$.
    Bob computes $s_n = (m''_n - m'_n)/p$.
Bob recovers $s^B := s$ from $\{s_n : n \in \{j_1, \ldots, j_k\}\}$ using the reconstruction function of the secret sharing scheme Σ.
Bob distributes to the mediators (t, D)-shares in the secret s^B that he had computed above; M_d's share is denoted $[s^B]_d$, $d \in [D]$.
M_d, for all $d \in [D]$, computes $[e]_d = [s^A]_d - [s^B]_d$.
The mediators recover $e := s^A - s^B$ from any t shares out of $\{[e]_d : d \in [D]\}$.
if $e = 0$ **then**
    t of the mediators send to Bob their vectors of shares $([x_1]_d, \ldots, [x_N]_d)$, $d \in [D]$.
    Bob recovers from those vectors the masks x_n, $n \in \{j_1, \ldots, j_k\}$.
    Bob computes $m_n = m'_n - x_n \bmod p$ for all $n \in \{j_1, \ldots, j_k\}$.
    else
        Abort

Output: Bob gets $m_{j_1}, \ldots, m_{j_k}$.

4.3.1 Exemplifying GOT for compartmented message sets

Assume that the set of messages, $U = \{m_1, \ldots, m_N\}$, is split into r disjoint subsets, called compartments,

$$U = \bigcup_{i=1}^{r} U_i\,, \qquad U_i \cap U_j = \emptyset\,,\ 1 \le i < j \le r\,.$$

Bob is allowed to retrieve messages only from one of those compartments. Hence, the access structure here is

$$\mathcal{A} = \{B \subset U : |B| \subseteq U_i \text{ for some } 1 \le i \le r\}\,.$$

The basis of this access structure is $\mathcal{A}^0 = \{U_i : 1 \le i \le r\}$, and its monotone increasing closure is

$$\Gamma = \Gamma(\mathcal{A}^0) = \{B \subset U : B \supseteq U_i \text{ for some } 1 \le i \le r\}\,. \tag{3}$$

The access structure in Eq. (3) is a simple case of a compartmented access structure [3, 20], namely, one in which the participants (messages) are split into disjoint compartments, and all participants within the same compartment play the same role in the access structure. The access structure Γ can be easily realized as follows. (What follows is the computation that Alice does in Line 2 of Protocol 6 in case her access structure is as described above.)

Alice selects a random secret $s \in \mathbb{Z}_p$ and then, for each compartment U_i, $1 \le i \le r$, she will assign to all messages in that compartment random secret shares that add up to s. Specifically, if $U_i = \{m_{j_h} : 1 \le h \le |U_i|\}$ then Alice selects uniformly at random $|U_i| - 1$ secret shares, $s_{j_h} \in \mathbb{Z}_p$, $1 \le h \le |U_i| - 1$, and then she sets $s_{j_{|U_i|}} = s - \sum_{h=1}^{|U_i|-1} s_{j_h} \bmod p$.

5 Oblivious polynomial evaluation

Here we consider the distributed versions of the oblivious polynomial evaluation problem [14] and its multivariate extension [22].

5.1 Univariate polynomials

The problem of oblivious evaluation of univariate polynomials is defined as follows.

Definition 6. *(OPE) Assume that Alice has a polynomial* $f(x) = \sum_{n=0}^{N} c_n x^n$ *of degree at most* N *over the field* $\mathbb{Z}_p$*, while Bob has a value in the field* $\alpha \in \mathbb{Z}_p$*. They wish to enable Bob to learn* $f(\alpha)$*, and nothing else on* f*, while Alice remains oblivious to* α*.*

We propose here a solution that is based on DSP (Section 3). In our solution (Protocol 7) Alice sets a vector of the polynomial coefficients $\mathbf{a} = (c_0, \ldots, c_N)$, while Bob sets a vector of the powers of his secret value α,

$$\mathbf{b} = (\alpha^0, \alpha^1, \ldots, \alpha^N)\,. \tag{4}$$

With that setting of the two vectors, their scalar product $\mathbf{a} \cdot \mathbf{b}$ equals $f(\alpha)$. Hence, OPE can be reduced to DSP. However, while Alice's vector $\mathbf{a}$ can be any N-dimensional vector, Bob's vector $\mathbf{b}$ must be of the form as in Eq. (4), for some field element α. Namely, the DVV sub-protocol (Definition 3) must ascertain that $\mathbf{b} \in W := \{(\alpha^0, \alpha^1, \ldots, \alpha^N) : \alpha \in \mathbb{Z}_p\}$, while if $\mathbf{b}$ is indeed in W, the DVV sub-protocol must not leak any further information beyond that.

Our OPE protocol is presented in Protocol 7. First, Alice and Bob set their secret vectors $\mathbf{a}$ and $\mathbf{b}$ to be submitted to the secure DSP computation (Lines 1-2). Then, they distribute to the mediators (t, D)-secret shares in those vectors (Lines 3-5). Note that the first entry in Bob's vector is always 1. Hence, there is no need for Bob to generate and distribute shares in that entry. Instead, each mediator M_d, $d \in [D]$, can internally set $[b_1]_d = 1$. To keep our pseudo-code simple we retain the description of the shares' generation and distribution in Lines 3-5, but the execution of Line 5 for $n = 1$ will be as described above.

The DVV sub-protocol takes place in Lines 6-9. If the validation fails, the protocol aborts. We elaborate below on the details of this computation. If the validation passes successfully, the mediators proceed to compute $(2t-1, D)$-shares in the scalar product (that equals $f(\alpha)$) and send those shares to Bob only (Lines 10-12). Bob proceeds to reconstruct the required output value (Line 13).

The DVV in Lines 6-9 is based on the observation that $\mathbf{b}$ is of the form as in Eq. (4) if and only if the value ω that is computed in Line 7 equals 0 for every $3 \leq n \leq N+1$. In order to compute ω, each mediator computes

$$[c]_d := [b_{n-1}]_d \cdot [b_2]_d - [b_n]_d\,. \tag{5}$$

Protocol 7: Oblivious Polynomial Evaluation

Parameters: p - field size, N - the degree of the secret polynomial f, D - number of mediators, $t = \lfloor (D+1)/2 \rfloor$.

Inputs: Alice has a secret polynomial $f(x) = \sum_{n=0}^{N} c_n x^n$ over $\mathbb{Z}_p$; Bob has a secret value $\alpha \in \mathbb{Z}_p$.

Alice sets $\mathbf{a} = (c_0, \ldots, c_N) := (a_1, \ldots, a_{N+1})$.
Bob sets $\mathbf{b} = (\alpha^0, \alpha^1, \ldots, \alpha^N) := (b_1, \ldots, b_{N+1})$.
forall $n \in [N+1]$ **do**
    Alice sends to M_d, $d \in [D]$, a (t, D)-share in a_n, denoted $[a_n]_d$.
    Bob sends to M_d, $d \in [D]$, a (t, D)-share in b_n, denoted $[b_n]_d$.
forall $3 \leq n \leq N+1$ **do**
    The mediators compute $\omega := b_{n-1} b_2 - b_n$.
    if $\omega \neq 0$ **then**
        Abort
forall $d \in [D]$ **do**
    M_d computes $s_d \leftarrow \sum_{n \in [N]} ([a_n]_d \cdot [b_n]_d)$.
    M_d sends s_d to Bob.
Bob uses any $2t-1$ shares out of $\{s_1, \ldots, S_D\}$ to reconstruct $\mathbf{a} \cdot \mathbf{b} = f(\alpha)$.

Output: Bob gets $f(\alpha)$.

In view of our discussion in Section 2.1, the set $\{[b_{n-1}]_d \cdot [b_2]_d : d \in [D]\}$ is a set of $(2t-1, D)$-shares in $b_{n-1} b_2$. In addition, as $\{[b_n]_d : d \in [D]\}$ is a set of (t, D)-shares in b_n, it is also a set of $(2t-1, D)$-shares in b_n (because $t \leq 2t-1$). Hence, by the affinity of secret sharing, the set $\{[c]_d : d \in [D]\}$, where $[c]_d$ is as defined in Eq. (5), is a set of $(2t-1, D)$-shares in $\omega = b_{n-1} b_2 - b_n$. Therefore, the mediators can use any $2t-1$ shares from that set in order to recover ω. If the recovered value is zero, then they had validated that $w = b_{n-1} b_2 - b_n = 0$ without learning any further information on $\mathbf{b}$. Otherwise, they infer that $\mathbf{b}$ is not of the form as in Eq. (4) so they abort.

Protocol 7 provides perfect privacy for Alice and Bob due to the assumption of honest majority for the set of mediators. We note that if Bob is dishonest and submits an illegal vector $\mathbf{b}$ to the protocol, the mediators will detect that by running into a nonzero value ω. In that case, ω reveals the value of $b_{n-1} b_2 - b_n$, for some index $3 \leq n \leq N+1$. Therefore, the mediators reveal in that case some information on $\mathbf{b}$ beyond the fact that it is

not in the form of Eq. (4). Hence, as we have already noted earlier, the DVV sub-protocol offers perfect privacy only for honest parties, as indeed intended.

5.2 Multivariate polynomials

Here we consider the problem of oblivious evaluation of multivariate polynomials. We begin by defining multivariate polynomials (Definitions 7 and 8) and then define the corresponding MPC problem (Definition 9).

Definition 7. *(Monomial) Let $\mathbb{Z}_p$ be a finite field , $\mathbf{x} = (x_1, \ldots, x_k)$ be a k-dimensional vector over $\mathbb{Z}_p$ and $\mathbf{j} = (j_1, \ldots, j_k)$ be a k-dimensional vector of nonnegative integers. Then the monomial $\mathbf{x}^{\mathbf{j}}$ is defined as $\mathbf{x}^{\mathbf{j}} := \prod_{i=1}^{k} x_i^{j_i}$.*

Definition 8. *(Multivariate Polynomial) let $\mathbb{Z}_+^k := \{\mathbf{j} = (j_1, \ldots, j_k) : j_i \in \mathbb{Z}_+ = \{0, 1, 2, \ldots\} : 1 \leq i \leq k\}$ be the set of all k-tuples of nonnegative integers, and $\mathbb{Z}_+^{k,N}$ be the subset of $\mathbb{Z}_+^k$ consisting of all tuples of which the sum of components is at most N, i.e: $\mathbb{Z}_+^{k,N} := \{\mathbf{j} \in \mathbb{Z}_+^k : |\mathbf{j}| := \sum_{i=1}^{k} j_i \leq N\}$. An N-degree k-variate polynomial $f(\mathbf{x})$ over the field $\mathbb{Z}_p$, where $\mathbf{x} = (x_1, \ldots, x_k) \in \mathbb{Z}_p^k$, is defined as:*

$$f(\mathbf{x}) = \sum_{\mathbf{j} \in \mathbb{Z}_+^{k,N}} a_{\mathbf{j}} \cdot \mathbf{x}^{\mathbf{j}}, \quad a_{\mathbf{j}} \in \mathbb{Z}_p. \tag{6}$$

Definition 9. *(OMPE) Assume that Alice has an N-degree multivariate polynomial $f(\mathbf{x}) = f(x_1, \ldots, x_k)$, while Bob has a point $\boldsymbol{\alpha} = (\boldsymbol{\alpha}_1, \ldots, \boldsymbol{\alpha}_k) \in \mathbb{Z}_p^k$. They wish to enable Bob to learn $f(\boldsymbol{\alpha})$, and nothing else on f, while keeping Alice oblivious to $\boldsymbol{\alpha}$.*

Similarly to our OPE protocol, Protocol 7, also OMPE can be solved by reducing it to DSP, with the needed prior validations. The vector that Alice will submit to the protocol consists of the coefficients of her polynomial, $\mathbf{a} = (a_{\mathbf{j}} : \mathbf{j} \in \mathbb{Z}_+^{k,N})$. The vector that Bob will submit to the protocol is the following:

$$\mathbf{b} = (b_{\mathbf{j}} : \mathbf{j} \in \mathbb{Z}_+^{k,N}), \quad \text{where} \quad b_{\mathbf{j}} := \boldsymbol{\alpha}^{\mathbf{j}}. \tag{7}$$

It is easy to see that the dimension of these vectors is $\binom{N+k}{k}$.

First, it is necessary to agree upfront on an ordering of $\mathbb{Z}_+^{k,N}$ so that in the scalar product between the two vectors, each power of $\boldsymbol{\alpha}$ will be multiplied by

the corresponding polynomial coefficient. We suggest ordering the set $\mathbb{Z}_+^{k,N}$ by arranging its monomials into $N+1$ tiers, as follows. The 0th tier would be $T_0 := Z_+^{k,0}$, and then the nth tier, $n = 1, \ldots, N$, would be $T_n := Z_+^{k,n} \setminus Z_+^{k,n-1}$; namely, the nth tier T_n consists of all monomials of degree exactly $n \in \{0, 1, \ldots, N\}$. The order within each tier would be lexicographical.

Protocol 8: Oblivious Multivariate Polynomial Evaluation

Parameters: p - field size, k-number of variables, N - the degree of the secret polynomial f, D - number of mediators, $t = \lfloor (D+1)/2 \rfloor$.

Inputs: Alice has a secret$N$-degree k-variate polynomial $f(\mathbf{x})$, Eq. (6); Bob has a secret point $\boldsymbol{\alpha} = (\boldsymbol{\alpha}_1, \ldots, \boldsymbol{\alpha}_k) \in \mathbb{Z}_p^k$.

1 Alice sets $\mathbf{a} = (a_{\mathbf{j}} : \mathbf{j} \in \mathbb{Z}_+^{k,N})$, according to the ordering convention.

2 Bob sets $\mathbf{b} = (b_{\mathbf{j}} = \boldsymbol{\alpha}^{\mathbf{j}} : \mathbf{j} \in \mathbb{Z}_+^{k,N})$, according to the ordering convention.

3 **forall** $\mathbf{j} \in \mathbb{Z}_+^{k,N}$ **do**

4 Alice sends to M_d, $d \in [D]$, a (t, D)-share in $a_{\mathbf{j}}$, denoted $[a_{\mathbf{j}}]_d$.

5 Bob sends to M_d, $d \in [D]$, a (t, D)-share in $b_{\mathbf{j}}$, denoted $[b_{\mathbf{j}}]_d$.

6 **forall** $2 \leq n \leq N$ **do**

7 **forall** $\mathbf{j} \in T_n$ **do**

8 Select a monomial $\mathbf{h} \in T_{n-1}$ such that $\mathbf{j} = \mathbf{h} + \mathbf{e}_i$ for some $1 \leq i \leq k$, where $\mathbf{e}_i$ is the i-th unit vector.

9 The mediators compute $\omega := b_{\mathbf{h}} \cdot b_{\mathbf{e}_i} - b_{\mathbf{j}}$.

10 **if** $\omega \neq 0$ **then**

11 **Abort**

12 **forall** $d \in [D]$ **do**

13 M_d computes $s_d \leftarrow \sum_{\mathbf{j} \in \mathbb{Z}_+^{k,N}} \left([a_{\mathbf{j}}]_d \cdot [b_{\mathbf{j}}]_d \right)$.

14 M_d sends s_d to Bob.

15 Bob uses any $2t-1$ shares out of $\{s_1, \ldots, S_D\}$ to reconstruct $\mathbf{a} \cdot \mathbf{b} = f(\boldsymbol{\alpha})$.

Output: Bob gets $f(\boldsymbol{\alpha})$.

Protocol 8 starts with Alice and Bob setting their inputs vectors $\mathbf{a}$ and $\mathbf{b}$ in accord with the ordering convention (Lines 1-2). Then they distribute to the mediators (t, D)-shares in them (Lines 3-5). Here too we observe that the

first entry in $\mathbf{b}$, i.e. $\mathbf{b_j}$ for $\mathbf{j} = (0, \ldots, 0)$, equals 1 (see Eq. (7)). Therefore, the mediators can set on their own corresponding shares. Hence, in Line 5 for $\mathbf{j} = (0, \ldots, 0)$ Bob does not generate and distribute shares; instead, each mediator M_d, $d \in [D]$, sets $[b_{\mathbf{j}}]_d = 1$.

After completing the distribution of shares, the mediators perform the relevant DVV sub-protocol in order to validate that the secret input vector $\mathbf{b}$ is of the form as in Eq. (7) (Lines 6-11). It is executed similarly to the validation in Protocol 7. To that end we state the following lemma, which we prove below.

Lemma 10. *The vector* $\mathbf{b} = (b_{\mathbf{j}} : \mathbf{j} \in \mathbb{Z}_+^{k,N})$*, where* $\mathbf{b_j} = 1$ *for* $\mathbf{j} = (0, \ldots, 0)$*, is of the form as in Eq. (7) if and only if* $\omega = 0$ *in all stages of the validation loop in Lines 6-11 of Protocol 8.*

In the final stage of Protocol 8, the mediators compute $(2t-1, D)$-shares in the scalar product and send them to Bob (Lines 12-14) who uses them in order to recover the scalar product (Line 15).

Proof of Lemma 10. Assume that $\mathbf{b}$ is as in Eq. (7). Then for every multi-index $\mathbf{j} \in \mathbb{Z}_+^{k,N}$, the corresponding entry in $\mathbf{b}$ is $b_{\mathbf{j}} := \boldsymbol{\alpha}^{\mathbf{j}}$. Hence, for any $2 \leq n \leq N$ and for any $\mathbf{j} \in T_n$, there exists at least one monomial $\mathbf{h} \in T_{n-1}$ such that $\mathbf{j} = \mathbf{h} + \mathbf{e}_i$, for some $1 \leq i \leq k$. Let us compare the monomial $b_{\mathbf{j}} := \boldsymbol{\alpha}^{\mathbf{j}}$ with the monomial $b_{\mathbf{h}} := \boldsymbol{\alpha}^{\mathbf{h}}$. The two multi-indices $\mathbf{j}$ and $\mathbf{h}$ equal in all entries except for the ith entry, where $j_i = h_i + 1$. Therefore,

$$b_{\mathbf{j}} := \boldsymbol{\alpha}^{\mathbf{j}} = \boldsymbol{\alpha}^{\mathbf{h}} \cdot \alpha_i = b_{\mathbf{h}} \cdot b_{\mathbf{e}_i} \,.$$

Hence, such a vector will pass all stages of the DVV in Lines 6-11.

Assume next that $\mathbf{b}$ does not comply with the form as in Eq. (7). That means that

$$\mathbf{b} = (1, \alpha_1, \ldots, \alpha_k, b_{\mathbf{j}} : 2 \leq |\mathbf{j}| \leq N) \,,$$

where there exists at least one entry $b_{\mathbf{j}}$, where $2 \leq |\mathbf{j}| \leq N$, that is not of the form as in Eq. (7). Let us focus on the first multi-index $\mathbf{j}$ that is not of that form. Namely, $\mathbf{j}$ is the first multi-index for which

$$b_{\mathbf{j}} \neq \boldsymbol{\alpha}^{\mathbf{j}} \,, \tag{8}$$

where $\boldsymbol{\alpha} = (\alpha_1, \ldots, \alpha_k)$. Assume that $|\mathbf{j}| = n \in [2, N]$ and let i be any index between 1 and k such that $\mathbf{j} = \mathbf{h} + \mathbf{e}_i$ for some $\mathbf{h} \in T_{n-1}$. By the minimality of $\mathbf{j}$ it means that

$$b_{\mathbf{h}} = \boldsymbol{\alpha}^{\mathbf{h}} \,. \tag{9}$$

From Eqs. (8) and (9) it follows that the validation check in Lines 9+10 would fail for those multi-indices. That completes the proof. □

Example. Let us illustrate the validation process in the case where $k = 2$ (two-variate polynomials) and $N = 2$ (degree two). Bob is expected to submit here vectors of the form

$$\mathbf{b} = (b_{(0,0)}, b_{(1,0)}, b_{(0,1)}, b_{(2,0)}, b_{(1,1)}, b_{(0,2)}) = (1, \alpha_1, \alpha_2, \alpha_1^2, \alpha_1\alpha_2, \alpha_2^2) \,.$$

Since the first entry is always 1, and the next two entries can be anything, validation is applied only on the last three entries — $b_{(2,0)}$, $b_{(1,1)}$, and $b_{(0,2)}$:

- To validate $b_{(2,0)}$, we observe that there is only one way to represent the multi-index $\mathbf{j} = (2, 0)$ as a sum $\mathbf{h} + \mathbf{e}_i$, namely, $(2, 0) = (1, 0) + (1, 0)$. Hence, the DVV sub-protocol checks whether $b_{(2,0)} = b_{(1,0)} \cdot b_{(1,0)}$. Therefore, validation of this entry succeeds if and only if $b_{(2,0)} = \alpha_1^2$.
- Similarly, $b_{(0,2)}$ is validated if and only if $b_{(0,2)} = \alpha_2^2$.
- To validate $b_{(1,1)}$, we observe that $\mathbf{j} = (1, 1) = \mathbf{h} + \mathbf{e}_i$ with $\mathbf{h} = (1, 0)$ and $\mathbf{e}_i = (0, 1)$ or with $\mathbf{h} = (0, 1)$ and $\mathbf{e}_i = (1, 0)$. In either case, the DVV sub-protocol checks whether $b_{(1,1)} = b_{(1,0)} \cdot b_{(0,1)} = \alpha_1 \cdot \alpha_2$.

As another example, let us consider the case $k = 3$ and $N = 8$. In that case the two vectors $\mathbf{a}$ and $\mathbf{b}$ are of dimension $\binom{8+3}{3} = 165$. To validate $b_{\mathbf{j}}$ with $\mathbf{j} = (3, 1, 4)$ we observe that $\mathbf{j}$ can be expressed as $\mathbf{j} = \mathbf{h} + \mathbf{e}_i$ in three ways: $\mathbf{h} = (2, 1, 4)$ and $\mathbf{e}_i + (1, 0, 0)$, or $\mathbf{h} = (3, 0, 4)$ and $\mathbf{e}_i + (0, 1, 0)$, or $\mathbf{h} = (3, 1, 3)$ and $\mathbf{e}_i + (0, 0, 1)$. The DVV sub-protocol chooses arbitrarily one of those sums and then it checks whether $b_{\mathbf{j}} = b_{\mathbf{h}} \cdot b_{\mathbf{e}_i}$.

6 Communication complexity

Here we discuss the communication complexity of our protocols. We measure the complexity by counting field ($\mathbb{Z}_p$) elements, where each field element can be represented by $\lceil \log p \rceil$ bits,

We separate the overall communication traffic to three parts:

- $\mathrm{Com}_{\mathrm{AM}}$: Messages sent between Alice and the mediators.
- $\mathrm{Com}_{\mathrm{BM}}$: Messages sent between Bob and the mediators.
- $\mathrm{Com}_{\mathrm{MM}}$: Messages sent among the mediators.

For Protocol 1 for the DSP problem we have $\mathrm{Com}_{\mathrm{AM}} = \mathrm{Com}_{\mathrm{BM}} = (N+1)D$, since Alice and Bob send to each of the D mediators shares in each of the N entries in their vectors and, at the end, each mediator sends a single share back to Alice and Bob. As in this protocol the mediators do not communicate among themselves, we have $\mathrm{Com}_{\mathrm{MM}} = 0$.

The communication costs of Protocol 2 for the DOT_1^N problem are as follows: $\mathrm{Com}_{\mathrm{AM}} = ND$ (Lines 3-4), $\mathrm{Com}_{\mathrm{BM}} = (N+1)D$ (Lines 3+5 and Lines 16+18). As for the communication between the mediators, it is executed in the DVV sub-protocol. We have here $ND(D-1)$ due to the first part in the validation (Lines 6-10) and $D(D-1)$ due to the second part (Lines 11-15); hence, in total we have $\mathrm{Com}_{\mathrm{MM}} = (N+1)D(D-1)$.

We move on to Protocol 3 for the DOT_k^N problem. Its communication costs are as in Protocol 2 with one difference: at the end, Bob receives from each mediator N field elements and not just one. Hence, the costs of this protocol are:

$$\mathrm{Com}_{\mathrm{AM}} = ND, \quad \mathrm{Com}_{\mathrm{BM}} = 2ND, \quad \mathrm{Com}_{\mathrm{MM}} = (N+1)D(D-1)\,. \tag{10}$$

We note that the DOT_k^N problem could also be solved by invoking Protocol 2 k times, where Alice's part has to be executed just once. The communication costs of this alternative course of action are:

$$\mathrm{Com}_{\mathrm{AM}} = ND, \quad \mathrm{Com}_{\mathrm{BM}} = k(N+1)D, \quad \mathrm{Com}_{\mathrm{MM}} = k(N+1)D(D-1)\,. \tag{11}$$

Comparing Eq. (11) to Eq. (10) we see that such an alternative course of action is less efficient than Protocol 3 for every $k \geq 2$.

Next, we consider Protocol 4 for the problem of priced OT. Its communication costs are exactly as those of Protocol 3, with the exception of the additional Lines 1-7. We assume that the weights are publish knowledge, so we do not include them in our costs. Alice has to send D shares in T and so does Bob, so that adds the term D to both $\mathrm{Com}_{\mathrm{AM}}$ and $\mathrm{Com}_{\mathrm{BM}}$. The computation in Line 6 adds $D(D-1)$ to $\mathrm{Com}_{\mathrm{MM}}$. Hence, we end up with the following costs:

$$\mathrm{Com}_{\mathrm{AM}} = (N+1)D, \quad \mathrm{Com}_{\mathrm{BM}} = (2N+1)D, \quad \mathrm{Com}_{\mathrm{MM}} = (N+2)D(D-1)\,.$$

Next, we consider the general OT protocol, Protocol 6. In the heart of that protocol (Line 10), the parties execute OT_k^N (Protocol 3) over messages $\{m''_1, \ldots, m''_N\}$ in $\mathbb{Z}_q$. Since $q \approx p^2$, the communication costs of this invocation of Protocol 3 are twice those in Eq. (10). In addition, Alice sends to the mediators $(N+1)D$ shares in $\mathbb{Z}_p$ (Lines 7-9), Bob sends to the mediators D shares (Line 15) and receives from them $tN \leq DN$ shares (Line 19), and the mediators send among themselves $D(D-1)$ shares (Line 17). Adding up everything yields the following costs:

$$\mathrm{Com}_{\mathrm{AM}} = (3N+1)D, \quad \mathrm{Com}_{\mathrm{BM}} = (5N+1)D, \quad \mathrm{Com}_{\mathrm{MM}} = (2N+3)D(D-1)\,.$$

We proceed with Protocol 7 for the OPE problem. In Lines 3-5 both Alice and Bob send to the mediators $(N+1)D$ shares. Then, in the DVV sub-protocol, the mediators send among themselves $D(D-1)$ field elements $N-1$ times. Finally, the mediators send to Bob D field elements (Lines 10-12). The overall communication costs are therefore

$$\mathrm{Com}_{\mathrm{AM}} = (N+1)D, \quad \mathrm{Com}_{\mathrm{BM}} = (N+2)D, \quad \mathrm{Com}_{\mathrm{MM}} = (N-1)D(D-1)\,.$$

Protocol 8 is analyzed similarly, with two differences. Here, the dimension of the vectors is $\binom{N+k}{k}$, and the number of entries in Bob's vector that need to be verified in the DVV sub-protocol is $\binom{N+k}{k} - k - 1$. Hence, we arrive at the following costs,

$$\mathrm{Com}_{\mathrm{AM}} = N_k D, \quad \mathrm{Com}_{\mathrm{BM}} = (N_k+1)\,D, \quad \mathrm{Com}_{\mathrm{MM}} = (N_k-k-1)\,D(D-1)\,,$$

where $N_k := \binom{N+k}{k}$.

The communication costs of all protocols are summarized in Table 1.

Problem	Protocol	Com_{AM}	Com_{BM}	Com_{MM}
SP	1	$(N+1)D$	$(N+1)D$	0
OT_1^N	2	ND	$(N+1)D$	$(N+1)D(D-1)$
OT_k^N	3	ND	$2ND$	$(N+1)D(D-1)$
Priced OT	4 & 5	$(N+1)D$	$(2N+1)D$	$(N+2)D(D-1)$
General OT	6	$(3N+1)D$	$(5N+1)D$	$(2N+3)D(D-1)$
OPE	7	$(N+1)D$	$(N+2)D$	$(N-1)D(D-1)$
OMPE	8	N_kD	$(N_k+1)D$	$(N_k-k-1)D(D-1)$

Table 1: Communication costs of all distributed protocols with D mediators. N denotes the dimesion of the vectors in SP, the number of messages in all OT protocols, and the degree of the polynomials in the OPE and OMPE protocols. The parameter k in Protocol 8 denotes the number of variables, while $N_k = \binom{N+k}{k}$.

7 Experiments

7.1 Implementation details

We implemented our protocols in Java on a Lenovo Ideapad Gaming 3 laptop, powered by an AMD Ryzen 7 5800H processor and 16GB of RAM. The operating system was Windows 11 64-bit, and the environment was Eclipse-Workspace. A 64-bit prime number p was chosen at random for the size of the underlying field $\mathbb{Z}_p$. To enable computations modulo such prime, we used the BigInteger Java class.

All experiments were conducted on randomly generated vectors (or sets of messages or polynomials). Each experiment was repeated ten times and the average runtimes are reported. The standard deviation is omitted from the graphical display of our results since it is barely noticeable.

7.2 Results

In the first experiment we tested our basic protocol that solves DSP, Protocol 1. Figure 1 shows the runtimes for Alice and Bob and the average runtimes for the mediators as a function of N (the dimension of the two vectors). Those runtimes grow linearly in N. Figure 2 displays those runtimes as a

function of D. As expected, the runtime of Alice and Bob grows linearly with D (as they need to compute $O(D)$ shares in their inputs) while the mediators' runtime is not affected by D and only slightly fluctuates randomly between 50 and 85 milliseconds for all tested values of D.

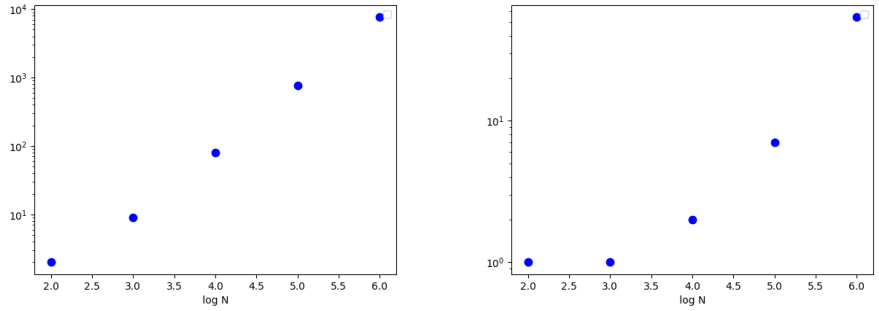


Figure 1: Runtimes (milliseconds) for Protocol 1 (DSP), as a function of $\log_{10}(N)$, for $D = 7$. The left plot shows the runtimes for Alice and Bob; the right plot shows the average runtimes for the mediators. The runtimes are presented on a logarithmic scale.

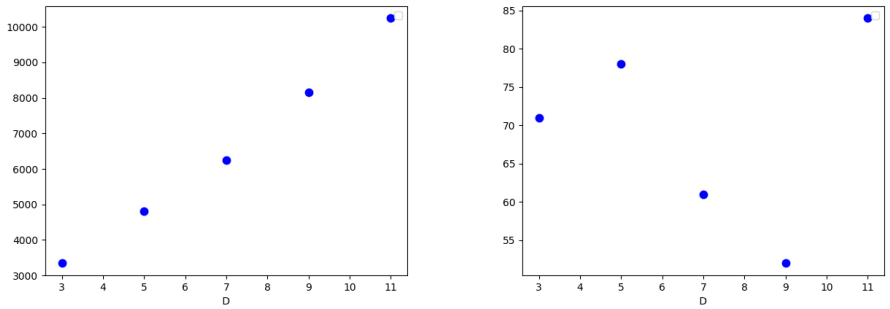


Figure 2: Runtimes (milliseconds) for Protocol 1 (DSP), as a function of D, for $N = 10^6$. The left plot shows the runtimes for Alice and Bob; the right plot shows the average runtimes for the mediators. The runtimes are presented on a linear scale.

In the next experiment we tested Protocol 3 that solves the OT_k^N problem. Here we focus only on the mediators, since Bob's computations in that

protocol are the same as in Protocol 1, while Alice's computations are the same as in the beginning of Protocol 1. (Recall that in Protocol 3 the output of the scalar product goes only to Bob, so Alice has nothing to do beyond the initial sharing of her vector.) Hence, Bob's runtimes in Protocol 3 are just as they were in Protocol 1, for the same setting of N and D (the setting of k has no effect on Bob's runtime), while Alice's runtimes in Protocol 3 are smaller than her runtimes in Protocol 1 for the same setting of N and D.

Turning our attention to the mediators' runtimes, their average (over all runs of the protocol and over all mediators) are shown in Figure 3. The dependence on N is linear. As for D, while in Protocol 1 the mediators' runtimes do not depend on D, here they do depend on D, linearly, due to the DVV part of the protocol. Finally, their runtime is not affected by k, as can be seen in the last plot in Figure 3.

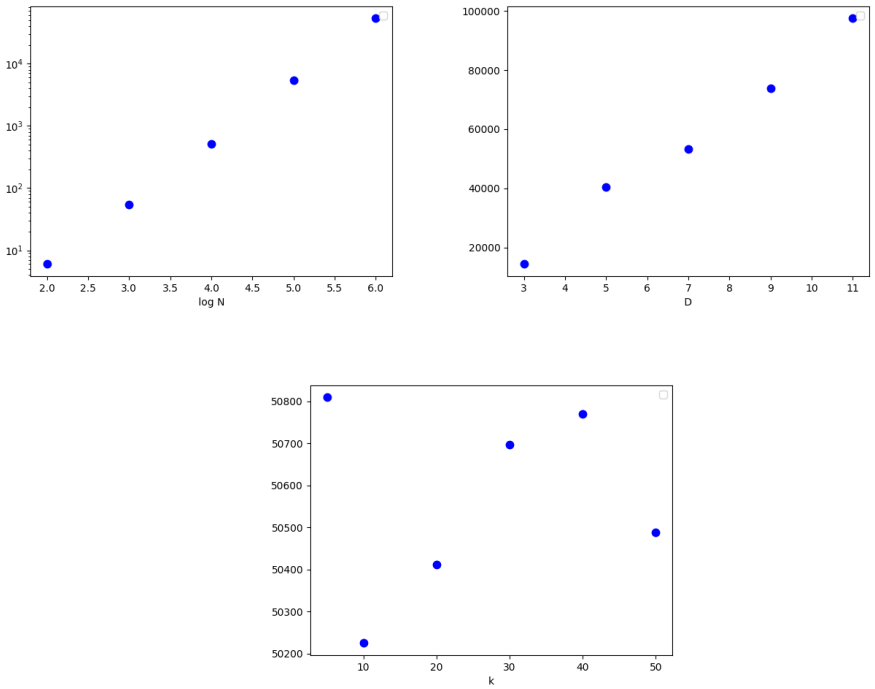


Figure 3: Average runtimes (milliseconds) for the mediators in Protocol 3 (OT_k^N). Top left: runtimes, on a logarithmic scale, as a function of $\log_{10}(N)$, for $D = 7$ and $k = 10$. Top right: runtimes as a function of D, for $N = 1000000$ and $k = 10$. Bottom: runtimes as a function of k, for $N = 1000000$ and $D = 7$.

Next, we consider the OMPE protocol — Protocol 8. We ran that protocol with random polynomials of degrees $N \in \{5, 10, 20, 30, 40, 50\}$, where the number of variables was $k = 1, 2, 3$ — see Figure 4. Note that when we set $k = 1$ in Protocol 8 it coincides with Protocol 7. The shown runtimes grow linearly with $\binom{N+k}{k}$, since that is the size of the two vectors in the underlying scalar product.

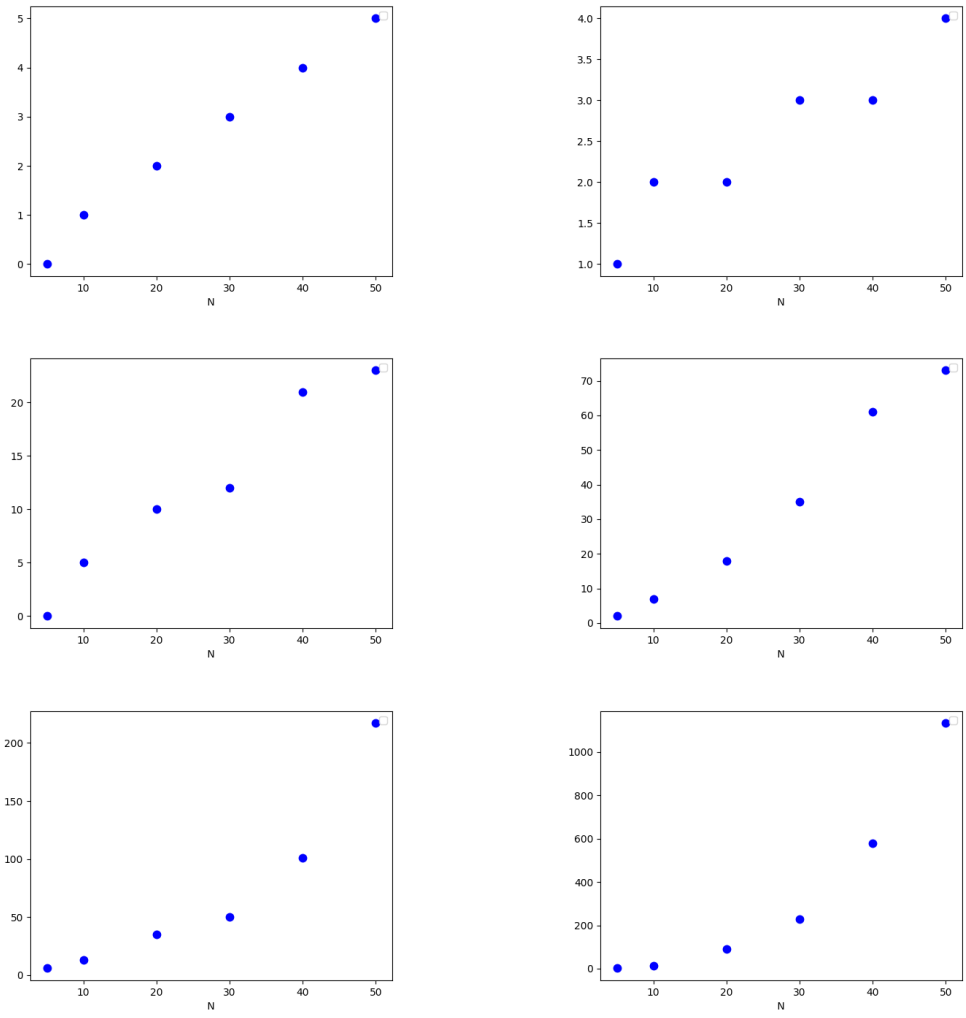


Figure 4: Runtimes (milliseconds) for Protocol 8 (OMPE)), as a function of N, the polynomial degree, for $k = 1$ (top), $k = 2$ (middle), and $k = 3$ (bottom). The left plots show the runtimes for Bob; the right plots show the average runtimes for the mediators.

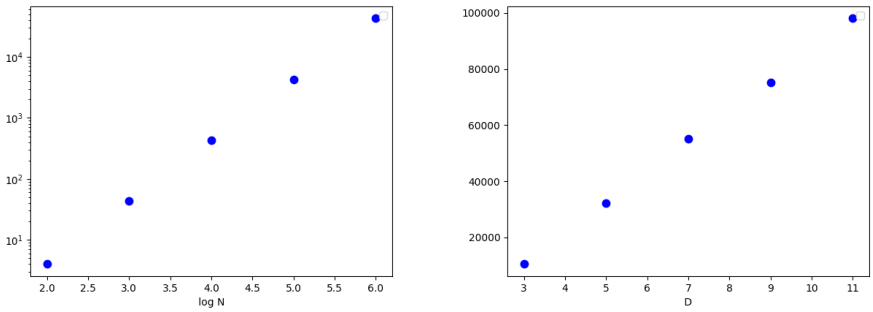


Figure 5: Average runtimes (milliseconds) for the mediators in Protocol 4 (POT)). Left: runtimes as a function of N, for $T = 100$ and $D = 7$; the runtimes are presented on a logarithmic scale. Right: runtimes as a function of D, for $T = 100$ and $N = 1000000$.

We turn our attention to the POT protocol — Protocol 4. Like in Protocol 3, we ignore the runtimes of Alice and Bob and focus on the mediators' average runtime and demonstrate its linear dependence on N and on D, see Figure 5.

Finally, we tested the GOT protocol, Protocol 6, with the access structure that we described in Section 4.3.1. In all of our experiments we used compartments of equal size, $|U_i| = 10$, $1 \leq i \leq r$. The runtimes for Bob and the mediators, as a function of N and D, are reported in Figures 6 and 7.

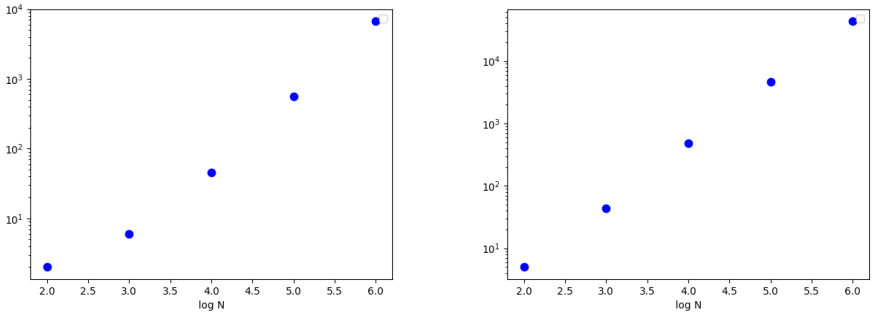


Figure 6: Runtimes (milliseconds) for Protocol 6 (GOT)), in the case of compartmented access structures as a function of N, for $D = 7$. The left plot shows the runtimes for Bob; the right plot shows the average runtimes for the mediators. The runtimes are presented on a logarithm scale.

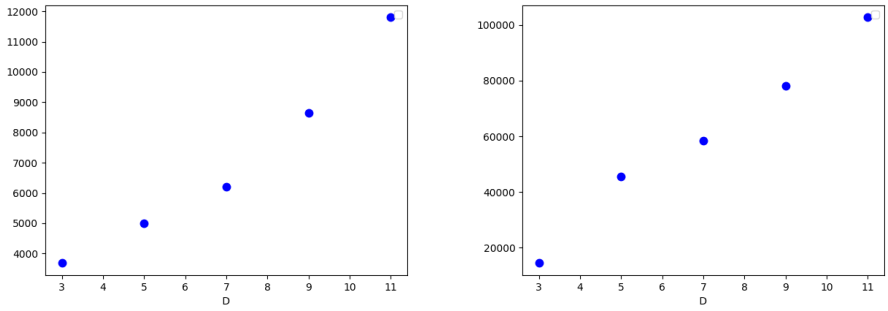


Figure 7: Runtimes (milliseconds) for Protocol 6 (GOT)), in the case of compartmented access structures as a function of D, for $N = 1000000$. The left plot shows the runtimes for Bob; the right plot shows the average runtimes for the mediators. The runtimes are presented on a linear scale.

References

[1] Joël Alwen, Jonathan Katz, Yehuda Lindell, Giuseppe Persiano, Abhi Shelat, and Ivan Visconti. Collusion-free multiparty computation in the mediated model. In *CRYPTO*, pages 524–540, 2009.

[2] Joël Alwen, Abhi Shelat, and Ivan Visconti. Collusion-free protocols in the mediated model. In *CRYPTO*, pages 497–514, 2008.

[3] Ernest F. Brickell. Some ideal secret sharing schemes. In *EUROCRYPT*, pages 468–475, 1989.

[4] O. Catrina and F. Kerschbaum. Fostering the uptake of secure multiparty computation in e-commerce. In *ARES*, pages 693–700, 2008.

[5] Benny Chor, Eyal Kushilevitz, Oded Goldreich, and Madhu Sudan. Private information retrieval. *J. ACM*, 45(6):965–981, 1998.

[6] I. Damgård and Y. Ishai. Constant-round multiparty computation using a black-box pseudorandom generator. In *CRYPTO*, volume 3621, pages 378–394, 2005.

[7] Ivan Damgård and Jesper Buus Nielsen. Scalable and unconditionally secure multiparty computation. In *CRYPTO*, pages 572–590, 2007.

[8] L. Dery, T. Tassa, and A. Yanai. Fear not, vote truthfully: Secure multiparty computation of score based rules. *Expert Systems with Applications*, 168:114434, 2021.

[9] L. Dery, T. Tassa, A. Yanai, and A. Zamarin. Demo: A secure voting system for score based elections. In *ACM CCS*. To appear. Manuscript can be requested from ISF authorities, 2021.

[10] A. Ben Horin and T. Tassa. Privacy preserving collaborative filtering by distributed mediation. In *RecSys*, pages 332–341. ACM, 2021.

[11] Yuval Ishai and Eyal Kushilevitz. Private simultaneous messages protocols with applications. In *ISTCS*, pages 174–184, 1997.

[12] S. Kamara, P. Mohassel, and M. Raykova. Outsourcing multi-party computation. *IACR Cryptology ePrint Archive*, 2011. 272.

[13] Joe Kilian. Founding cryptography on oblivious transfer. pages 20–31. ACM, 1988.

[14] Moni Naor and Benny Pinkas. Oblivious transfer and polynomial evaluation. In *STOC*, pages 245–254, 1999.

[15] Michael O. Rabin. How to exchange secrets by oblivious transfer. Technical Report TR-81, Aiken Computation Laboratory, Harvard University, 1981.

[16] Johannes Schneider. Lean and fast secure multi-party computation: Minimizing communication and local computation using a helper. In *SECRYPT*, pages 223–230, 2016.

[17] Adi Shamir. How to share a secret. *Commun. ACM*, 22(11):612–613, 1979.

[18] E. Shmueli and T. Tassa. Mediated secure multi-party protocols for collaborative filtering. *ACM Transactions on Intelligent Systems and Technology*, 11:1–25, 2020.

[19] Tamir Tassa. Generalized oblivious transfer by secret sharing. *Des. Codes Cryptogr.*, 58(1):11–21, 2011.

[20] Tamir Tassa and Nira Dyn. Multipartite secret sharing by bivariate interpolation. In *ICALP*, pages 288–299, 2006.

[21] Tamir Tassa, Tal Grinshpoun, and Avishay Yanai. PC-SyncBB: A privacy preserving collusion secure DCOP algorithm. *Artificial Intelligence*, 297:103501, 2021.

[22] Tamir Tassa, Ayman Jarrous, and Yonatan Ben-Ya'akov. Oblivious evaluation of multivariate polynomials. *J. Math. Cryptol.*, 7(1):1–29, 2013.

[23] A.C. Yao. Protocols for secure computation. In *FOCS*, pages 160–164, 1982.

Graphical Learning Algorithm to Interference-Aware Routing in Communication Networks *

Raz Paul[1], Kobi Cohen[1], and Gil Kedar[2]

[1] Ben-Gurion University of the Negev, Beer Sheva 8410501 Israel
razpa@post.bgu.ac.il, yakovsec@bgu.ac.il
[2] Ceragon Networks Ltd., Tel Aviv, Israel.
gilke@ceragon.com

Abstract. Wireless interference between flows in wireless networks result in degraded performances when data signals from different flows can mutually interfere with each other along their routes. Our goal is to develop a routing algorithm in wireless interference networks to maximize the network utility. However, achieving an optimal solution is computationally demanding due to the extensive state and action spaces involved. To address this challenge, we propose a Dual-stage Interference-Aware Multi-flow Optimization of Network Data-signals (DIAMOND) algorithm. DIAMOND is designed to support a hybrid centralized-distributed implementation, which aligns with the characteristics of 5G and beyond technologies deploying centralized units. In the centralized stage, a novel graph neural network (GNN) reinforcement learning (RL) routing agent computes the multi-flow transmission strategy. Subsequently, in the distributed stage, performance enhancements are achieved through innovative distributed learning updates. Theoretical analysis proves that DIAMOND converges to the optimal routing strategy as time increases. Simulation results demonstrate the superior performance of DIAMOND compared to existing methods.

Keywords: Wireless Networks · distributed learning · deep reinforcement learning (DRL) · graph neural network (GNN).

1 Introduction

The rapid advancement of communication network technology in 5G and beyond has been accompanied by a growing demand for wireless communication services. However, one of the main challenges that persists is the scarcity of available spectrum, which hampers the ability to meet this increasing demand. As a result, developing efficient algorithms for data transmission in wireless networks becomes crucial in effectively utilizing the limited spectral resources.

In recent years, significant improvements have been made by developing machine learning algorithms for managing flow transmissions in order to tackle uncertainties in

* A full version of this paper was submitted to *IEEE Transactions on Wireless Communications* and is available as a pre-print at [1].

This work was supported by the Israel Ministry of Economy under the Magnet consortium program.

random channel and network conditions. Several studies [2–12] have analyzed the long-term reward optimization of users in the network using a multi-armed bandit learning framework. The learning strategies have included various methods, such as reinforcement learning and upper confidence bound (UCB)-based algorithms [13–15], as well as deep reinforcement learning that uses deep neural networks in the optimization [16–19]. While most of these online learning methods have focused on single-hop transmissions, in this paper we apply the learning to multi-hop link states to enable efficient path selection for flow transmission.

Our objective is to devise a multi-flow transmission strategy that effectively routes data flows across wireless interference networks, all while maximizing a specific network utility metric. Unlike deterministic (e.g., [20, 21]) or stochastic random (e.g., [22–26]) link weight approaches that are commonly used in certain scenarios, these methods do not hold validity in wireless interference networks. This is primarily due to the interference caused by data signals from different flows, which impact the link capacities along their respective routes. Consequently, finding an optimal solution to this problem often involves computationally expensive tasks due to the large state and action spaces associated with wireless interference networks.

2 Problem Statement

We represent the wireless communication network using a directed connected graph $G = (\mathcal{V}, \mathcal{E})$, where $\mathcal{V}$ denotes the set of nodes (i.e., users) in the network, and $\mathcal{E}$ represents the set of edges. Given N flows, we define the selected route vector for all flows as $\sigma \triangleq (\sigma_1, \sigma_2, ..., \sigma_N)$, and the set of all allowed routes for flow n as $\mathcal{A}_n = \{\varphi_1^n, \varphi_2^n, \ldots, \varphi_{K_n}^n\}$. Additionally, we consider a bounded utility function $u_n(\sigma)$ for flow n, such as the achievable rate or a monotonically increasing function based on the achievable rate. It is important to note that the utility of flow n depends not only on its selected route but also on the selected routes of other flows that may interfere with it.

The objective is to find a multi-flow transmission path vector σ that solves the network utility maximization (NUM) problem [21] defined as:

$$\sigma^* = \underset{\{\sigma_n \in \mathcal{A}_n\}_{n=1}^N}{\arg\max} \sum_{n=1}^{N} u_n(\sigma). \tag{1}$$

In wireless networks, the interference between data signals from different flows affects the link capacities, resulting in reduced achievable rates. The achievable rate of user n is influenced by the bandwidth of the bottleneck link B_ℓ and the signal-to-interference-plus-noise ratio (SINR) at the receiver of link ℓ. Further details can be found in [1].

The combinatorial optimization problem of finding the optimal multi-flow transmission path vector to solve (1) is known to be NP-hard [27]. The exponential number of possible route allocations makes it impractical to solve optimally using brute-force methods. Therefore, there is a need for computationally-efficient algorithms. Additionally, even for small networks, computing the optimal allocation is often costly. Hence, traditional learning-based approaches using supervised models are not feasible. In the next section, we present a novel algorithm that overcomes these limitations to solve (1).

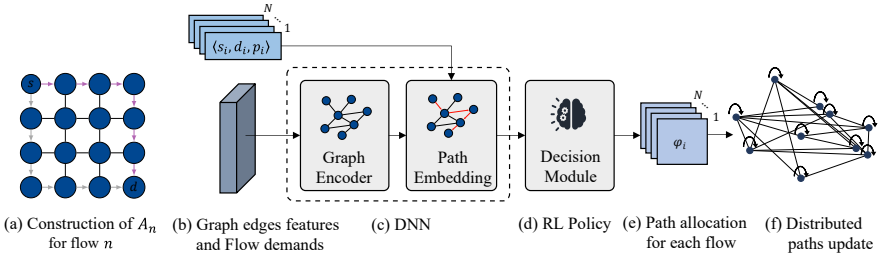


Fig. 1: An overview of the proposed DIAMOND framework: (a) A construction of the search space for flow n for $n = 1, 2, ..., N$. (b)-(e) *Centralized GRRL module:* (b) The RL agent receives N flow demands, and the network state as link features. (c) The network state is processed with a graph encoder GNN to produce an embedding for each link as well as a global graph embedding. The flow demands and embedding are processed by the path-embedding module that outputs an embedding for each of the possible routing options. (d) The RL agent makes an allocation decision based on the path embedding. (e) The transmission paths of all N flows are allocated at once to the network. (f) *Distributed NB3R module:* Each flow updates its path allocation distributedly, based on the NB3R policy, which refines the allocation.

3 The Proposed DIAMOND Algorithm

We introduce a novel algorithm, dubbed Dual-stage Interference-Aware Multi-flow Optimization of Network Data-signals (DIAMOND), which addresses the challenges of multi-flow optimization in wireless networks. DIAMOND is designed to support a hybrid centralized-distributed implementation, aligning with the centralized unit deployments characteristic of 5G and beyond technologies. The centralized stage of DIAMOND utilizes a unique module called GRRL (Graph neural network Routing agent via Reinforcement Learning) to compute the multi-flow transmission strategy. This module, implemented on a centralized unit similar to OSPF [28], given the interference map and flow demands, as given in 5G networks. By optimizing the network utility, GRRL generates a path allocation vector, assigning a single route per flow. It employs a reinforcement learning (RL) approach, training a policy network based on a novel Graph Neural Network (GNN) architecture. This captures the ability of GNN to efficiently search the large search space and approximate well the optimal solution. It is designed in a generic way that handles general parameter values, number of nodes, edges, and flows. To avoid local maxima while dynamically update strategies, a distributed stage called NB3R (Noisy Best-Response for Route Refinement) is incorporated. NB3R introduces a novel module design where each source node asynchronously updates its path in a probabilistic manner, aiming to approach the global solution of the Network Utility Maximization (NUM) problem stated in Equation (1). An illustration of DIAMOND's modules can be found in Figure 1. Detailed implementation of DIAMOND, and theoretical convergence analysis to the optimal solution are provided in [1].

Table 1: Algorithm comparison for various network configurations.

			Average Flow Rates [Mbps] ↑				Max Delay [time-steps] ↓			
N	V	E	RB	OSPF	DQN+GNN	**DIAMOND (ours)**	RB	OSPF	DQN+GNN	**DIAMOND (ours)**
100	200	300	0.611	2.061	9.937	**17.239**	314.2	372.1	339.9	**113.8**
70	70	140	1.887	4.201	20.597	**30.594**	179.2	147.7	155.5	**60.7**
30	10	45	18.553	20.21	52.57	**77.689**	18.8	24.2	26.1	**10.8**

4 Results

We present numerical examples to demonstrate the performance of the proposed DIAMOND algorithm across diverse wireless interference network environments. These examples encompass different numbers of flow demands (N), nodes (V), and edges (E).

In our simulations, all V network nodes are deployed randomly within a $1000m \times 1000m$ area. A random topology is generated by establishing E links, ensuring the graph forms a single connected component. Flow demands (source, destination, and packet load), link's capacities and gains are randomly assigned.

The GRRL agent is initially trained on a small-scale problem with parameters $(N = 20, V = 10, E = 20)$ and subsequently fine-tuned on a larger-scale setting with parameters $(N = 30, V = 20, E = 30)$. To highlight the generalization capabilities of the GNN and the adaptive NB3R algorithm, all testing results are obtained from different problem settings. Through these numerical examples, we aim to showcase the versatility and effectiveness of the GNN-based DIAMOND approach.

We compared the following algorithms: (i) *Random Baseline (RB)*: The random baseline is a heuristic method that selects the best path allocation from 100 independent trials of randomly chosen actions. (ii) *Open Shortest Path First (OSPF)* [28]: The popular OSPF protocol that employs the Dijkstra algorithm to route data through the shortest path. (iii) DQN+GNN [29]: The recently proposed DQN+GNN algorithm that utilizes a DRL algorithm based on the well-known offline-RL DQN [30] algorithm. For each simulation experiment, we averaged the results over 10 independent random settings, including flow demands and link capacities.

Table 1 presents the results for lightly-loaded networks with $V = 2N$ and $E = 1.5V$ (line 1), moderately-loaded networks with $V = N$ and $E = 2V$ (line 2), and a scenario with a higher ratio of nodes to edges (line 3). More extensive simulation results are provided in [1]. The results demonstrate that DIAMOND outperforms all other algorithms in both average rate and packet delay across all scenarios, which represent different realistic network configurations. These findings justify the application of DIAMOND in 5G topologies, characterized by dense networks with a large number of edges.

References

1. Paul, R., Cohen, K., Kedar, G.: Multi-flow transmission in wireless interference networks: A convergent graph learning approach (2023)
2. Tekin, C., Liu, M.: Online learning in opportunistic spectrum access: A restless bandit approach. In: IEEE International Conference on Computer Communications (INFOCOM). pp. 2462–2470 (2011)
3. Tekin, C., Liu, M.: Online learning of rested and restless bandits. IEEE Transactions on Information Theory **58**(8), 5588–5611 (2012)
4. Liu, H., Liu, K., Zhao, Q.: Learning in a changing world: Restless multiarmed bandit with unknown dynamics. IEEE Transactions on Information Theory **59**(3), 1902–1916 (2012)
5. Cohen, K., Zhao, Q., Scaglione, A.: Restless multi-armed bandits under time-varying activation constraints for dynamic spectrum access. In: 48th Asilomar Conference on Signals, Systems and Computers. pp. 1575–1578. IEEE (2014)
6. Gafni, T., Cohen, K.: Learning in restless multi-armed bandits using adaptive arm sequencing rules. In: IEEE International Symposium on Information Theory (ISIT). pp. 1206–1210 (2018)
7. Bistritz, I., Leshem, A.: Distributed multi-player bandits-a game of thrones approach. In: Advances in Neural Information Processing Systems. pp. 7222–7232 (2018)
8. Turğay, E., Bulucu, C., Tekin, C.: Exploiting relevance for online decision-making in high-dimensions. IEEE Transactions on Signal Processing **69**, 1438–1451 (2020)
9. Yemini, M., Leshem, A., Somekh-Baruch, A.: Restless hidden Markov bandit with linear rewards. In: IEEE Conference on Decision and Control (CDC). pp. 1183–1189 (2020)
10. Gafni, T., Cohen, K.: Learning in restless multiarmed bandits via adaptive arm sequencing rules. IEEE Transactions on Automatic Control **66**(10), 5029–5036 (2020)
11. Gafni, T., Cohen, K.: Distributed learning over markovian fading channels for stable spectrum access. IEEE Access **10**, 46652–46669 (2022)
12. Gafni, T., Yemini, M., Cohen, K.: Learning in restless bandits under exogenous global markov process. IEEE Transactions on Signal Processing **70**, 5679–5693 (2022)
13. Agrawal, R.: Sample mean based index policies with $O(logn)$ regret for the multi-armed bandit problem. Advances in Applied Probability pp. 1054–1078 (1995)
14. Auer, P., Cesa-Bianchi, N., Fischer, P.: Finite-time analysis of the multiarmed bandit problem. Machine learning **47**(2-3), 235–256 (2002)
15. Tabei, G., Ito, Y., Kimura, T., Hirata, K.: Multi-armed bandit-based routing method for in-network caching. In: IEEE Asia-Pacific Signal and Information Processing Association Conference (APSIPA). pp. 1899–1902 (2021)
16. Wang, S., Liu, H., Gomes, P.H., Krishnamachari, B.: Deep reinforcement learning for dynamic multichannel access in wireless networks. IEEE Transactions on Cognitive Communications and Networking **4**(2), 257–265 (2018)
17. Yu, Y., Wang, T., Liew, S.C.: Deep-reinforcement learning multiple access for heterogeneous wireless networks. IEEE Journal on Selected Areas in Communications **37**(6), 1277–1290 (2019)
18. Naparstek, O., Cohen, K.: Deep multi-user reinforcement learning for distributed dynamic spectrum access. IEEE Transactions on Wireless Communications **18**(1), 310–323 (2019)
19. Bokobza, Y., Dabora, R., Cohen, K.: Deep reinforcement learning for simultaneous sensing and channel access in cognitive networks. IEEE Transactions on Wireless Communications (2023)
20. Ying, L., Shakkottai, S., Reddy, A., Liu, S.: On combining shortest-path and back-pressure routing over multihop wireless networks. IEEE/ACM Transactions on Networking **19**(3), 841–854 (2010)

21. Srikant, R., Ying, L.: Communication networks: an optimization, control, and stochastic networks perspective. Cambridge University Press (2013)
22. Liu, K., Zhao, Q.: Adaptive shortest-path routing under unknown and stochastically varying link states. In: 10th International Symposium on Modeling and Optimization in Mobile, Ad Hoc and Wireless Networks (WiOpt). pp. 232–237. IEEE (2012)
23. Tehrani, P., Zhao, Q.: Distributed online learning of the shortest path under unknown random edge weights. In: IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP). pp. 3138–3142 (2013)
24. Talebi, M.S., Zou, Z., Combes, R., Proutiere, A., Johansson, M.: Stochastic online shortest path routing: The value of feedback. IEEE Transactions on Automatic Control **63**(4), 915–930 (2017)
25. Huang, Z., Xu, Y., Pan, J.: Tsor: Thompson sampling-based opportunistic routing. IEEE Transactions on Wireless Communications **20**(11), 7272–7285 (2021)
26. Amar, O., Cohen, K.: An online learning approach to shortest path and backpressure routing in wireless networks. arXiv preprint arXiv:2204.03620 (2022)
27. di Ianni, M.: Efficient delay routing. Theor. Comput. Sci. **196**(1–2), 131–151 (apr 1998). https://doi.org/10.1016/S0304-3975(97)00198-9
28. Sidhu, D., Fu, T., Abdallah, S., Nair, R., Coltun, R.: Open shortest path first (ospf) routing protocol simulation. In: Conference Proceedings on Communications Architectures, Protocols and Applications. p. 53–62. SIGCOMM '93, Association for Computing Machinery, New York, NY, USA (1993). https://doi.org/10.1145/166237.166243
29. Almasan, P., Suárez-Varela, J., Badia-Sampera, A., Rusek, K., Barlet-Ros, P., Cabellos-Aparicio, A.: Deep reinforcement learning meets graph neural networks: An optical network routing use case. CoRR **abs/1910.07421** (2019)
30. Mnih, V., Kavukcuoglu, K., Silver, D., Rusu, A.A., Veness, J., Bellemare, M.G., Graves, A., Riedmiller, M., Fidjeland, A.K., Ostrovski, G., Petersen, S., Beattie, C., Sadik, A., Antonoglou, I., King, H., Kumaran, D., Wierstra, D., Legg, S., Hassabis, D.: Human-level control through deep reinforcement learning. nature **518**(7540), 529–533 (2015)

Blockchain-Enabled Car Sharing Platform: Enhancing Reliability and Vehicle History Management

Chen Ben Tolila, Yarden Hovav, Hadassa Daltrophe
Shamoon College of Engineering, Ashdod, Israel
{chenbe3,yardeho,hadasda1 }@ac.sce.ac.il

Index Terms—Blockchain, crowdsourcing, Car-sharing, Smart-contracts, POA(proof-of-authority)

The rising expenses associated with car ownership have driven individuals to seek more affordable alternatives, such as car rentals. However, conventional car rental services often come with high costs due to leasing company overhead expenses. Consequently, car sharing has emerged as a popular and cost-effective solution that not only reduces expenses but also promotes eco-friendliness by reducing the overall number of vehicles on the roads. Nonetheless, centralization and reliability remain persistent challenges in car-sharing implementation.

To address these issues, we propose a decentralized crowd car sharing and renting platform called CrowdCarLink, leveraging blockchain technology's power. This innovative platform enables both individuals and leasing companies to rent out vehicles while securely recording the maintenance history of each vehicle on the blockchain. Within CrowdCarLink, garages are pivotal contributors, adding vehicle information in a reliable and immutable manner. By utilizing blockchain technology, our platform ensures transparency and fosters trust, effectively overcoming the limitations imposed by centralization.

Our architectural design incorporates smart contracts which help streamline processes and facilitate seamless transactions within the platform.
To demonstrate the feasibility of our approach, we have developed a prototype utilizing a private Ethereum blockchain with Proof of Authority (PoA) consensus.

We believe that the architectural design and the practical solution presented here will play an integral role in shaping the future of smart transportation. By offering a cost-effective and efficient solution, our platform aims to benefit individuals and the environment alike, paving the way for a more sustainable and advanced transportation ecosystem.

Secrecy with Intent: Malware Propagation through Deep Learning-driven Steganography

Mikhail Diyachkov[1], Arkadi Yakubov[1], Hadassa Daltrophe[1], and Kiril Danilchenko[2]

[1] Shamoon College of Engineering, Israel
{mikhadi,arkadya}@ac.sce.ac.il
hadasda1@sce.ac.il
[2] University of Waterloo, Canada
kdanilch@uwaterloo.ca

Abstract. This paper explores the use of deep learning-driven steganography for propagating malware covertly. Traditional antivirus systems effectively detect and neutralize malware, but they struggle to identify codes concealed within steganographic images. We propose an innovative approach that leverages deep learning techniques to transform malware into an image and embed it within a cover image. This technique allows for the covert delivery of malware with minimal visible changes to the cover image, enabling successful evasion of antivirus checks.
In our proposed methodology, we first develop a specialized neural network model capable of transforming malware into image representations. Once the cover image with embedded malware is created, we distribute it to unsuspecting targets through various channels, such as online platforms or social engineering tactics. When the target downloads or interacts with the seemingly innocent cover image, the embedded malware is extracted and executed on their system. This covert delivery method bypasses traditional antivirus checks, allowing the malware to remain undetected and potentially carry out its malicious activities. Combining deep learning techniques, steganography, and covert delivery, our proposed methodology presents a significant challenge for traditional antivirus systems.

Keywords: cyber attack · machine Learning · adversarial strategy · hidden communication · malicious tactic

1 Introduction

The immense volume of data and resources available online has revolutionized the speed of learning and coding. Consequently, many individuals, from students to professional coders, often utilize code fragments found on the web, which can be easily implemented on their personal devices. However, it's crucial to tread carefully when executing internet-sourced code to evade any potential hazards linked to malevolent intentions. The matter intensifies when the code is surreptitiously embedded within an ostensibly harmless image. In such scenarios, the

code remains invisible, and individuals may unknowingly execute the concealed code, oblivious to its existence or potential risks.

Steganography, the craft of covertly embedding text within plain sight, can potentially conceal malicious software within an array of file types, encompassing images, audio files, or documents. The power of machine learning can be harnessed to augment the efficiency of such hidden malware, enabling the creation of more advanced and elusive strategies.

In this study, we delve into a novel and potentially transformative application of steganography: the use of a specially designed neural network model to integrate malware within images, which can subsequently be extracted and activated. This method introduces a fresh perspective in the ongoing fight for cybersecurity, carrying substantial implications for identifying and preventing digital threats.

Attack lifecycle

The lifecycle of how the attacker and its victim (the 'user') interact is illustrated in Figure 1 and described as follows. The attacker's main objective is to cleverly

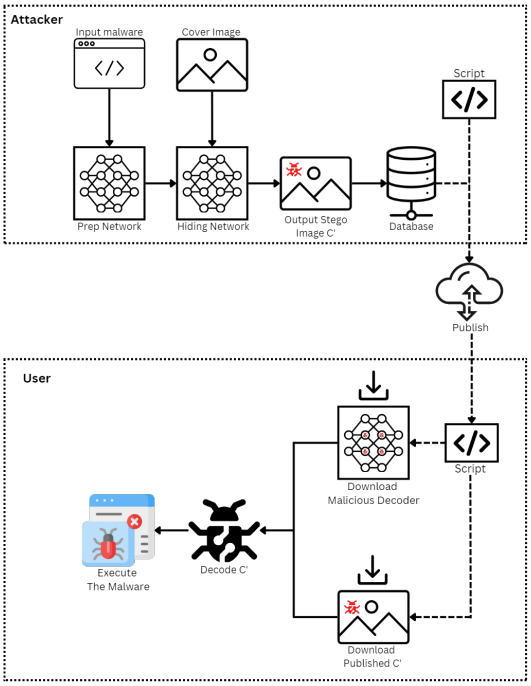


Fig. 1. Flowchart Illustrating of the attack lifecycle, which provides a comprehensive understanding of the attack process.

hide the malware within images so that the alterations are practically invisible to the naked eye. On the other side, we have the unsuspecting user. This individual or system becomes the recipient of the contaminated model, often without their knowledge. The following steps outline the attacker's scheme:

Attacker side

1. **Develop the neural network model:** This is a specific type of neural network model with a unique capability: it can seamlessly embed malware into images and later retrieve the hidden malware from these images (the exact mechanics of how this neural network operates and the principles it relies on, will be discussed in greater detail later in the paper).
2. **Embed the malicious code:** The input malware ('secret') is embedded as an image within another innocent image ('cover') using the neural network.
3. **Write an intriguing article:** The article isn't just informative; it's also persuasive. It explains the workings of this special neural network model in a way that piques curiosity and encourages readers to explore further. The goal is to make the model sound groundbreaking and innovative, sparking readers' interest to the point that they will want to download and experiment with it.
4. **Makes the model and the image publicly accessible:** After creating sufficient buzz through the article, the attacker makes this model available online. However, the distribution method is far from conventional. The attacker might employ a strategy known as 'supply chain pollution.' This crafty technique circulates harmful software by integrating it into legitimate supply chains. Using this method, the attacker can effectively disseminate the tainted model into numerous repositories or other locations.

User side

1. **Download the intriguing model:** This can happen by actively searching for and downloading described in the attacker's article or through an automatic update process. Regardless of the method, once the receiver has the model, they inadvertently extract the concealed malware. This isn't a random process, though. The malware is equipped with a set of predefined rules, which it follows to verify its integrity and successfully integrate itself into the receiver's system without raising any red flags.
2. **Execute the model locally with the malicious image:** The code of the deep learning network, the OCR, and the stego-image, are in the hand of the user. The user executes the code to study about deep learning, and it interpreted the malware code, which runs locally on the user's machine.

The open-source paradigm supports the above scheme. With the acceptance of this paper, we aim to foster further research and collaboration by providing open-source access to our implementation code.

2 Related Work

Recent years have witnessed the popularity of internet-based code-sharing platforms, notably Git [1], where users and programmers exchange code globally. According to a statistical analysis by Kinsta [2], Git has garnered an impressive user base of approximately 100 million developers worldwide, thereby highlighting its significance in global code-sharing activities. The platform attracts an estimated 14 million visitors daily and nearly 96.4 million daily page impressions, further substantiating its robust utilization in the global developer community.

As the use of code from the internet is so widespread, antivirus software is also evolving to prevent infection by various viruses. Consequently, we decided to hide the code in images using steganography and machine learning methods to circumvent antivirus protections.

Steganography, an ancient practice rooted in concealing a text within plain view, has evolved significantly over time. Its origins can be traced back to the 15th century, when the physical hiding of messages was commonplace [4]. In the present era, modern steganography focuses on discreetly conveying digital messages. Working covertly, embedding confidential information within unsuspecting cover images is characteristic of steganography and presents it as a captivating field of inquiry, particularly within the realm of cybersecurity.

Steganography methods can be grouped into three categories: traditional methods [11] that do not involve machine learning or deep learning algorithms, methods based on Convolutional Neural Networks (CNNs) [4], and methods based on Generative Adversarial Networks (GANs) [5]. One traditional method, Least Significant Bits (LSB) [6] substitution, converts secret information into binary form and then replaces the least significant bits of the cover image with the binary data. Another traditional method, Pixel Value Differencing (PVD) [13], takes the difference between consecutive pixels to determine where to hide secret bits while maintaining the consistency of the cover image. In recent years, research on steganography has benefited from developing deep learning methods, including CNNs and GANs, and their use in steganography and steganalysis.

Initially, the encoder-decoder architecture was used for data compression, with the encoder compressing the input into a smaller representation and the decoder accurately reconstructing the original input. However, this architecture is not ideal for generative models as the encoder output is not regulated. Variational Autoencoders (VAEs) presented in [9] address this regulation problem by combining the encoder and decoder modules with modifications to the penultimate layers. VAEs and Generative Adversarial Networks (GANs) have been used for data generation, such as images and text, and have also been applied in media steganography and creating fake content for deception.

In a paper by Wang et al. [12], a novel approach is introduced for stealthily transmitting malware through a neural network model. The malware is inserted into neurons, enabling its covert delivery with minimal or no effect on the neural network's performance. Additionally, the unchanged structure of the model allows it to bypass antivirus detection.

Baluja [4] uses deep neural networks to place a color image within another image of the same size. He utilized two full-colored images of resolution 64×64. However, this low resolution presents a significant challenge when the objective is to conceal an image containing code. As such, in its original form, [4] architecture does not offer an adequate solution when the goal is to hide a script code. As illustrated in Figure 2, the secret image is not effectively hidden. Following, we describe how we tackled this challenge and how we could hide the malicious code using a deep neural network based on [4] idea.

Fig. 2. Example of a stego image generated using the original model proposed by Baluja et al. [4], showcasing the limited effectiveness of the model in concealing a complex secret image at 64×64 resolution.

3 Attack lifecycle

The architecture we use in this study, grounded in the autoencoder methodology, comprises three key networks: *a Preparation Network*, *a Hiding Network*, and *a Revealing Network*. The *Preparation Network* is a neural network designed for image preparation. The *Hiding Network* assumes the responsibility of concealing the images. Lastly, the *Revealing Network* is employed to disclose hidden or secret images.

Our first step was to enable the model to handle images with high resolution, specifically 256×256. To accomplish this, we used the ImageNet [10] dataset for the cover images, and developed a new dataset for the secret images using the following procedure:

The input malware, a Python code, is transformed into an image using the Python PIL library [3]. This results in a new dataset comprising 10000 images.

These images, referred to as the 'secret', are designed to conceal the malware within a cover image. We used cover images from ImageNet and generated 10,000 images with random text arranged like code. The secret image is then fed into the *Preparation Network*. This network transforms the color-based images into a representation with three channels of useful features.

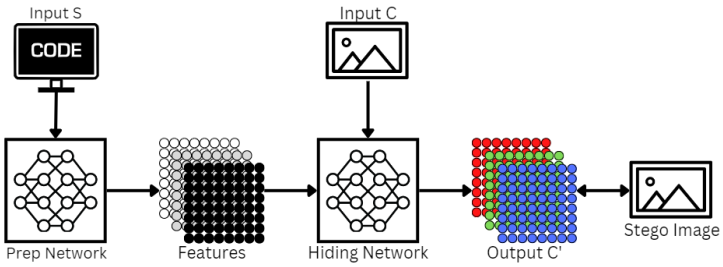


Fig. 3. The attacker initially takes a code represented as an image with text; the first, preparation network extracts features from this image. Following this, the hiding network takes in the features from the first network as well as the carrier image in which it conceals these features. The output of this network is a stego image.

This transformation is achieved by applying three convolutional layers with 3×3, 4×4, and 5×5 filters and three input channels each. Additionally, three more convolutional layers are used with the same filters but with 65 channels each. The output from these layers, combined with the cover image, is subsequently processed by the *Hiding Network*.

The architecture responsible for concealing the extracted features from the secret within the cover image is the *Hiding Network*. This process is accomplished by performing fusion operations, where the feature maps are concatenated and passed through three convolutional layers, each with 68 channels. The cover image contributes an additional three channels. This operation is repeated five times, adding up to a total of 15 layers. This flow is visually summarized in Figure 3.

The *Revealing Network* or decoder receives the stego image, which contains the concealed secret, and reconstructs the secret from it (see Figure 4). This reconstruction process involves passing the stego image through three convolutional layers, each with three channels (RGB). Subsequent fusion operations are performed, and the data is passed through three layers using the same filters previously mentioned. This operation is repeated four times, adding 12 layers to the network. Finally, the decoder applies a final convolutional layer with 3x3 filters and 65 input channels to generate the reconstructed secret image.

To extract the code from the image and execute it within the decoder application, we employed the Tesseract OCR [7]. Both the *Revealing Network* and the

OCR are published to the user within the deep-learning tutorial as plain code. Note that these codes are neutral, identified by the anti-virus as legitimate, and would easily execute on the user's machine.

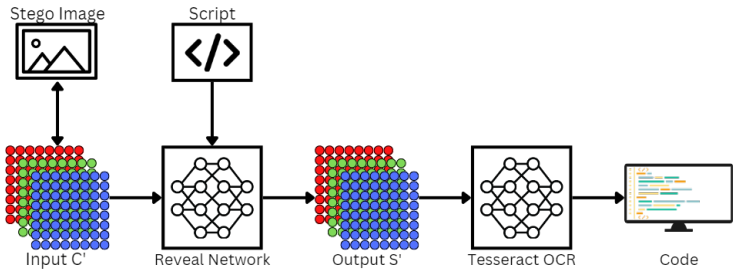


Fig. 4. The decoder's functionality of taking the stego image and extracting the concealed code via the *Revealing network*. A user-derived script subsequently takes this revealed secret to the OCR, which identifies the covert code embedded within the image, and relays it back. Following this, the script executes the hidden code.

Tuning the stenographic success

To quantify the success of the stenographic hiding and correctly define the parameters to achieve good hiding, an error function must be defined that quantifies the difference between the cover, the secret, the container, and the revealed images.

The following error function proposed by Baluja et al. [4] which designed to minimize the difference between the cover image and the stego image and between the secret image and the revealed secret image. The error terms corresponding to these differences are combined in a weighted sum, with the weight for the reconstruction error of the secret image being determined by a hyperparameter, beta (β). The error function is given by:

$$E(c, c', s, s') = \frac{1}{m \cdot n} \sum_{i=1}^{m} \sum_{j=1}^{n} (c(i,j) - c'(i,j))^2 + \beta \cdot \frac{1}{m \cdot n} \sum_{i=1}^{m} \sum_{j=1}^{n} (s(i,j) - s'(i,j))^2 \tag{1}$$

In the above equation, c and c' are the cover and stego images, s and s' are the secret and revealed secret images, and m and n are the dimensions of the images. The term β is a weighting parameter that determines the weight given to the reconstruction error of the secret image.

In our research, modifications were made to this error function that impact all three networks: Preparation Network, Hiding Network, and Revealing Network.

This was achieved by leveraging both the error rates of the secret and cover images. We suggest the following error function:

$$E_r(c, c', s, s') = \frac{(1-\beta)}{m \cdot n} \sum_{i=1}^{m} \sum_{j=1}^{n} (c(i,j) - c'(i,j))^2 + \frac{\beta}{m \cdot n} \sum_{i=1}^{m} \sum_{j=1}^{n} (s(i,j) - s'(i,j))^2 \tag{2}$$

The key distinction between the two error functions is the application of the weighting parameter, β, to the error term related to the difference between the cover and stego images and $(1 - \beta)$ to the error term related to the difference between the secret and revealed secret images. This modification gives the attacker more control over the relative importance of the two error terms. The flexibility to adjust these weights may lead to improved outcomes, depending on the specific application or use case, as shown in our results below.

4 Results

Train evaluation

We initiated the training phase for our model, which consisted of a bunch of epochs. In this context, an epoch refers to one full cycle through the entire training dataset. We selected a batch size of 32 images, a standard choice that balances computational efficiency and learning stability. This decision was also influenced by the capacity of our available hardware, specifically the 16 GB of RAM.

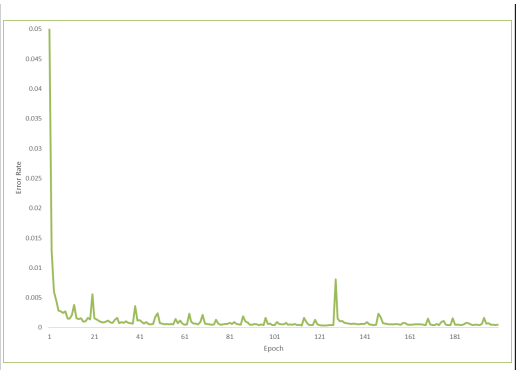
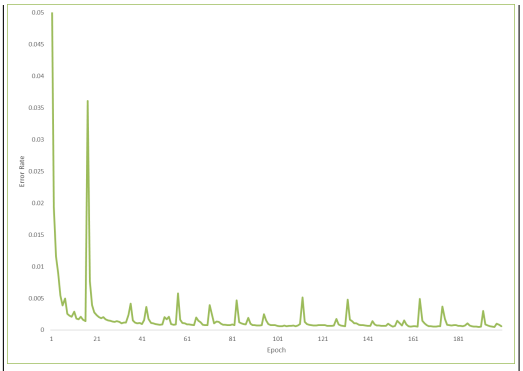
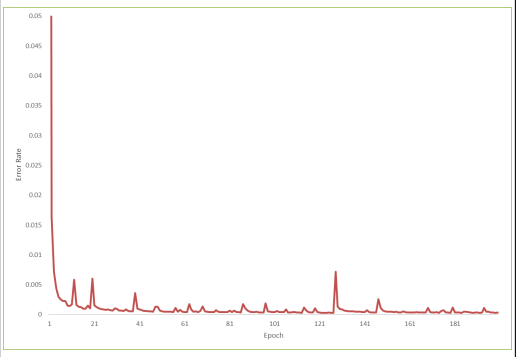
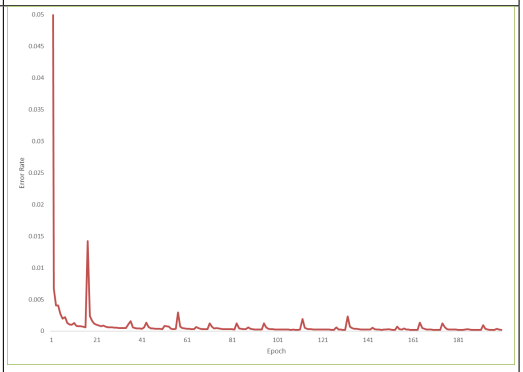
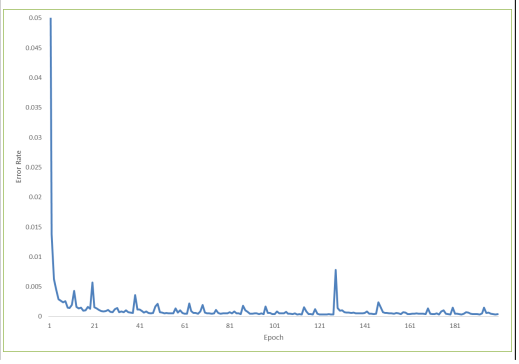
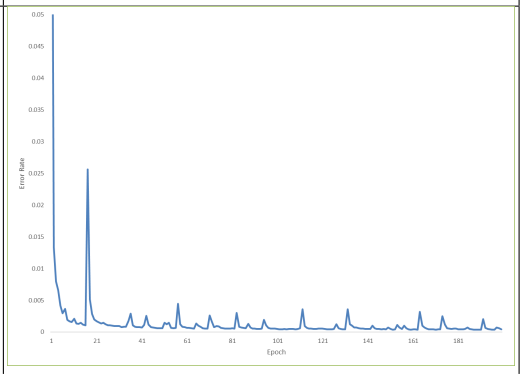
For our experiments, we utilized cover images from ImageNet [10], a large-scale image database commonly used in machine learning. Alongside these, we generated an additional 10,000 images featuring randomly arranged text that emulated a code structure. The model was trained on these images, each with dimensions of 256x256. We created each batch by randomly selecting 14 images as the secret and 14 as the covers. With 164 batches in each epoch, our model was set to run for a total of 200 epochs.

In our training process, we experimented with various values of the weighting parameter, β, which adjusts the trade-off between hiding the secret image and preserving the visual quality of the cover image. The outcomes of these experiments were compared to identify the most effective value.

We utilized the Adam optimizer [8], known for its effectiveness in training deep learning models, to optimize the learning process. Additionally, we implemented a learning rate scheduler to adjust the learning rate during the training process dynamically. This was based on whether the model's validation loss plateaued, a strategy designed to help the model converge further or potentially escape local minima.

After several hours of running the training process, we successfully obtained a fully trained model. The evolution of the error function can be observed in Table 1, which charts the error against the number of completed epochs. The

Table 1. Error for different weighting parameter, β

	$\beta = 0.25$	$\beta = 0.48$
weighted cover difference		
weighted secret difference		
Error function		

first row relates to the weighted difference between the cover and the stego images, namely $\frac{(1-\beta)}{m \cdot n} \sum_{i=1}^{m} \sum_{j=1}^{n} (c(i,j) - c'(i,j))^2$. The second row relates to the weighted difference between the secret (i.e., the malware code) and the revealed images, namely $\frac{\beta}{m \cdot n} \sum_{i=1}^{m} \sum_{j=1}^{n} (s(i,j) - s'(i,j))^2$. The last row indicates the whole error function as appears in Equation 2.

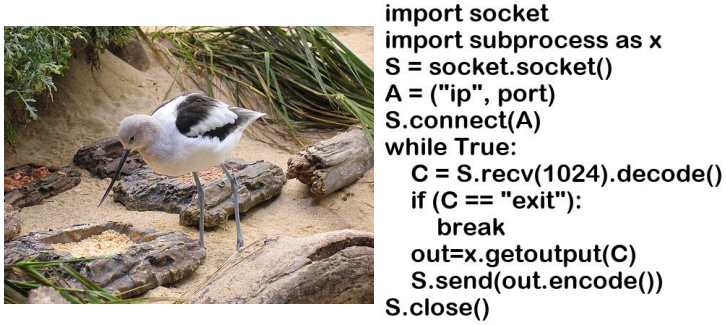
Table 2. The error at the end of the training process

β	weighted cover difference	weighted secret difference	Error function
0.25	0.00048	0.00033	0.00044
0.48	0.00058	0.00020	0.00040
0.6	0.00059	0.00016	0.00033

In the experiments, we varied the weighting parameter beta in three different models to study its impact on steganography. Beta was set to 0.25, 0.48, and 0.6, respectively. In Table 2, we present the performance of the models at the end of the trained process (i.e., the mean error values of the last batch) for each beta. (Due to the lack of space and the similar behavior, Table 1 presents only results for $\beta = 0.25, 0.48$, but in Table 2, the results for $\beta = 0.6$ also appear).

Visual example

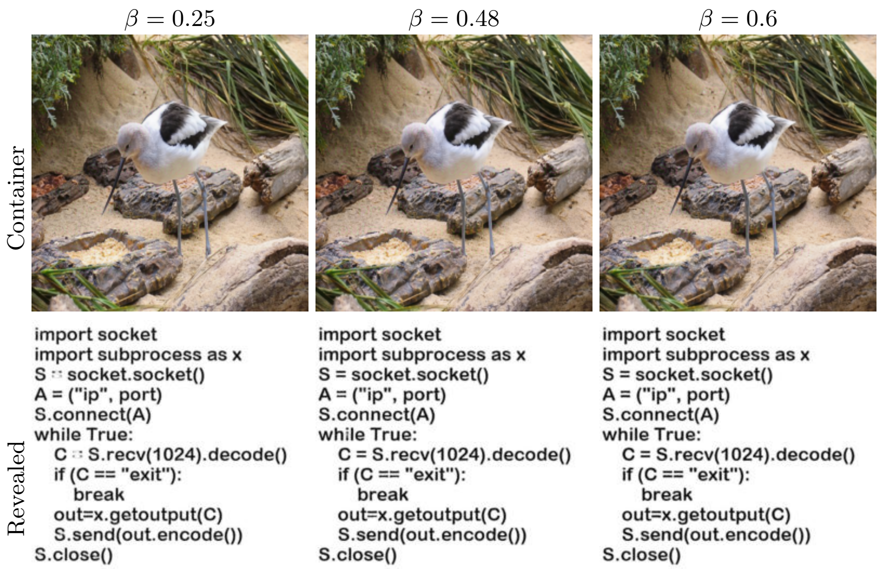
To get a visual demonstration of the visibility of the stego image and the quality of the ability to extract the code from it, Fig 5 show an example of a particular cover image and a secret image.



```
import socket
import subprocess as x
S = socket.socket()
A = ("ip", port)
S.connect(A)
while True:
   C = S.recv(1024).decode()
   if (C == "exit"):
      break
   out=x.getoutput(C)
   S.send(out.encode())
S.close()
```

Fig. 5. Cover image and secret image (malware code) examples.

Table 3. Performance of the trained models for the input images of Fig 5 with varying beta parameters



In Table 3, we exhibit the model results for this example using various values for the weighted error parameter (β). The 'Container' row demonstrates the resulting stego images post the encoding process, where the secret images have been embedded into the cover images. The 'Revealed' row shows the secret images posted after the decoding process.

The visual inspection of the revealed images and the comparison with the original secret images give an insight into the model's performance. Also, the stego images, when compared to the cover images, indicate the degree of perceptual transparency achieved in the process. The results inferred that a beta value of 0.25 yielded the most optimal outcome, offering a good balance between the concealment of the secret image and the perceptual transparency of the stego image.

Figure 6 visually represents the pixel-by-pixel difference between an example of a Cover and a Container images, calculated using the Mean Squared Error (MSE). This measure quantifies the discrepancy between the corresponding pixels in both images by computing the squared differences. Visually, we can observe that while increasing beta, the values of the secret image stand out more in the MSE matrix (for $\beta = 0.6$, one can actually read part of the secret code), even though it is difficult to recognize that code in the container image.

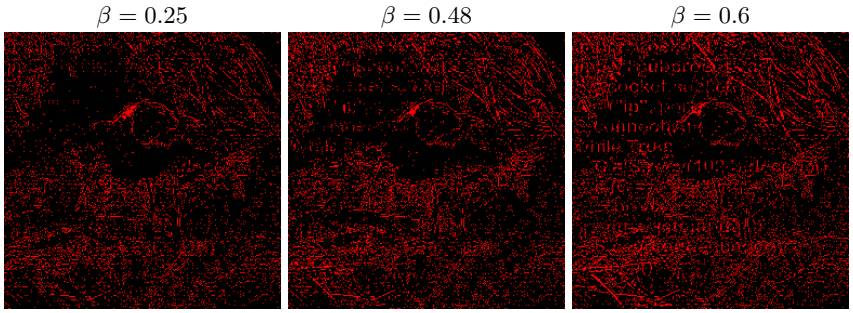


Fig. 6. Visual representation of the Mean Squared Error (MSE) computed difference between the Cover and Container images for $\beta = 0.25, 0.48$ and 0.6.

5 Discussion and Conclusions

In this study, we investigated the use of deep neural networks for embedding and extracting secret information within images. The primary objective is to extract text from the secret image while ensuring that the secret remains invisible in the stego image. We introduced a novel approach that allows control over the weight distribution between cover and secret losses using beta values, enabling us to strike a balance between successful extraction and concealment.

Through rigorous experimentation and analysis, we evaluated the performance of our model on different beta values. The table above summarizes the results obtained for each beta, including the cover image, secret image, stego image, revealed secret, and the difference between the cover and stego images.

Our findings demonstrate that by adjusting the beta value, we can effectively control the visibility of the secret image in the stego image. Lower beta values prioritize the cover loss, resulting in a stego image where the secret remains hidden. On the other hand, higher beta values assign more weight to the secret loss, allowing for the successful extraction of the embedded text.

Furthermore, we emphasize that our focus was primarily on the successful extraction of text from the secret image, rather than achieving a perfect representation of the secret image. This approach strikes a balance between covert communication and maintaining the confidentiality of the hidden information.

It is worth noting that for betas 0.6 and 0.48, the difference image reveals parts of the embedded secret. Although not easily visible in the stego image itself, these partial revelations in the difference image could potentially pose a security risk in certain scenarios. However, it is important to consider the overall objective of extracting text from the secret image, which was successfully achieved without compromising the secrecy of the hidden information.

By achieving the objective of extracting text from the secret image while ensuring its invisibility in the stego image, our research contributes to the field of steganography. It provides a valuable tool for applications requiring secret communication and data hiding.

Overall, our study highlights the effectiveness of deep neural networks in achieving the dual objective of successfully extracting text from the secret image and concealing the secret in the stego image. These findings have significant implications for the development of secure communication systems and provide a foundation for further advancements in the field of steganography.

References

1. GitHub: Let's build from here — github.com. https://github.com, [Accessed 19-Jun-2023]
2. Key GitHub Statistics in 2023 (Users, Employees, and Trends) — kinsta.com. https://kinsta.com/blog/github-statistics/github-usage, [Accessed 19-Jun-2023]
3. Pillow — pillow.readthedocs.io. https://pillow.readthedocs.io/en/stable/, [Accessed 18-Jun-2023]
4. Baluja, S.: Hiding images in plain sight: Deep steganography. Advances in neural information processing systems **30** (2017)
5. Goodfellow, I., Pouget-Abadie, J., Mirza, M., Xu, B., Warde-Farley, D., Ozair, S., Courville, A., Bengio, Y.: Generative adversarial nets **27** (2014)
6. Johnson, N.F., Jajodia, S.: Exploring steganography: Seeing the unseen. Computer **31**(2), 26–34 (1998). https://doi.org/10.1109/MC.1998.4655281
7. Kay, A.: Tesseract: an open-source optical character recognition engine. Linux Journal **2007**(159), 2 (2007)
8. Kingma, D.P., Ba, J.: Adam: A method for stochastic optimization. arXiv preprint arXiv:1412.6980 (2014)
9. Kingma, D.P., Welling, M.: Auto-encoding variational bayes. arXiv preprint arXiv:1312.6114 (2013)
10. Russakovsky, O., Deng, J., Su, H., Krause, J., Satheesh, S., Ma, S., Huang, Z., Karpathy, A., Khosla, A., Bernstein, M., et al.: Imagenet large scale visual recognition challenge. International journal of computer vision **115**, 211–252 (2015)
11. Subramanian, N., Elharrouss, O., Al-Maadeed, S., Bouridane, A.: Image steganography: A review of the recent advances. IEEE Access **9**, 23409–23423 (2021). https://doi.org/10.1109/ACCESS.2021.3053998
12. Wang, Z., Liu, C., Cui, X.: Evilmodel: hiding malware inside of neural network models. In: 2021 IEEE Symposium on Computers and Communications (ISCC). pp. 1–7. IEEE (2021)
13. Wu, D.C., Tsai, W.H.: A steganographic method for images by pixel-value differencing. Pattern Recognition Letters **24**(9), 1613–1626 (2003). https://doi.org/10.1016/S0167-8655(02)00402-6

Real-time video tracking on small single-board computers (SBCs)

Technical Report

David Denisov[1], Dan Feldman[1], Shlomi Dolev[2], and Michael Segal[2]

[1] University of Haifa, Haifa, Israel.
E-mail: David Denisov: daviddenisovphd@gmail.com.
[2] Ben-Gurion University of the Negev, Beer-Sheva, Israel

Abstract. We present an efficient method to track multiple moving objects in a video. To this end, we utilize motion vectors and clusters, which are computed very efficiently in common video encoders, usually via dedicated hardware. We suggest a provably good tracking algorithm for clustering these vectors, by considering them as segments. For this, we utilize Coresets which are essentially a weighed set of points that approximates the fitting loss for every model, up to a multiplicative factor of $1 \pm \varepsilon$. We demonstrate the empirical contribution of our algorithm by running it on a micro-computer (Le-Potato) with a real-time video.

1 Introduction

The goal of this work is to provide a novel tracking method incorporating classical machine learning in contrast to the current prevalent deep learning methods, with the primary goal of reducing the running time and improving the robustness.

1.1 Video tracking

The problem of tracking objects in RGB videos is a well-studied problem, for which numerous heuristics using various approaches were proposed. A meta-survey on such approaches [14] states that in recent years there were thousands of papers published on this subject. One of the very prominent approaches is utilizing neural networks. While neural networks yield unprecedented results, such improvements come at the price of training, which along with the labeling of data is rather costly. Another challenge is the cost of utilizing the results after the training, which requires at least mid-level GPUs to achieve 30-fps (frames per second) in real-time. Moreover, recently [8], demonstrated the problem where even small changes in the data may "fool" the network. We note that while there were works on addressing similar problems in recent years using more sophisticated training, see for example [3], it is uncertain whether more sophisticated "attacks" could cause such problems.

1.2 Motion vectors

Motion vectors are computed in real-time as part of existing encoders for videos, such as H.264 [13], H.265, etc. [6]. In general, those are mapping from one frame to another, with the goal of usually minimizing some loss function, such as Mean Squared Error (MSE) between the RGB values, to allow keeping only this mapping (as a vector) and the difference between the blocks.

In our work, we consider the rather simple case where the mapping is from a frame to its previous frame. For a more detailed explanation on motion vectors and their computation see [13].

1.3 Our approach

We use a very different approach that does not suffer from the aforementioned drawbacks. There is no training data or training process, the computation is efficient as evidenced by our analysis and tests, and if the algorithm produces a wrong result the reason for this faulty result can be easily traceable. For this, we utilize motion vectors, which are computed in real-time as part of existing encoders for videos. We consider each vector as a segment and compute a provable clustering of the segments.

We emphasize that for each motion vector, we add its degree to $(0, 1)$ and $(1, 0)$; thus we have 4-dimensional segments. This allows us to take into consideration the direction of the vectors in our clustering.

2 Empirical evaluation

Tracing method: We have implemented a clustering-based method in python 3.8 [12] utilizing [2], [1], [10], and Numpy [4].

Goal: In those tests, we aim to demonstrate that by utilizing our clustering method we can in real-time and on standard hardware track the movement of objects in a video.

We emphasize that since the computation utilizes only the motion vectors, and not the RGB part of the image, this method also allows privacy preservation to some degree while providing real-time object tracking.

Input video: In this test, we have used a clip of the Big Buck Bunny video [7] that contains 400 frames in the resolution of 720x1280p. We have chosen this video due to its prevalence in the video tracking community, and is licensed under the Creative Commons Attribution 3.0 license, which essentially entails "you can freely reuse and distribute this content, also commercially, for as long you provide a proper attribution"; cited from the official site for the project, https://peach.blender.org/about. We have chosen this part of the video since it contains a bunny walking across the stationary background, thus the tracking can be validated rather simply in a visual sense.

2.1 Laptop test

In this experiment, we have run the experiment on a standard Laptop with an Intel Core i3-1115G4, and 16GB of RAM. The video was streamed in real-time from a file, we have utilized a clip at 720x1280p resolution from [7]; provided in the supplementary material.

In what follows we provide a few examples from the clip, the entire clip of the tracking result is provided in the supplementary.

Fig. 1. A subset of the results for the experiment at Section 2.1. The left column is the largest cluster of the motion vectors computed and the right column is the center of this cluster and mean direction. The top row demonstrates a section of the video where the bunny rises from the cave. The middle and bottom rows demonstrate a section of the video where the bunny walks to the left and are taken with a small time difference in the video.

As can be seen in the top images the movement direction is to the opposite direction from the movement of the bunny. This occurred due to the bunny entering the field of view, and thus the closest (in color) parts from the previous frame are the parts already seen, and thus the motion vectors pointed in the

opposite direction from the actual movement. Unfortunately by only looking at the motion vectors we cannot remove this problem when objects enter or disappear from the field of view. Nonetheless, observe that the cluster found, which aims to represent the center of the moving object is the center of the bunny, and thus there is no visually evident problem in identifying the center of the object.

On the other hand, as can be seen in the bottom images, when the bunny has entirely entered the field of view and started walking, we obtain visually logical tracking with the predicted movement following the movement of the bunny and the cluster being the center of the object.

Running time: To provide context for the computation time we note that the entire running time (including decoding the video and saving the resulting tracking video) was 1.44 seconds, of which the decoding the video and saving the resulting tracking video took 1.22 seconds, i.e., our tracking algorithm took only 0.22 seconds. Since there are 400 frames in the clip, the tracking algorithm processed above $1,800$ frames per second, and even including decoding the video and saving the resulting tracking video we obtain 325 frames per second.

To put it into perspective, the running time of YOLOv8 [9], which is an improvement over YOLOv5 [5], over the same clip is 33.4 seconds, which entails a processing rate of 12 frames per second, output attached at the supplementary material.

2.2 Re-running for Low-end board:

We have rerun the previous test (Section 2.1) for Libre computes AML-S905X-CC (also known as Le Potato) https://libre.computer/products/aml-s905x-cc/, which is a small single-board computer similar to Raspberry Pi [11]; we have used the official Raspberry Pi OS distributed for Le Potato.

Due to missing support for ARM architecture, we have extracted the motion vectors beforehand and transferred them as a Numpy array; thus this is not exactly a stand-alone test that utilizes only the video.

In this, we aim to demonstrate that our methods can support extremely low-end systems in reasonable real-time. We have obtained essentially the same results, as can be expected since the only sources for noise are ties broken arbitrarily in the clustering and noise from the samples we take.

The entire running time (excluding innit, but including decoding the video and saving the resulting tracking video) was 16.2 seconds, of which decoding the video and saving the resulting tracking video took 14.61 seconds, i.e., our tracking algorithm took only 1.59 seconds. Since there are 400 frames in the clip, the tracking algorithm processed above 250 frames per second, and even including decoding the video and saving the resulting tracking video we obtain 27 frames per second.

Acknowledgement

This research was (partially) funded by the Israeli Science Foundation (Grant No. 465/22).

References

1. Bommes, L., Lin, X., Zhou, J.: Mvmed: Fast multi-object tracking in the compressed domain. In: 2020 15th IEEE Conference on Industrial Electronics and Applications (ICIEA). pp. 1419–1424 (2020). https://doi.org/10.1109/ICIEA48937.2020.9248145
2. Bradski, G.: The OpenCV Library. Dr. Dobb's Journal of Software Tools (2000)
3. Chen, B., Chin, T.J., Klimavicius, M.: Occlusion-robust object pose estimation with holistic representation. In: WACV (2022)
4. Harris, C.R., Millman, K.J., van der Walt, S.J., Gommers, R., Virtanen, P., Cournapeau, D., Wieser, E., Taylor, J., Berg, S., Smith, N.J., Kern, R., Picus, M., Hoyer, S., van Kerkwijk, M.H., Brett, M., Haldane, A., del Río, J.F., Wiebe, M., Peterson, P., Gérard-Marchant, P., Sheppard, K., Reddy, T., Weckesser, W., Abbasi, H., Gohlke, C., Oliphant, T.E.: Array programming with NumPy. Nature **585**(7825), 357–362 (Sep 2020). https://doi.org/10.1038/s41586-020-2649-2, https://doi.org/10.1038/s41586-020-2649-2
5. Jocher, G., Stoken, A., Borovec, J., Chaurasia, A., Changyu, L., Hogan, A., Hajek, J., Diaconu, L., Kwon, Y., Defretin, Y., et al.: ultralytics/yolov5: v5. 0-yolov5-p6 1280 models, aws, supervise. ly and youtube integrations. Zenodo (2021)
6. Pereira, F.C., Pereira, F.M.B., Pereira, F.C., Pereira, F., Ebrahimi, T.: The MPEG-4 book. Prentice Hall Professional (2002)
7. Roosendaal, T.: Big buck bunny. p. 62. SIGGRAPH Asia '08, Association for Computing Machinery, New York, NY, USA (2008). https://doi.org/10.1145/1504271.1504321, https://doi.org/10.1145/1504271.1504321
8. Su, J., Vargas, D.V., Sakurai, K.: One pixel attack for fooling deep neural networks. IEEE Transactions on Evolutionary Computation **23**(5), 828–841 (2019)
9. Terven, J., Cordova-Esparza, D.: A comprehensive review of yolo: From yolov1 to yolov8 and beyond. arXiv preprint arXiv:2304.00501 (2023)
10. Thakur, A., Papakipos, Z., Clauss, C., Hollinger, C., Boivin, V., Lowe, B., Schoentgen, M., Bouckenooghe, R.: abhitronix/vidgear: Vidgear v0.2.5 (Feb 2022). https://doi.org/10.5281/zenodo.6046843, https://doi.org/10.5281/zenodo.6046843
11. Upton, E., Halfacree, G.: Raspberry Pi user guide. John Wiley & Sons (2016)
12. Van Rossum, G., Drake, F.L.: Python 3 Reference Manual. CreateSpace, Scotts Valley, CA (2009)
13. Wiegand, T., Sullivan, G., Bjontegaard, G., Luthra, A.: Overview of the h.264/avc video coding standard. IEEE Transactions on Circuits and Systems for Video Technology **13**(7), 560–576 (2003). https://doi.org/10.1109/TCSVT.2003.815165
14. Zou, Z., Shi, Z., Guo, Y., Ye, J.: Object detection in 20 years: A survey. arXiv preprint arXiv:1905.05055 (2019)

Efficient Home Therapy Scoring: Rapid MediaPipe Integration with Precise OpenPose and CNN *

Y. Segal and O. Hadar

Communication Systems Engineering Department,
Ben Gurion University of the Negev (BGU),
Beer-Sheva, 84105,
Israel

Abstract. This research proposes a unique approach to collect anonymized patient data on limb movements during physical therapy exercises using a combination of OpenPose and MediaPipe skeleton extraction technologies. The proposed system allows the system to score the quality of the patient's movement. The study compares expert skeleton extraction using OpenPose with real-time patient skeleton extraction using MediaPipe, ensuring the system's effectiveness in a patient's home environment. Additionally, a fully linked network is employed as a MediaPipe to OpenPose conversion to address MediaPipe's inaccuracy in skeletal structure. To classify the accuracy and quality of patients' movements, we use EfficientNet as our CNN, which is compact and suitable for implementation in regular mobile phones. We created a human gesture database and utilized EfficientNet to tag, measure, and infer human gestures. We imitated patients' movements and augmented them to address the lack of labeled physiotherapy exercise videos to enhance the database's performance. The skeletons are then grouped into a single array and fed to the CNN, obtaining their low-dimensional vector representation, which is categorized using the empirical scoring technique to measure the patient's exercise compared to their physiotherapist. The proposed system is tested on a dataset of six different physiotherapy exercises, achieving an accuracy of 91.8%.

.

Keywords: OpenPose, Anonymous Gestures, Simulation, Siamese Network, Physiotherapy exercises, Metaverse

* This work was supported a grant from the Ministry of Science & Technology, Israel & The Ministry of Education, Youth and Sports of the Czech Republic.

1 Introduction

The metaverse has become one of the hottest research areas in the industry today, which requires new methods to analyze human gestures [1]. With the recent COVID-19 epidemic, the importance of remote diagnosis and treatment has been brought into sharp contrast, highlighting the need for remote physiotherapy treatment. Artificial neural networks have advanced where systems can collect, analyze, and interpret human gestures in a remote 3D environment through a camera video feed. The study compares expert skeleton extraction using OpenPose with real-time patient skeleton extraction using MediaPipe, ensuring the system's effectiveness in a patient's home environment. Scoring is enabled for users to measure their performance and see their improvement.
OpenPose [2] is a computer vision library that allows users to extract 2D and 3D pose information from a video. It provides the spatial location of each human joint, enabling tracking of human movement. On the other hand, MediaPipe [3] is a cross-platform framework that provides the real-time perception of objects and human body parts in the physical world, using techniques such as computer vision, machine learning, and depth sensing. While OpenPose provides precise skeleton extraction, MediaPipe offers real-time patient skeleton extraction, making it ideal for use in a patient's home environment. EfficientNet is our CNN, a compact and suitable neural network for implementation in regular mobile phones to classify patient movements' accuracy and quality. EfficientNet is a state-of-the-art neural network architecture that achieves state-of-the-art performance on image recognition tasks while requiring fewer parameters and lower computational resources than previous models. This paper aims to extract human body movements from a video, including gesture name, gesture time, repetition count, gesture speed, movement acceleration, and more. We will present techniques for monitoring and evaluating the individuals' non-contact physiotherapy activities using algorithms such as Autoencoder [4], Siam's Twines [5], and DTWNet [6]. Patients' movements will be imitated and augmented to address the lack of labeled physiotherapy exercise videos to enhance the database's performance. We will describe a neural network as a tool for analyzing, measuring, and labeling skeleton graphs in time and space. The practical results of our research study will be presented through collaboration with the University of Prague and within the framework of the Israeli Ministry of Science and Technology. Our innovative solution will help to lower the danger of physical contact and the associated costs of rehabilitation by enabling learning and assessment of physiotherapy activity in a home environment during rehabilitation. We will demonstrate how to combine data from multiple sources and extract metrics to provide a quantitative description and labeling of motions using only one or a few cameras. The VOCD will also be utilized to identify transient patterns in the patient's motions while performing the exercise and generate a therapy report for the therapist. The underlying concept of the algorithm is to demonstrate how the study findings could be used, for example, for remote patient rehabilitation. We aim to provide patients with quantifiable data and inform them of discrepancies between their necessary and actual gestures. This capability may be employed for remote therapy, particularly in cases involving a large number of patients recuperating from hip, knee, elbow, or shoulder surgeries. The tutorial outcomes will enable participants to create a family of neural network architectures,

creating a wide range of contactless medical solutions. More broadly, this will permit tags to be attached to motions in various industries, from athletics to choreography to physical security to intelligence and behavioral analysis.

2 Related Work

2.1 Human Skeletal Data Extraction Methods

OpenPose is a real-time multi-person system that can detect key points in the human body, hand, face, and foot movements using images or frames in videos [1], [2], [7], [8]. This system generates 75-component skeleton vectors (25 key points with three components - x, y, and c) representing the data. Although OpenPose is a powerful tool, it requires a high-performance computer. To increase performance, we developed a method using the H264 video encoder motion vector as a vertex tracking algorithm, eliminating the need for OpenPose [1], [9]. Another approach to avoiding OpenPose is to use Mediapipe. Body tracking is critical to the Mediapipe method, but the cost is its accuracy. Other key points extraction systems require assistance before or during operation, making them unsuitable for home use. Our research aims to develop an affordable and user-friendly system that mix between OpenPose and Mediapipe. A gold standard for body monitoring involves using infrared markers, but it is expensive and complex. Commercial motion capture systems like Vicon [10], OptiTrack [10], [11], Qualysis [12] , and BTS [13] use multiple synchronous infrared cameras, reactive markers, and unique body models. While these systems are highly accurate and offer multiple current models, they are expensive, time-consuming to prepare for measurements, require trained operators, and cannot be used at home. This work presents high-quality skeleton extraction techniques that use offline OpenPose for extracting the expert key point. For The patient, we need real-time channels like Google Mediapipe [3]. The developers of OpenPose have published several articles that provide a comprehensive account of the system's research and its evaluation of posture based on previously taught body part models. The system is frame-based, assigning an independent position estimate to each image. Other systems, such as Microsoft Kinect [14], Xsens [15], Rokoko [16], and the intelligent captain suit manufactured, are examples of full-body commercial capturing suits that can be used for general and specialized applications. Some experts from Dublin University have also established a framework for obtaining the 3D trajectory of a golf swing or other sports equipment using inertial sensors such as accelerometers, gyroscopes, and magnetic sensors. Mathematical transformations and Kalman filters estimate a 3D position in space based on acceleration, angular velocity, and gravitational force. Overall, multiple techniques are available for human skeletal data extraction, with each system having advantages and disadvantages. By mixing such techniques, we aim to develop an efficient and accurate system for extracting human skeletal data that can be used for various applications.

2.2 Extracting Human Pose key points: A Review of Techniques and Applications.

Human pose accuracy is critical to various applications, such as medical rehabilitation, sports performance analysis, and security monitoring. Several techniques and frameworks have been developed to extract and analyze human pose data from images and videos, including OpenPose, MediaPipe, AlfaPose, PoseNet, OpenCV, Mmpose, BlazePose, HRNet, YOLO, and PNASNet.

OpenPose is a widely-used open-source software library that uses deep learning to extract and analyze human pose data from 2D images and videos [8]. The system can detect up to 135 key points, including body parts, face, and hands, with high accuracy and real-time performance. The OpenPose model uses a multi-stage convolutional neural network (CNN) architecture with part affinity fields (PAFs) to estimate the body part locations and their connection. The system has been applied to various applications, such as medical rehabilitation, dance analysis, and sports performance.

MediaPipe is another open-source software library that provides real-time multimedia processing, including human pose estimation [3]. The system uses a hybrid approach that combines a 2D Human pose estimation model and a 3D Human pose estimation model to improve accuracy and robustness. The 2D model uses a deep neural network to estimate the 2D pose key points, while the 3D model uses a regression tree to estimate the 3D pose key points. The system can also detect face landmarks, hand landmarks, and object detection.

AlfaPose is a deep learning-based framework that uses a multi-stage approach to estimate human pose in images and videos [17]. The system first detects human body parts using a YOLO object detection model, then estimates the joint body locations using a convolutional pose machine (CPM) model. The system has been trained on a large-scale dataset and achieved state-of-the-art accuracy in several benchmarks.

PoseNet is a deep learning-based model that can estimate human pose in real-time using a single RGB camera [18]. The system uses a fully convolutional neural network (FCN) architecture to estimate the joint body locations. The system has been optimized for real-time performance and can be used in various applications, such as fitness tracking, augmented reality, and gaming.

OpenCV is an open-source computer vision library that provides various image and video processing algorithms, including human pose estimation [19]. The library includes several pre-trained models, such as the HOG+SVM model, which can detect body parts in images. The library also provides various algorithms for image segmentation, object tracking, and feature extraction, which can be used to improve pose estimation accuracy.

Mmpose is an open-source pose estimation framework that provides various models and algorithms for Human pose estimation [20]. The framework includes several state-of-the-art models, such as the HRNet model, which uses a high-resolution network to estimate joint body locations. The framework also provides various data augmentation techniques and loss functions to improve accuracy and robustness.

BlazePose is a real-time Human pose estimation model that uses a lightweight neural network architecture for fast and accurate pose estimation [21]. The model uses a

convolutional neural network with residual connections to estimate joint body locations. The model has been optimized for mobile and embedded devices and can be used in various applications, such as virtual try-on and fitness tracking.

HRNet is a high-resolution network that can estimate human pose accurately and efficiently [22]. The network uses a multi-resolution fusion strategy to combine the high-resolution and low-resolution features, improving accuracy and reducing computational complexity. The network has achieved state-of-the-art accuracy in several benchmarks and can be used in various applications, such as sports performance analysis and medical rehabilitation.

YOLO (You Only Look Once) is an object detection framework that can detect and track human body parts in images and videos with high accuracy and real-time performance [23]. The system uses a deep neural network with a single forward pass to detect and track objects in an image. The system has been applied to various applications, such as pedestrian detection, face detection, and sports action recognition.

In conclusion, measuring human pose accuracy is a critical task in various applications, and several techniques and frameworks have been developed to extract and analyze human pose data from images and videos. These include OpenPose, MediaPipe, AlfaPose, PoseNet, OpenCV, Mmpose, BlazePose, HRNet, YOLO, and PNASNet. Each of these techniques and frameworks has advantages and disadvantages, and the choice of the most suitable method depends on the specific application requirements, such as accuracy, real-time performance, and resource constraints.

2.3 Measuring pose accuracy

Measuring human pose accuracy is crucial in various applications, including virtual reality, gaming, medical diagnosis, and sports training. The accuracy of human pose estimation methods is usually evaluated by comparing the estimated key points with the actual key point locations, which are often obtained using expensive sensors, such as VICON [10]. However, recent advances in machine learning have enabled the use of neural network (NN) architectures for human pose estimation, which can achieve high accuracy without expensive sensors. One commonly used method for measuring key point location accuracy is the mean per joint position error (MPJPE) [2], which computes the average distance between the estimated and real key point locations, typically in millimeters. Another method is the Percentage of Correct Key points (PCK) [24], which measures the percentage of estimated key points that are within a certain threshold distance from the key point locations. Compared to sensor-based methods.

3 Proposed Method

The primary objective of this study is to investigate human body movements by comparing them to a ground truth movement, which is an exercise performed by a physiotherapist. To achieve this objective, we propose a schema (see **Fig. 1**) that involves extracting body movements from a video stream and representing them as a collection of vectors that depict a graph of the human skeleton. The OpenPose algorithm is used to extract the human posture from a video frame, which is then converted

into a vector with 75 dimensions. Since each exercise has its own dominant vertices, an algorithm is required to determine which vertices are relevant. To solve this problem, we use the PCA algorithm, which measures the variance of each vertex and selects the dominant vertices. An autoencoder is used to reduce the 75-dimensional vector to 15 main dimensions, which are then fed into the DTW algorithm to calculate the distance between the therapist and the patient vectors. This distance is normalized to produce an exercise score. To overcome the challenge of creating a large database in the medical field, we have developed an algorithm that generates human movement from a textual description, which is then converted into a vector representation similar to that produced by OpenPose. The resulting data set is processed through a neural network, and the vector dimension of the vector is reduced. In summary, the proposed schema involves using the OpenPose algorithm to extract human posture, the PCA algorithm to select dominant vertices, an autoencoder to reduce the vector dimension, and the DTW algorithm to calculate the distance between vectors. The synthesized data set generated from a textual description is processed through a neural network, and the vector dimension of the vector is reduced.

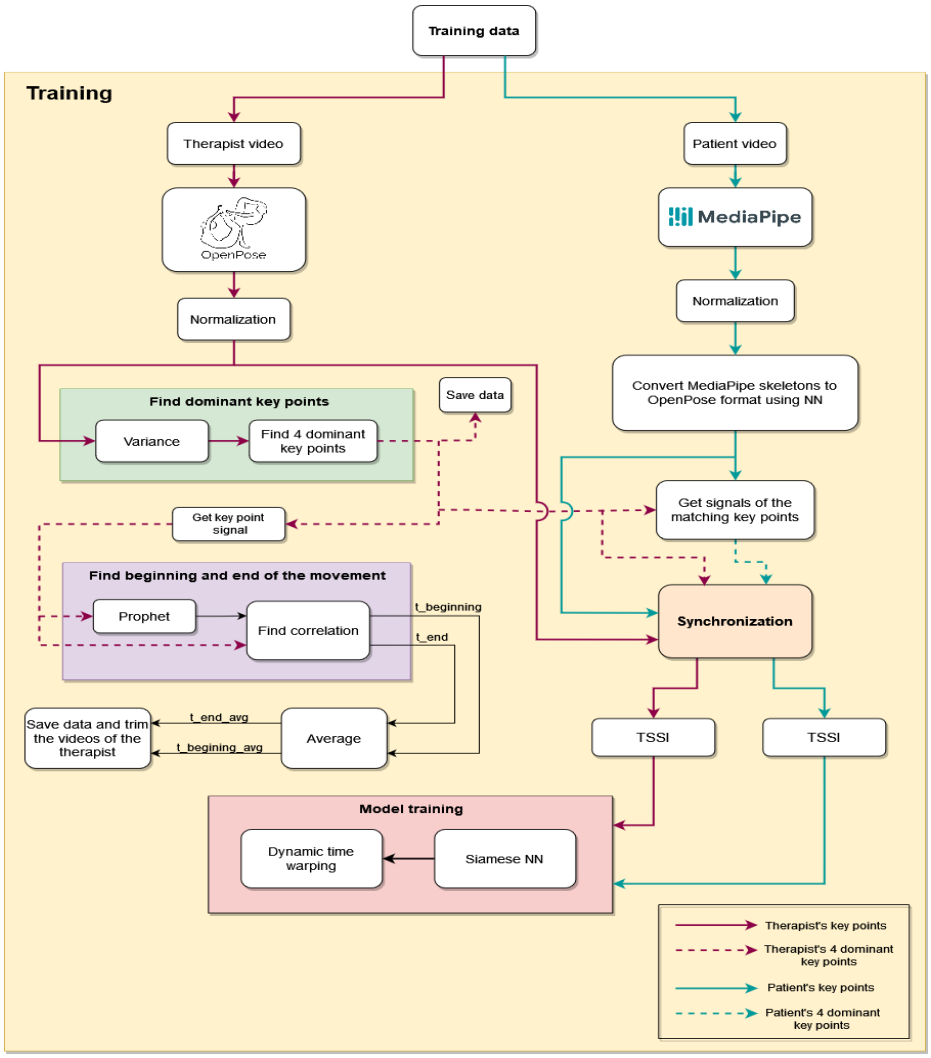


Fig. 1. Model

In the synchronization process, for each dominant key point of the therapist, the corresponding matching key point of the patient is selected, and the Fourier Transform is calculated to find the dominant frequency for each key point pair. The phase difference between each pair of matching key points is then calculated using the phase detection algorithm, and the average phase is calculated to synchronize the patient and therapist's movements. The output of this process is two synchronized skeletons, one for the therapist and one for the patient.

4 Patients Database

4.1 General description.

The proposed training set is based on physiotherapy exercises developed by Ben-Gurion University and the University of Prague with the support of the Chief Scientists of Israel and the Czech Republic. There are six basic physiotherapy exercises in the database, which have been carefully selected to be suitable for analyzing and processing with a single camera (two-dimensional processing)..

4.2 Database exercises content

There are 100 participants in the database, who each perform six exercises.

(1) AFR - Exercise arm full range (side view)
(2) ARO - Exercise arm rotation (front view)
(3) LBE - Exercise leg backward extension (exercise with a chair, side view)
(4) LFC - Exercise lifting from the chair (side view)
(5) SLL - Exercise side leg lift (lower limb abduction, front view behind the chair)
(6) TRO - Exercise trunk rotation (front view)

Ten cycles comprise each exercise (e.g. rotating the right arm). Exercises are performed once with a right tilt and once with a left tilt (for example, once with a right foot rotation and once with a left foot rotation). A total of about 7500 motion cycle videos have been tagged and timed in the database.

4.3 Database as human skeletons

The entire database has been encoded as skeletons - a skeleton in every frame. A NUMPY array file, and a JSON file are used to maintain the database. Performing exercises creates skeletal structures. The human body is represented by 25 vertices in each skeleton. The vertex has three components: Coordinate X, Coordinate Y, and Coordinate C, which indicates the level of certainty about each point in the skeleton on a scale from 0 to 1 (1-absolute certainty, 0 absolute uncertainty). Therefore, a single skeleton extracted from a frame is represented by a single vector with 75 components (3 X 25). The JSON files are structured so that one file exists for each frame, and inside each file is a list of several skeletons, one for each person in the image. Both the MATLAB files and the NUMPY files are organized in the same manner. Every file contains a matrix. The matrix contains 75 columns and the number of rows is equal to the number of frames in the video. In other words, each row describes an individual skeleton. If more than one skeleton was sampled in a single frame, then only the first skeleton, marked with ID 0, will be extracted.

5 Gesture generator via text script (augmentation)

In our work, we require a large dataset with enough training examples, but we use synthetic data in the absence of such databases. We use synthetic data to classify body movements from synthetic human skeletal vectors, which can be achieved by designing and training a neural network. To create synthetic data for training the neural network, we gathered and filmed videos of predetermined movements, then converted them to OP format and manually tagged them. We created a "Pose Dictionary" corresponding to each pose, defined according to OP, and complementing the vertices of a human skeletal body. Each movement is built from a sequence of poses in the pose dictionary, and each movement is derived from a sequence of poses. We created an artificial skeletal vector to train the neural network with many artificial skeletons. To create these vectors, we used a Python program that takes a CSV file as input and creates a set of vectors representing human skeletal movement from input parameters, including the name, speed, number of repetitions, and noise from the movement.

We then divided the database into training and testing sets. To improve the synthetic data further, we created a Synthetic Motion Simulator that takes a CSV file as input containing movement names, speed, repetitions, and noise. The output consists of a .npy matrix of the synthetic movements and a video file of the skeletal movement in a .mp4 format. The process of creating one movement is depicted in **Fig. 2**.

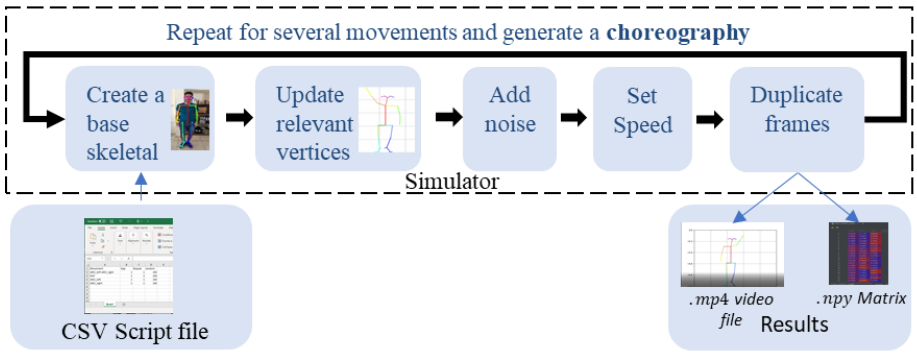


Fig. 2. Simulator Flow

Initially, we updated only the relevant vertices for the movement while the other body parts remained still, which did not reflect natural human movement. We later added noise to the movement of the objects, making them look like they were moving more naturally. We also added the number of repetitions and the movement speed to the parameters. By combining several movements in a parallel manner or a sequence, we were able to create a new human movement. We can see in **Fig. 3** and **Fig. 4** that the relevant vertices have changed, however, the body as a whole remains the same.

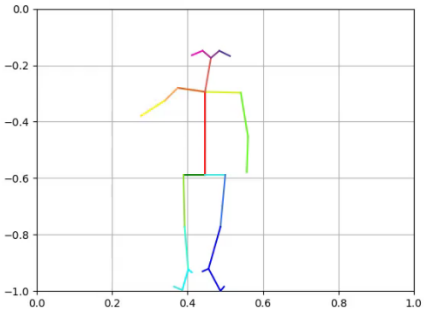


Fig. 3. ARO right movement.

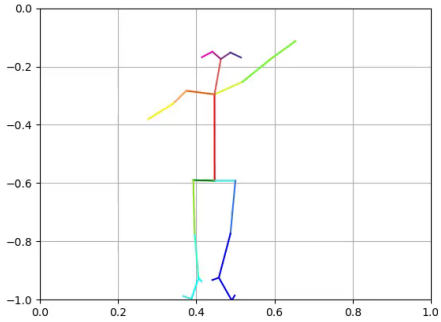


Fig. 4. ARO right and ARO left combined

In conclusion, synthetic data is a valuable solution for creating datasets when an adequate amount of real-world data is unavailable. Our proposed system of synthetic data creation and neural network classification can be used to train models to recognize body movements. The Synthetic Motion Simulator can also improve synthetic data, making it look more realistic and usable for training. This technology has significant potential in various fields, such as physical therapy and sports training, to monitor and evaluate body movements to improve performance and reduce the risk of injury.

6 The Neural Network

We have investigated the Super Object technique for understanding human movement. Our methodology seeks to provide a comprehensive solution for various human movement analysis issues. To implement this technique, we utilized TSSI. To demonstrate the effectiveness of the Super Object technique, we conducted experiments using a Convolutional Neural Network (CNN) with a resolution of 49x49, tailored for small images. Our CNN classification model employed the state-of-the-art EfficientNet-B7 classifier architecture, which has achieved remarkable performance in image classification tasks. EfficientNet is a revolutionary technique for scaling CNNs that consider the depth and breadth of the network, as well as the resolution of the input image. By striking a balance between accuracy, computational cost, and data requirements, EfficientNet is a suitable technique for improving CNN performance in various applications. We applied various strategies, such as data augmentation and transfer learning, to enhance our model's learning and prediction abilities. For the transfer learning approach, we froze the first three layers of the EfficientNet network and focused on training the last layers responsible for learning the unique characteristics of human motion. This approach resulted in improved performance while minimizing the risk of overfitting. The final model had six classes corresponding to the six distinct human motions in our dataset: AFR, ARO, LBE, LFC, SLL, and TRO. Figure 5 illustrates the categorization and training process outcomes, including the loss and accuracy functions' growth as a function of the number of epochs. Utilizing a pre-trained network with a clean dataset significantly improved learning speed and precision, demonstrating the effectiveness of the Super Object approach in studying human movement. In **Error!**

Reference source not found. confusion matrix, we summarized the categorization results, which demonstrate the feasibility and applicability of the Super Object method for assessing human movement. It is worth mentioning that, for the sake of simplicity, we only used the dataset of Czech pupils in this experiment. These students performed exercises in a controlled environment and laboratory settings, resulting in collecting clean and consistent data for demonstration purposes. This approach enabled us to demonstrate the Super Object technique and its potential for human movement analysis clearly and straightforwardly.

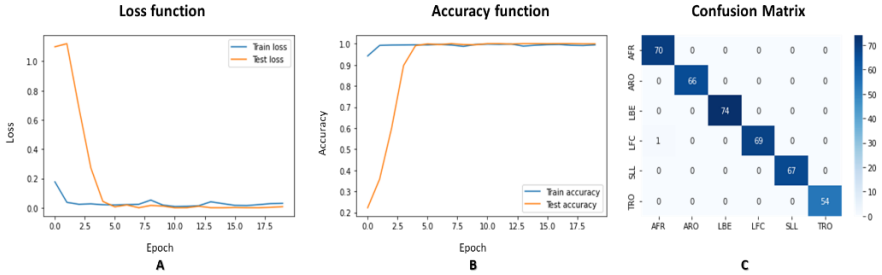


Fig. 5. *The Classifier results*

We trained the neural network with over of 100K frames of synthetic data. The goal of the Encoder is to compress the data and thus save the most important information and optimize the calculations. The goal of the decoder part is to update the weights of the NN. Therefore, after training the model the encoder model is saved, and the decoder is discarded.

7 The Classifier

The method of our classifier can be seen in **Fig. 6** :

- Training the autoencoder and save the encoder, as we explained before.
- Insert to the trained encoder the synthetic motion and all other predefined movements.
- Compare the encoded (compressed) synthetic motion with all other predefined encoded motions.
- The classifier returns the decision of which motion it thinks has entered.

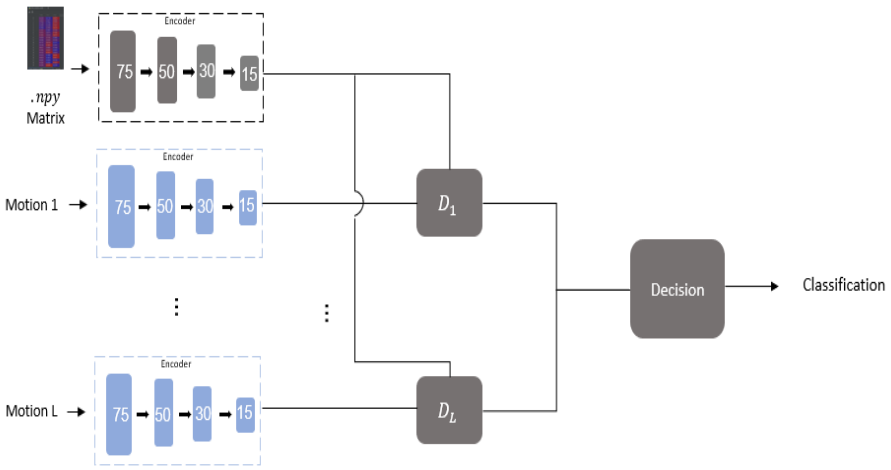


Fig. 6. *The Classifier structure*

7.1 L1 Distance

At first, after the motion is compressed (encoded), according to [1] we performed a Euclidean Distance measurement for each of the 6 predetermined motions, which are also compressed by a 15-vector. The distance formula :

$$D = \frac{1}{L}\sum_{i=1}^{n} |x_i - y_i| \quad (1)$$

x – The tested motion
y – Predetermined motion
i – One frame of the motion
n – Total number of frames
L – number of movements (in our case – 6)

The motion that was closest to the tested motion (minimum distance) was the motion to which it was classified. This is a naive classification method because it is limiting the number of frames of each motion to be the same length as predefined motions.

This is a big problem since the speeds of the movements vary from person to person and we also want to deal with different numbers of repetitions of the motions. Therefore, we change the decision formula to Dynamic Time Warping (DTW).

7.2 Dynamic Time Warping (DTW) distance

DTW is used to compare the similarity or calculate the distance between two arrays or time series with different lengths, shifts, and speeds. DTW compares the amplitude of the first signal at time T with the amplitude of the second signal at time T+1 and T-1 or T+2 and T-2. This makes sure it does not give a low similarity score for signals with similar shapes and different phases. The DTW calculation flow can be seen in **Fig. 7**.

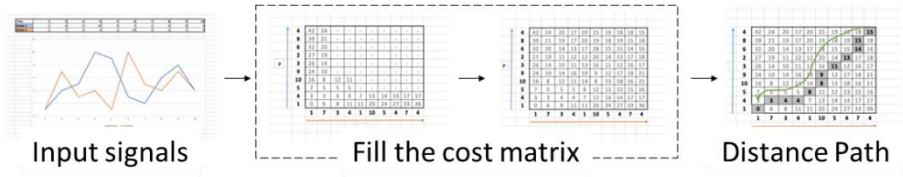


Fig. 7. DTW calculation flow

DTW algorithm:

1. Take two vectors, if one is longer than the other, it's padded with zeroes. Create an empty cost matrix.
2. Fill the cost matrix from left and bottom corner according to:

$$M(i,j) = |P(i) - Q(j)| + min\,(M(i-1,j-1),\ M(i,\ j-1),\ M(i-1,j))\ (2)$$

M – the cost matrix,
i for iterating P time series
j for iterating Q time series

3. Warping path from the top right corner of the matrix to the bottom left. The path is identified based on the minimum neighbor.
4. Calculate the distance:

$$D = \frac{\sum_{i=1}^{k} d(i)}{k}\ (3)$$

d – the warping path
k – the length of the warping path

For classified the synthetic motions as describe in **Fig. 8**, we compared the encoded parts of each predefined motion.

1. The encoder compressed each frame to a 15-vector.
2. The DTW compares the $0th$ place of the motion being performed (in all frames) to the $0th$ place of the predefined motion, and so on the $1,2,\ \ldots,\ 14th$ place.
3. Each one of these 15 vectors classifies to a predefined motion according to the minimum DTW value.
4. The selected motion is the one to which most vectors have been classified.

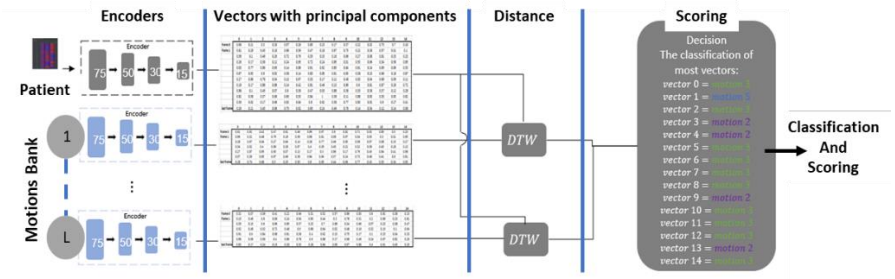


Fig. 8. The Autoencoder-DTW Classifier

8 Experimental Results

The runtime model takes a patient's video as input, which is first pre-processed by activating MediaPipe on the video, applying a normalization algorithm to rescale the skeletons and normalize the data to a range of [0,1], and then converting the patient's skeleton from MediaPipe skeleton format to OpenPose skeleton format using NN. To synchronize the patient's movements with the therapist's movements, the key points from the patient's skeleton that match the dominant key points of the therapist are selected and fed into a synchronization block. The synchronized movements are then converted into TSSI representation. The pre-trained Siamese NN, which consists of two symmetric CNNs that find the similarity matrix, takes the TSSI images of the patient and the therapist as input, and the DTW algorithm is applied on the similarity matrix. The results are then scored, and a score is returned. The neural network that we built for evaluated the quality of the simulator was used. We test 2800 $(400_{\,per\ motion} \times 7_{predifined\ motions})$ motions, the data differs in type, speed, number of repetitions, the noise of the movement. The confusion matrix visualizes the performance of the system. Each row of the matrix represents the actual class of the tested motion while each column represents the predicted class. The accuracy is calculated by:

$$\text{Accuracy} = \frac{\text{trace(matrix)}}{total\ \text{number of tested data}} = 91.82\%(4)$$

	0	1	2	3	4	5	6
0	400.00000	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000
1	0.00000	400.00000	0.00000	0.00000	0.00000	0.00000	0.00000
2	0.00000	0.00000	319.00000	0.00000	0.00000	0.00000	81.00000
3	7.00000	0.00000	0.00000	340.00000	17.00000	0.00000	36.00000
4	0.00000	0.00000	0.00000	0.00000	400.00000	0.00000	0.00000
5	0.00000	0.00000	0.00000	0.00000	88.00000	312.00000	0.00000
6	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000	400.00000

Fig. 9. *Confusion Matrix*

9 Discussion

Our research proposes a unique approach to collect anonymized patient data on limb movements during physical therapy exercises using a combination of OpenPose and MediaPipe skeleton extraction technologies. The proposed system enables us to score the quality of a patient's movement during physical therapy sessions in real time. We tested the system using a dataset of 2800 motions that differed in type, speed, number of repetitions, and the noise of the movement. The confusion matrix visualizes the system's performance, with each row representing the actual class of the tested motion while each column represents the predicted class. To process a patient's video and extract their skeletal data, we employed MediaPipe and OpenPose skeleton extraction technologies. We first pre-processed the video by activating MediaPipe and then applied a normalization algorithm to rescale the skeletons and normalize the data to a range of [0,1]. Next, using a neural network, we converted the patient's skeleton from the MediaPipe skeleton to the OpenPose skeleton. We then selected the key points from the patient's skeleton that matched the dominant key points of the therapist and fed them into a synchronization block to synchronize the patient's movements with the therapist's movements. The synchronized movements were then converted into TSSI representation, and a pre-trained Siamese neural network that consisted of two symmetric CNNs was used to find the similarity matrix between the patient and therapist TSSI images. Finally, we applied the Dynamic Time Warping (DTW) algorithm to the similarity matrix, scored the results, and returned a score. To evaluate the accuracy and quality of the patient's movements, we employed EfficientNet as our CNN, which is compact and suitable for implementation in regular mobile phones. We created a human gesture database and utilized EfficientNet to tag, measure, and infer human gestures. To enhance the performance of the database, we imitated patient movements and augmented them to address the lack of labeled physiotherapy exercise videos. We then grouped the skeletons into a single array and fed them to the CNN to obtain their low-dimensional vector representation, which was categorized using the empirical scoring technique to measure the patient's exercise compared to their physiotherapist. Our results demonstrate the effectiveness of the proposed system, achieving an accuracy of 91.8% when tested on a dataset of six different physiotherapy exercises. The study compares expert skeleton extraction using OpenPose with real-time patient skeleton extraction using MediaPipe, ensuring the system's effectiveness in a patient's home environment. We also employed a fully linked network as a MediaPipe to OpenPose conversion to address MediaPipe's inaccuracy in skeletal structure. In conclusion, the proposed system is a promising approach to collecting anonymized patient data on limb movements during physical therapy exercises in real time. By combining OpenPose and MediaPipe skeleton extraction technologies, we can score the quality of a patient's movements and evaluate their accuracy compared to their physiotherapist. This system can improve the quality of physical therapy exercises and could be implemented in regular mobile phones to improve patient care and support remote therapy sessions.

10 Conclusions

We have explored the potential augmentation of a sample of human movement data with synthetic data simulation and neural network classification for recognizing body movements. Our proposed system has successfully utilized a Pose Dictionary, created from a sequence of poses, to train a neural network for classification. To enhance the synthetic data, we developed a Synthetic Motion Simulator that uses input parameters to generate a .npy matrix and video file of the skeletal movement. We found that this technology has significant potential in various fields, such as physical therapy and sports training, where it can monitor and evaluate body movements, improve performance, and reduce the risk of injury. With the integration of AI and Deep Learning, the possibilities for this technology are limitless. We envision a future where synthetic data creation and neural network classification can be applied to many other areas, such as robotics and virtual reality, to name a few. Our work has highlighted the value of synthetic data when real-world data is unavailable and the potential for neural networks to classify body movements accurately. Our system could be further improved with more advanced neural networks and increased training examples. With this in mind, we remain enthusiastic and optimistic about the future of this technology, and we are excited to see the potential it holds in improving human performance and quality of life.

11 Further work

As researchers, we are always looking to improve and enhance our work. While our current research has been successful in creating a synthetic data solution for body movement classification and a Synthetic Motion Simulator, we recognize that there are limitations to our work. Therefore, in future work, we plan to validate our framework fully by focusing on rehabilitation exercises done by patients and labeled by a panel of doctors who will award a quality rating. We have outlined several steps to enhance the accuracy and effectiveness of our system. Our first action is to expand the dataset and increase the diversity of choreography by adding more poses to the simulator. Secondly, we aim to create a deep neural network that integrates DTW classification into the system. Lastly, we intend to improve the neural network's performance by incorporating an accuracy detection feature, which can classify more intricate movements. We believe that these actions will significantly improve our system's capabilities and make it a more valuable tool for physical therapy and sports training. By continuing to innovate and push the boundaries of what is possible with synthetic data and neural networks, we hope to create a more effective and accessible solution for evaluating and monitoring body movements.

12 Acknowledgment

This work was supported a grant from the Ministry of Science & Technology, Israel & The Ministry of Education, Youth and Sports of the Czech Republic.

13 References

[1] Y. Segal and O. Hadar, 'Interpolation of missing frames of human body movements via video motion vectors - OpenPose accelerator', Guntur, India, 2021.

[2] Z. Cao, G. Hidalgo, T. Simon, S. Wei, and Y. Sheikh, 'OpenPose: realtime multi-person 2D pose estimation using Part Affnity Fields', *arXiv preprint arXiv:1812.08008*, 2018.

[3] C. Lugaresi *et al.*, 'MediaPipe: A Framework for Perceiving and Processing Reality'.

[4] D. E. Rumelhart, G. E. Hinton, and R. J. Williams, 'Learning representations by back-propagating errors', *Nature*, vol. 323, no. 6088, pp. 533–536, Oct. 1986, doi: 10.1038/323533a0.

[5] G. Koch, 'Siamese Neural Networks for One-Shot Image Recognition', Graduate Department of Computer Science, Toronto, 2015. [Online]. Available: http://www.cs.toronto.edu/~gkoch/files/msc-thesis.pdf

[6] X. Cai, T. Xu, J. Yi, J. Huang, and S. Rajasekaran, 'DTWNet: a Dynamic Time-Warping Network', *Advances in neural information processing systems 32*, 2019.

[7] S. Ananthakrishnan, 'Gesture, Emotions, Posture and Face Recognition using OpenPose/DLIB', Oct. 2018. https://github.com/srianant/computer_vision

[8] Z. Cao, G. Hidalgo, T. Simon, S.-E. Wei, and Y. Sheikh, 'OpenPose: Realtime Multi-Person 2D Pose', *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 7291–7299, 2017.

[9] Y. Segal *et al.*, 'Camera Setup and OpenPose software without GPU for calibration and recording in telerehabilitation', in *IEEE E-Health and Bioengineering*, Lasi, Romania, 2021.

[10] Y. Bai, H. Hu, Y. Li, C. Zhao, L. Luo, and R. Wang, 'Research methods for human activity space based on vicon motion capture system.', *2017 5th International Conference on Enterprise Systems (ES). IEEE*, Sep. 2017.

[11] K. H. Cheung, 'Optitrack - Estimation of Opti-track motion capture system data', Department of Electrical Engineering, Mar. 2020. [Online]. Available: http://dspace.cityu.edu.hk/handle/2031/9335

[12] P. Malý and F. Lopot, 'QUALISYS System Applied to Industrial Testing', *AMM*, vol. 486, pp. 135–140, Dec. 2013, doi: 10.4028/www.scientific.net/AMM.486.135.

[13] 'SMART-DX EVO | Highest precision motion capture camera | BTS'. https://www.btsbioengineering.com/products/smart-dx-evo/ (accessed Feb. 26, 2023).

[14] Z. Zhang, 'Microsoft kinect sensor and its effect.', *IEEE MultiMedia*, p. 19(2):4–10, Feb. 2012.

[15] 'Home - xsens 3d motion tracking', *https://www.xsens.com/*, Mar. 2020. www.xsens.com

[16] 'Rokoko - motion capture system - smartsuit pro', Mar. 2020. https://www.rokoko.com/en/

[17] H.-S. Fang, S. Xie, Y.-W. Tai, and C. Lu, 'RMPE: Regional Multi-Person Pose Estimation', presented at the Proceedings of the IEEE International Conference on Computer Vision, 2017, pp. 2334–2343. Accessed: Feb. 26, 2023. [Online]. Available:

https://openaccess.thecvf.com/content_iccv_2017/html/Fang_RMPE_Regional_Multi-Person_ICCV_2017_paper.html

[18] 'Real-time Human Pose Estimation in the Browser with TensorFlow.js'. https://blog.tensorflow.org/2018/05/real-time-human-pose-estimation-in.html (accessed Feb. 26, 2023).

[19] 'Python OpenCV - Pose Estimation', *GeeksforGeeks*, Jul. 14, 2021. https://www.geeksforgeeks.org/python-opencv-pose-estimation/ (accessed Feb. 26, 2023).

[20] 'open-mmlab/mmpose: OpenMMLab Pose Estimation Toolbox and Benchmark.', *GitHub*. https://github.com/open-mmlab/mmpose (accessed Feb. 26, 2023).

[21] V. Bazarevsky, I. Grishchenko, K. Raveendran, T. Zhu, F. Zhang, and M. Grundmann, 'BlazePose: On-device Real-time Body Pose tracking', *arXiv.org*, Jun. 17, 2020. https://arxiv.org/abs/2006.10204v1 (accessed Feb. 26, 2023).

[22] 'HRNet/HRNet-Human-Pose-Estimation: This repo is copied from https://github.com/leoxiaobin/deep-high-resolution-net.pytorch', *GitHub*. https://github.com/HRNet/HRNet-Human-Pose-Estimation (accessed Feb. 26, 2023).

[23] J. Redmon, S. Divvala, R. Girshick, and A. Farhadi, 'You Only Look Once: Unified, Real-Time Object Detection', presented at the Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, 2016, pp. 779–788. Accessed: Feb. 26, 2023. [Online]. Available: https://www.cv-foundation.org/openaccess/content_cvpr_2016/html/Redmon_You_Only_Look_CVPR_2016_paper.html

[24] Z. Wang, C. Liu, and L. Ma, 'LandmarkNet: a 2D digital radiograph landmark estimator for registration', *BMC Med Inform Decis Mak*, vol. 20, no. 1, p. 168, Dec. 2020, doi: 10.1186/s12911-020-01164-4.

[25] K. He, X. Zhang, S. Ren, and J. Sun, 'Deep Residual Learning for Image Recognition', Dec. 2015, doi: 10.48550/arXiv.1512.03385.

Deep Learning based Cryptanalysis on SLIM Cipher

Vignesh Rajakumar[1][0009−0000−7899−5392], Lakshmy KV[2][0000−0001−5344−2855], Chungath Srinivasan[3][0000−−0002−7027−4926], and M. Sethumadhavan[4][0000−0001−5476−5461]

TIFAC-CORE in Cyber Security
Amrita School of Computing, Coimbatore
Amrita Vishwa Vidyapeetham, INDIA

Abstract. This paper provides an in-depth exploration of the application and effectiveness of neural distinguishers in cryptanalysis. Focusing on the SLIM Cipher and building upon the work of Gohr, we aim to explain why neural distinguishers outperform traditional differential attacks and enhance contemporary methodologies. Our experiments centered on the 5^{th} round of the SLIM Cipher, and aimed to understand the influence of input difference, output difference, and previous round differentials on the accuracy of the neural distinguisher. Our investigations revealed that the choice of input difference was significant, and a singular focus on maximizing the differential probability might not always yield the best results. Neural distinguisher seems to observe not just about the distribution of differences from the ciphertext was key insight that was noticed, but also additional cryptanalytic features or properties. Furthermore, we found that the output difference in the ciphertext played a critical role in the predictions made by the neural distinguisher. By partially decrypting ciphertext pairs, we discovered that the output difference from the penultimate round was a major factor impacting the accuracy of the neural distinguisher. This insight led to an improved methodology, yielding impressive accuracy results. This paper we try to shed light on the complex workings of neural distinguishers and presents valuable insights that can guide the future development and application of these tools in cryptanalysis.

Keywords: Deep Learning · Differential Cryptanalysis · Cryptography · SLIM Cipher · Neural Distinguisher · Machine Learning.

1 Introduction

Nowadays symmetric key cryptography puts a significant emphasis on ensuring maximum security through well planned designs along with powerful arguments which are centered around factors such as immunity against differential/linear attacks and algebraic property analysis amongst others. Cryptanalysis plays an essential role in verifying whether or not a cipher can be deemed secure, so that only ciphers which have undergone comprehensive examination by experts

are trusted as being secure enough for use. However recently lightweight cryptography has seen an increase in proposals which has created a challenge for comprehensive cryptanalysis.

Cryptanalysts focus on creating an effective model for the issue. In the past decade, there have been significant advances in this area which has resulted in improved designs in cipher. Given the shortage of cryptanalysts, there is an increasing trend towards automating the attacker's tasks. This can be oberserved in trying to find differential and linear properties which are now modelled using Satisfiability/Satisfiability Modulo Theories [1] , Mixed Linear Integer Programming [2] or Constraint Programming [3]. Specialized solvers can solve these models reducing cryptanalysts to concentrate on efficient problem modeling. It is an ongoing study that whether a tool that recognizes weaknesses or patterns in ciphers can be developed which merely requires interaction with the cipher and minimal input from cryptanalysts.

The integration of machine learning and deep neural networks into cryptography has been a focus for researchers, aiming to optimize the outcomes. A significant part of this effort involves the creation of advanced neural distinguishers, capable of differentiating between ciphertexts produced by different encryption algorithms. In Gohr's Neural Distinguisher model, presented at the 2019 CRYPTO conference, emphasizes this effort as a unique neural distinguisher focused for lightweight cryptographic algorithms. Gohr's research further advanced the field by training the deep neural network on ciphertext pairs, half of which were obtained from plaintext pairs which has a fixed input difference and half of them are random. His trained network showed promising accuracy in classifying these pairs, notably when cracking the SPECK-32/64 block cipher [4]. By layering a key recovery method onto his neural distinguisher, Gohr developed an advanced key recovery attack that exceeded the performance of previous research on SPECK-32/64, such as [5], [6].

As reported In June 2019, an incident involving a leading Internet of Things (IoT) software management company found the importance of data privacy in the creation of IoT or smart devices. The company was discovered to have stored 2 billion records of user data, pulled from customer's home appliances, in an unsecured database. This event demonstrates why data privacy should always be a top priority when creating IoT or smart devices. Therefore, there is a need for effective solution for lightweight cryptography which require less computation.

The term "lightweight" doesn't have any definition. However, it typically refers to algorithms with smaller block and key sizes or that require less energy [7]. Relatively new block cipher was proposed named as SLIM[8].

SLIM is a 32-bit block cipher with an 80-bit key that was created as an incredibly light block cipher for devices with limited resources. It is a secure and effective solution for applications with constrained memory and processing power is what SLIM aims to deliver. A 16-bit subkey generated from the master key is used in each of the 32 rounds of the cipher's encryption and decryption operations, which are based on a Feistel-like structure.

In this , we focus on performing a broad cryptanalysis of the SLIM Cipher based on the Crypto 19 Gohr Neural Distinguisher model. Our investigation will cover the design and implementation of neural distinguishers, the methodology used to evaluate the security of the Gohr's Neural Distinguisher model, and a detailed analysis of the results obtained from the neural distinguisher. Additionally, we will explore potential improvements and modifications to the Gohr's Neural Distinguisher model and the SLIM cipher in light of our findings to enhance their security in real-world applications.

Our Contributions. In this article, we analyze how Gohr's Neural Distinguishers function when used with the SLIM-32/80 cipher (the 32-bit block 80-bit key version of SLIM). Our main objective is to gain an understanding of their decision-making processes by thoroughly analyzing sets of real and random ciphertext pairs. The chances for recognition by these networks increases as certain differences occur with greater frequency - a key observation made during our study. After looking for patterns closely, we can infer that differential conditions are being deduced not just from ciphertext pairs but also from penultimate rounds by these neural distinguishers. We propose a hypothesis that is tested through our experimental research.

2 Preliminaries

2.1 Specification of SLIM

SLIM cipher is a symmetric encryption algorithm specifically designed for devices, such as RFID tags. It features a 32-bit block similar to Feistel structure block design with 32 rounds, which has 32-bit plaintext and ciphertext encrypted using an 80-bit key. The blocks are divided as two halves of 16-bit, denoted as L_i and R_i, where *'i'* represents the ith round of the encryption or decryption process. SLIM incorporates four 4-bit S-boxes (originally adopted from the PRESENT cipher [9]) and a bitwise permutation box (P-box). The algorithm focuses on security and simplicity, achieving confusion and diffusion concepts. Simplicity is attained through the compact size of the S-box and the use of simple internal operations.SLIM employs each 16-bit generated from the 80-bit key, for a total of 32 sub-keys. The processes of encryption and decryption employ the same key, the only distinction being that the order of applying sub-keys during decryption is reversed compared to encryption. The fundamental architecture of the SLIM algorithm is shown in figure 1 [8].

Single Round Processing: Understanding SLIM requires observation of its layout in a single round. Firstly, divide the 32-bit input into two equal parts referred to as L_i and R_i. Next, through an XOR operation between R_i and sub-key K_i, we get an outcome that passes through a substitution box before forwarding it to a permutation process. Finally, this output undergoes another XOR with L_i - becoming inputs for upcoming rounds, while R_i switches over to being left half input for next rounds. The processing at each round can be

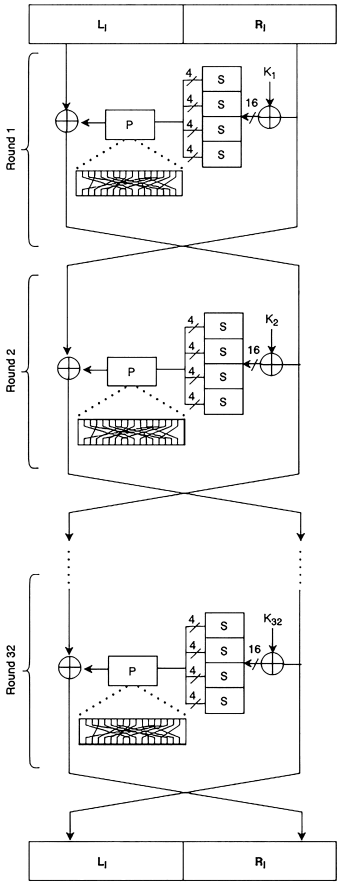


Fig. 1: SLIM encryption

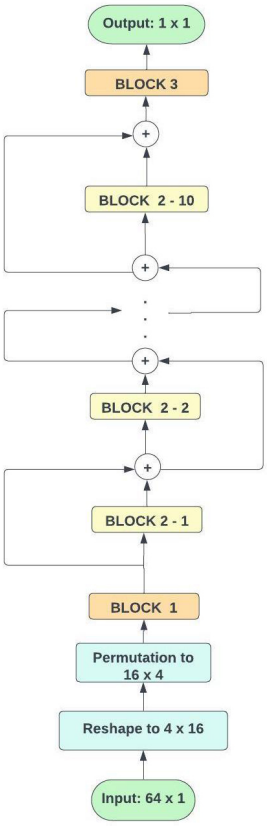


Fig. 2: Architecture of the Neural Distinguisher

represented by Equations 1 and 2.

$$L_i = R_{i-1} \tag{1}$$

$$R_i = L_{i-1} \oplus P(S(K_i \oplus R_{i-1})) \tag{2}$$

Substitution Layer: SLIM employs an S-box (refer to Table 1) that is applied four times concurrently. For strong S-box that prevents linear and differential attack and low area footprint in S-box, PRESENT S-box is chosen [9].

Permutation Layer: The P-box (refer to Table 2) of SLIM was designed to have no fixed point (every input bit is moved away from its original position).[10]

Key Generation: SLIM has a complex key schedule for 32 rounds and a block of 32-bit, it requires 32 sub-keys of 16-bit [8]. The generation is as follows:

x	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
S(x)	C	5	6	B	9	0	A	D	3	E	F	8	4	7	1	2

Table 1: PRESENT's S-box used in SLIM [8].

x	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16
P(x)	8	14	2	9	12	15	3	6	5	11	16	1	4	7	10	13

Table 2: SLIM P-box [8].

The first five sub-keys are generated directly from the 80-bit key as K_1,K_2 till K_5, which is then split into two 40-bit parts, KeyMSB and KeyLSB, for separate processing. K_1 is first (least significant) and K_2 is next, and so on. Each subkey is 16 bits.

2.2 Differential Cryptanalysis

Differential cryptanalysis is frequently used by cryptographers as a reliable method to evaluate algorithm security. This procedure involves carefully examining input-output pair differences. We require a thorough literature review that covers the fundamentals of applying differential cryptanalysis for assessing block ciphers and earlier work using it on the SLIM cipher in order to make sure that this study fully covers all of the bases.

2.3 Deep Neural Network

The main goal of differential cryptanalysis involves identifying specific differential characteristics that exhibit higher than expected probabilities than random processes. Deep Neural Networks (DNNs) has changed machine learning and artificial intelligence. Their structure is based on multiple interconnected neuron layers that model complex patterns found in high dimensional data sets. DNN aims to find optimal parameters for a given dataset and depend on optimization algorithms like stochastic gradient descent, where hyper-parameters play a crucial role in controlling the learning process. Training data empowers deep neural networks (DNNs) in establishing accurate non-linear features. However, these characteristics are not robust meaning small input noise may confuse the model. Key building blocks of DNNs include linear neural networks, one-dimensional convolutional neural networks, activation functions, and batch normalization. The depth of these networks also plays an essential role in their capacity for deep learning by facilitating the identification of complex relationships which are not feasible using traditional methods. By using back propagation algorithms throughout the training process they continuously refine neuron

connections while minimizing differences between predicted and actual outputs. With its reliance on significant amounts of data and computational resources, the process involved in training DNNs poses formidable obstacles from those lacking these necessities.

2.4 Neural Distinguisher

Researchers are focusing on creating "neural distinguishers" as one potential means of enhancing cryptanalysis procedures while working with encrypted data. These strategies analyze data using artificial intelligence and deep learning techniques, and the preliminary results are encouraging. To learn more about this expanding topic, completing a full literature survey is important with a specific focus on examining how neural distinguishers have been developed and employed in cryptography to analyze block ciphers.

2.5 Gohr's CRYPTO 2019

Deep Learning has recently driven remarkable advancements in various complex areas. These include tasks like language translation, enabling autonomous vehicles, and mastering complex board games at a level surpassing human abilities. In the field of cryptography, the practical application of machine learning techniques has been mainly targeted at side-channel analysis.

In Crypto 19, Gohr published a Deep Learning based cryptanalysis work on SPECK-32/64 [11].

Overview: The Gohr's Neural Distinguisher model, introduced at the CRYPTO 19 conference, represents concepts together of machine learning and conventional cryptanalysis methodologies. Utilizing a Convolutional Neural Network (CNN), a form of deep learning, it creates a neural distinguisher trained to differentiate pairs of ciphertext that originated from plaintext pairs following a specific input difference from random pairs. The deep structure and non-linear activation functions of the CNN allow the neural distinguisher to identify complex patterns and dependencies in the data. When applied to the SPECK cipher, this novel approach to differential cryptanalysis primarily distinguishes between real pairs of plaintext with a fixed input difference (0x0040/0000) and random pairs after encryption. Remarkably, the Deep Neural Network (DNN) based distinguisher outperforms traditional methods when tested on 5 to 8 rounds of SPECK-32/64.

Gohr's Neural Distinguisher: Figure 2 presents Gohr's Neural Distinguisher, a deep neural network comprised of several key components:

- Block 1 includes a 1D-Convolutional Neural Network with a kernel size of 1, followed by a batch normalization process and a Rectified Linear Unit activation function.

– Blocks 2-i contain between one to ten layers, each layer having two 1D-CNNs with a kernel size of 3, with batch normalization and a ReLU activation function.
– Block 3, includes as a non-linear classification block, composed of three perceptron layers separated by two batch normalization and ReLU functions, finishing with a sigmoid function [12].

The encryption process yields a 64×1 matrix of binary values derived from the ciphertext pair, which is then fed as input to the model. This matrix is subsequently reshaped into a 4×16 form. Each row of this matrix represents a 16-bit value, arranged in the sequence: C_l, C_r, C_{0l}, C_{0r}. C_l represent 16 bits of left and C_r represent right 16 bits of the first ciphertext and C_{0l} represent 16 bits of left and C_{0r} right 16 bits of the second ciphertext. This is then followed by a permutation operation that reorganizes it into a 16×4 structure.

In the first convolutional block (Block 1) as shown in 3, the input is the structured 16×4 matrix. This input is then processed through a convolutional layer equipped with 32 filters. The kernel size for this 1D Convolutional Neural Network (1D-CNN) is set to 1. The 1D CNN transformation applies to (C_l, C_r, C_{0l}, C_{0r}), resulting in a new representation as filters (filter1, filter2, until 32nd filter) .

Each filter represents a combination of the features (C_l, C_r, C_{0l}, C_{0r}), which is determined by the input values and weights derived by the 1D-CNN after applying the ReLU activation function.

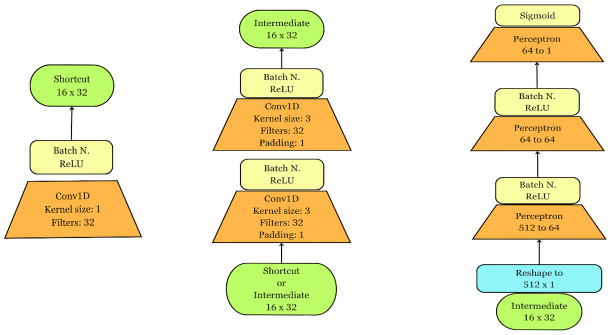


Fig. 3: Representation of Independent Blocks

The residual blocks of 10 layers - Blocks 2-i as shown in 3 , each have two 1D-CNNs with a kernel size of 3, padding size of 1, and stride size of 1. These parameters ensure that the length (or size) of the input data is preserved across layers. In the residual block the output from the initial block is linked with the input and is merged with the output from the subsequent layer and conclusion of each layer, the input signals are not wiped. The final output from a residual block is in the form of a feature tensor with dimensions 16×32.

The final classification block as shown in 3, receives the flattened output tensor from the previous block as its input. This input, which is a 512×1 vector, is then processed through three layers of a Multi-Layer Perceptron (MLP). Each of the first two layers in this MLP utilizes batch normalization and a ReLU activation function. The final layer, however, applies a sigmoid activation function to perform binary classification. The results from the Gohr's model for the 5^{th} round, the N_5 distinguisher achieved an accuracy of 0.929 ($\pm$ 5.13 $\times 10^{-4}$) with a true positive rate of 0.904 ($\pm$ 8.33 $\times 10^{-4}$) and a true negative rate of 0.954 ($\pm$ 5.91 $\times 10^{-4}$) [11].

2.6 One-dimensional Convolutional Neural Network

The 1D-CNN performs a convolution operation on a constant multi-temporal input signal [12]. The 1D-CNN can be viewed as several linear neural networks, each corresponding to a filter, working on different sub-parts of the input, which is called the kernel, slides along the input with a specific stride. The starting and ending points of this stride are defined by the padding..

2.7 Activation Layer

A neural network's activation layer is responsible for applying a non-linear function to the output of the previous layer. This non-linearity allows the network to understand complex correlations between model complex decision boundaries and input features. Activation functions like ReLU and Sigmoid has a important role in the learning capability and overall performance of the neural network. ReLU is a simple activation function that outputs the input value if it's positive and zero if it's negative. This non-linear function is computationally efficient and helps to mitigate the vanishing gradient problem during back-propagation. Sigmoid, on other hand, maps any input value to a range between 0 and 1. It is often used for binary classification problems in the output layer.

3 Methodology

The paper [12] concentrates into the workings of neural distinguishers in cryptanalysis, with a particular focus on why Gohr's models outperform traditional differential attacks and enhance the state-of-the-art methodologies. By conducting a series of experiments and analysis, researchers attempted to unpack the high performance of these distinguishers on multiple rounds of SPECK-32/64. In the paper [11] they have claimed that the difference 0x0040/0000 has been chosen based on its ability to deterministically transit to a difference with a low Hamming weight after one round. One significant insight was the identification of the input difference as a important factor in the performance of neural distinguishers. When the input difference was altered, and the distinguishers were only given access to the difference distribution, their performance declined. Moreover, the distinguishers, when trained only on the difference between two

ciphertexts, didn't perform as well as when provided with the actual ciphertext pairs. This implies that Gohr's Neural Distinguishers also potentially exploit additional cryptanalytic features or properties.

In our effort to understand the workings of the neural distinguisher, we have carried out a few experiments on SLIM Cipher in a cryptanalysis perspective. Our analysis revolves around the neural distinguisher's behavior on the 5^{th} round of the slim cipher. Initially we begin with an experiment to validate our hypothesis that maximizing differential probability when selecting a differential characteristic might not always be the ideal approach. We're seeking to determine if this strategy always holds true, or if there might be cases where other strategies could yield better results.Then the experiment focuses on the neural distinguisher's output difference. We'll be examining the significance of this output difference and how it influences the overall function and efficiency of the neural distinguisher. By understanding the role and importance of output difference, we can gain a deeper understanding of how the neural distinguisher operates. Then we try to understand the effect of previous rounds on the accuracy of the 5^{th} round. We're specifically interested in how the output differences from the penultimate rounds may impact the overall accuracy. We assume that understanding these dynamics can offer us key insights into the mechanisms of the neural distinguisher.

In the process, we're also planning to explore the significance of the frequency of the output difference, as this may offer another layer of understanding of how the neural distinguisher operates and performs.

3.1 Analyzing Input Difference

Based on the differential trial obtained in the paper [12], we have identified that the input difference with high probability in the 5^{th} round of SLIM cipher is $D804/0040$ with probability of 2^{-8}. To understand the importance of difference distribution we perform (Experiment A). With 10^6 ciphertext pairs with 100 set of keys and the input difference as $D804/0040$ we trained the neural distinguisher. Then conducted multiple test cases to understand that similar to speck cipher, SLIM cipher performed better with $0040/0$ compared to $D804/0040$ as shown in table 3. So this adds to the understanding that limiting to utilizing only the difference distribution, wasn't the best strategy. In the course of the experiment, it was also discovered that a neural distinguisher trained on a plaintext of size 10^6 encrypted with merely 10 or 100 keys could predict with the same accuracy that of a dataset of 10^6, each encrypted with a unique random key 4.

3.2 Analyzing Importance of Output Difference

Continuing with the analysis of the cipher through the perspective of the neural distinguisher, we recall the structure of the model. Since final layer is a sigmoid activation function, which produces outputs ranging from 0 to 1. Based on the output of this function, the neural distinguisher's predictions. For instance, an

Difference	Rounds	Epoch	Depth	Accuracy	TPR	TNR
0040/0	5	200	10	$0.814 \pm 1.02 \times 10^{-3}$	$0.829 \pm 1.28 \times 10^{-3}$	$0.800 \pm 1.29 \times 10^{-3}$
D804/0040	5	200	10	$0.607 \pm 3.74 \times 10^{-3}$	$0.501 \pm 5.53 \times 10^{-3}$	$0.706 \pm 5.36 \times 10^{-3}$

Table 3: Comparison in Input Difference

Difference	Rounds	Number of Keys	Dataset Size	Accuracy
0040/0	5	10	10^6	$0.799 \pm 7.25 \times 10^{-3}$
0040/0	5	100	10^6	$0.814 \pm 1.02 \times 10^{-3}$
0040/0	5	Random Keys (10^6)	10^6	$0.804 \pm 4.10 \times 10^{-3}$

Table 4: Comparison in keys regards to Input Difference

output score of 0.5 or higher leads the neural distinguisher to categorize the data as a 'real pair'. Conversely, an output score that falls below 0.5 prompts the neural distinguisher to label the data as a 'random pair'. For understanding neural distinguisher we divide the ciphertext pairs based on extreme scores. These include scores greater than 0.9 is categorized as 'good scores'. As the plaintext pair, we are using a fixed input difference of 0x0040/0 and believe that features learned by neural distinguishers are related to the difference, So we concentrate on differentials of the ciphertext pairs. To do this we perform next experiment (Experiment B):

1. With 10^6 5-round SLIM we create 5 lakh real ciphertext pairs, extract the set Good. We obtained around 1 lakh Good set.
2. Then we calculate the output difference of the cipher text and sorting based on their occurrence frequency.
3. For each distinct differential value δ:
   (a) Generate 10^4 distinct 32-bit random numbers and apply the difference δ to get 10^4 different numbers with the difference.
   (b) Input the pairs with δ into the 5-round neural distinguisher to generate the scores.
   (c) Note total count of pairs with a score ≥ 0.5.

We executed this experiment for the top 1000 differences δ with high frequency. In this process, we found out that the neural distinguisher could predict the values based on the output difference of SLIM cipher where out of 1000 almost 99% or 100% pairs have a score ≥ 0.5 and above 95% of the pairs have a score ≥ 0.9. This demonstrates that the difference in output is a vital factor for the neural distinguisher's prediction ability in terms of ciphers.It also show some indications that a neural distinguisher is more likely to identify a difference if it is more probable. Table 5 shows the top 25 difference and accuracy with high frequency.

No	Difference	Frequency	Accuracy
1	0006/2151	17	100%
2	0006/2153	17	100%
3	6006/311B	15	100%
4	0900/80E6	11	98%
5	2006/2359	11	98%
6	6006/3319	11	100%
7	6006/331B	11	100%
8	6006/3119	9	99%
9	0006/01D1	8	100%
10	0100/04F6	8	97%
11	0802/0440	8	100%
12	0900/04E6	8	97%
13	0900/84E7	8	93%
14	2000/02EE	8	100%
15	4006/235B	8	100%
16	6002/1019	8	97%
17	6006/309B	8	99%
18	6006/3299	8	99%
19	0100/00F7	7	97%
20	0100/84F6	7	100%
21	6006/3353	7	100%
22	0006/20D3	6	100%
23	0008/51A0	6	99%
24	0014/0D72	6	100%
25	00A0/0055	6	94%

Table 5: Difference Frequencies and Accuracies

3.3 Impact of Penultimate round

We further explore deeper than simply with the differences at the 5^{th} round. Our approach involved partially decrypting the ciphertext pairs from "Good" set for several rounds and then testing on these partially decrypted pairs. We looked at the difference of round 4 (after decrypting 1 round respectively). We analyzed the 5 round SLIM cipher with output difference of the 4^{th} round in this experiment (Experiment C).

1. For every pair of ciphertexts in the set G, we decrypt i rounds utilizing their corresponding keys. Then, calculate the corresponding difference. We can then name these differences as δ_{5-i}.
2. Now we encrypt another 10^5 plaintext with a difference of `0x0040/0` with 100 random keys. Then we decrypt them for one round and check if the pairs of differences. If the pair's of differences is in δ_{5-i}, we keep the pair else we discard.
3. The non discarded pairs are encrypted and tested using the neural distinguisher.

For the 4^{th} round, using $i = 1$, we generated 210438 distinct differences, resulting in a total dataset size of 10^6. Upon testing this dataset, which also consisted of 10^6 entries, we refined it using the output difference of round 4, leading to a refined dataset size of 250623. This refined approach allowed us to achieve remarkable precision, with 99.99% of these ciphertext pairs scoring above 0.5, and 99% of those being true positive pairs.

3.4 Impact of Frequency in Output Difference

To further understand the frequency and distribution of output differences in ciphertext pairs resulting from five-round and six-round encryptions and their implications for neural distinguishers and cryptanalysis, we perform an experiment.

In Experiment D, we follow these steps:

- We generate a dataset comprising 10^6 entries, divided equally between random values and values encrypted using the Slim cipher with a fixed difference. This dataset represents the output of both five-round and six-round encryptions.
- From this dataset, we isolate only the 'real' ciphertext pairs, i.e., those pairs originating from plaintexts with a fixed input difference.
- For both the five-round and six-round encryption data, we calculate the output difference for each ciphertext pair. This can be done using the XOR operation between the two ciphertexts in a pair.
- We then determine the frequency of each output difference in the ciphertext pairs across both five-round and six-round encryption datasets.

We trained a neural distinguisher with Round 6 ciphertext pair and we got an accuracy of 51%. The highest frequency in round 5 is 18, whereas the highest frequency in round 6 is just 2. So the difference that is being repeated is very less. From this, we can consider that the frequency of output difference could have a major impact on the learning of the neural distinguisher. A significant difference in the frequency of the output difference between two datasets could indicate a difference in the underlying patterns or structures within the data. The file with the higher frequency of a particular output difference (Round 5) might suggest that the corresponding input difference is more likely to occur under the cipher's operations. If certain output differences occur more frequently, the neural distinguisher will likely learn to recognize these patterns, which can improve its ability to differentiate between different types of data. From this, we can say high frequency of output differences in a dataset may enhance the neural distinguisher's learning capabilities by highlighting recurrent patterns. This can lead to improved differentiation between various data types and bolster the overall cryptanalysis process.

4 Results

Our experimental investigation on the application of neural distinguishers in cryptanalysis yielded significant results.

1. We found that the choice of input difference played a major role in the performance of neural distinguishers. In the first set of experiments (Experiment A), we observed that using an input difference of `0x0040/0` led to an accuracy of 0.814, compared to `D804/0040`. This result suggests that the choice of input difference should not be restricted to those with high probabilities in the differential distribution.
2. We also investigated the effect of the number of keys used for encryption. We found that a neural distinguisher trained with a plaintext of size 10^6 encrypted with merely 10 or 100 keys could predict with similar accuracy to a dataset of 10^6, each encrypted with a unique random key.
3. Experiment B indicated the strong impact of output differences on the neural distinguisher's prediction ability. Specifically, we found that among the top 1000 output differences with high frequencies, nearly all had a score of $\geq$ 0.5, suggesting that the output difference was a critical factor for the neural distinguisher's prediction ability.
4. We also observed the potential influence of prior rounds on the accuracy of the neural distinguisher. In Experiment C, partially decrypting the ciphertext pairs and considering the output difference from the penultimate round significantly improved the accuracy. We obtained remarkable precision, with 99.99% of the tested ciphertext pairs scoring above 0.5, and 99% of those being true positive pairs.
5. Experiment D demonstrated that a higher frequency of output differences in a dataset, as seen in the 5-round encrypted data, can significantly enhance the learning and accuracy of the neural distinguisher, underscoring the relevance of output difference frequency in training neural networks for cryptanalysis tasks.

Therefore, these results deepen our understanding of neural distinguishers operation and enhance their implementation in cryptanalysis. By considering factors such as the input and output differences and the effects of previous rounds, we can significantly improve the performance and accuracy of neural distinguishers.

5 Conclusion

Our research was aimed to increase our understanding of the role and behavior of neural distinguishers in cryptanalysis, focusing primarily on the SLIM cipher. We discovered that input difference, output difference, and the decryption of previous rounds play significant roles in the distinguisher's performance. Rather than merely choosing input differences with high probabilities in the differential distribution, diversifying these can strengthen the neural distinguisher's performance. We found evidence that a higher frequency of output differences directly correlates with higher accuracy, and decrypting even a single round can yield valuable attributes, thereby enhancing accuracy. Interestingly, the number of keys utilized for encryption did not significantly influence the accuracy, rendering the training of neural distinguishers more efficient on SLIM cipher. These

findings suggest promising applications of neural distinguishers in cryptanalysis, encouraging further research for an in-depth understanding of these

References

1. N. Mouha and B. Preneel, "A proof that the arx cipher salsa20 is secure against differential cryptanalysis," IACR Cryptol. ePrint Arch., vol. 2013, p. 328, 2013.
2. N. Mouha, Q. Wang, D. Gu, and B. Preneel, "Differential and linear cryptanalysis using mixed-integer linear programming." in Inscrypt, ser. Lecture Notes in Computer Science, C. Wu, M. Yung, and D. Lin, Eds., vol. 7537. Springer, 2011, pp. 57–76. [Online]. Available: http://dblp.uni-trier.de/db/conf/cisc/inscrypt2011.html#MouhaWGP11
3. S. Sun, D. Gérault, P. Lafourcade, Q. Yang, Y. Todo, K. Qiao, and L. Hu, "Analysis of AES, SKINNY, and others with constraint programming." IACR Trans. Symmetric Cryptol., vol. 2017, no. 1, pp. 281–306, 2017. [Online]. Available: `http://dblp.uni-trier.de/db/journals/tosc/tosc2017.html#SunGLYTQH17`
4. R. Beaulieu, D. Shors, J. Smith, S. Treatman-Clark, B. Weeks, and L. Wingers, "The simon and speck families of lightweight block ciphers," Cryptology ePrint Archive, Paper 2013/404, 2013, https://eprint.iacr.org/2013/404. [Online]. Available: https://eprint.iacr. org/2013/404
5. I. Dinur, "Improved differential cryptanalysis of round-reduced speck," 08 2014.
6. L. Song, Z. Huang, and Q. Yang, "Automatic differential analysis of arx block ciphers with application to speck and lea," vol. 9723, 07 2016, pp. 379–394.
7. J. S. Teh, L. J. Tham, N. Jamil, and W.-S. Yap, "New differential cryptanalysis results for the lightweight block cipher boron," Journal of Information Security and Applications, vol. 66, p. 103129, 2022. [Online]. Available: https://www.sciencedirect.com/science/article/pii/ S2214212622000205
8. B. Aboshosha, R. Ramadan, A. Dwivedi, A. El-Sayed, and M. Dessouky, "Slim: A lightweight block cipher for internet of health things," IEEE Access, vol. 8, pp. 2169–3536, 11 2020.
9. A. Bogdanov, L. Knudsen, G. Leander, C. Paar, A. Poschmann, M. Rob- shaw, Y. Seurin, and C. Vikkelsoe, "Present: an ultra-lightweight block cipher," vol. 4727, 09 2007, pp. 450–466.
10. Y. Y. Chan, C.-Y. Khor, J. S. Teh, W. Teng, and N. Jamil, Differential Cryptanalysis of Lightweight Block Ciphers SLIM and LCB, 01 2023, pp. 55–67.
11. A. Gohr, Improving Attacks on Round-Reduced Speck32/64 Using Deep Learning, 08 2019, pp. 150–179.
12. A. Benamira, D. Gerault, T. Peyrin, and Q. Q. Tan, "A deeper look at machine learning-based cryptanalysis," Cryptology ePrint Archive, Paper 2021/287, 2021, https://eprint.iacr.org/2021/287. [Online]. Available: https://eprint.iacr.org/2021/287 7

Beyond the Average: Unpacking the Distributions of Time to Weaponization and Exploitation

Miguel A. S. Bicudo[1], Daniel S. Menasché[1], and Anton Kocheturov[2]

[1] Institute of Computing, Federal University of Rio de Janeiro, Rio de Janeiro, Brazil
[2] Siemens Corporation, Princeton, NJ, USA

Abstract. In this paper we aim at predicting the distribution of time to weaponization of vulnerabilities. We train an XGBoost model with events related to the life-cycle of vulnerabilities to make improvements of the predicted distributions over time, as more data becomes available. The distributions are then compared with other methods of predicting exploits, such as EPSS.

Keywords: vulnerability · exploit · prediction · XGBoost.

1 Introduction

Assessing the risk posed by system vulnerabilities is crucial for determining the order in which they should be patched, as the process of patching systems can be time-consuming and may involve maintenance costs [1, 5]. This research paper focuses on the time it takes for a vulnerability to be weaponized and how its distribution changes over time. These factors can serve as indicators that the risk associated with the vulnerability is increasing or decreasing. Unlike previous studies, our approach aims to incorporate dynamic information about vulnerabilities, including data from advisories, online forums, and other sources that publish information after it becomes available through the National Vulnerability Database (NVD).

Other studies have tackled this challenge from alternative perspectives, such as Expected Exploitability (EE) [4] and the Exploit Prediction Scoring System [2, 3]. EE provides the probability of a functional exploit being developed based on metrics related to exploit code and social media, in addition to vulnerability data. Similarly, the Exploit Prediction Scoring System predicts the likelihood of exploitation in the wild. The current version (EPSS v3) relies on proprietary features and observations indicating activity related to the vulnerability. Both EE and EPSS utilize dynamic vulnerability data. However, they do not consider the distribution of time until weaponization.

Gaps in prior work. EPSS provides the probability of exploitation occurring in a fixed window size of 30 days, whereas EE provides the probability of a weaponization occurring any time in the future, without considering any time frame. We envision that these limitations, of considering a fixed window size or

an infinite time horizon, open up opportunities for investigation, which we begin to tackle in this preliminary report.

Our contribution. In summary, our key contribution is a novel approach to estimate the distribution of time until first weaponization or first exploitation in the wild, given previous information about events associated with a vulnerability.

Outline. The rest of this paper is organized as follows. In Section 2 we present our challenges and insights. Then, Section 3 presents our methodology, including the production of the ground-truth. Finally, Section 4 presents our evaluation and concluding remarks.

2 Challenges and Insights

2.1 Where to Find Data About Vulnerabilities?

Challenge: When it comes to predicting weaponization, numerous challenges arise. One primary challenge is the acquisition of meaningful data to train the model. Many valuable sources of information are inaccessible due to paywalls, making it difficult to obtain high-quality, open resources.

Insight: The approach involves gathering data from diverse sources, including the National Vulnerability Database (NVD) and the resources linked from NVD.

Details: The NVD serves as a central hub in the vulnerability life cycle. To ensure that all data remains current, we have developed crawlers and parsers that automate the extraction of information from both NVD and the various external resources linked to each vulnerability page. Subsequently, the extracted data is stored in a structured format within a PostgreSQL database.

2.2 How to Cope with Static and Dynamic Data?

Challenge: The data pertaining to vulnerabilities typically falls into two categories: static and dynamic. Static data remains unchanged after its initial publication in NVD, while dynamic data relates to events that occur and are added over time following the disclosure of the vulnerability.

Insight: To effectively handle the different types of data, static and dynamic information should be treated differently. Static data is stored in a single record per vulnerability, whereas dynamic data is stored in a single record per event.

Details: All the data published by NVD regarding each vulnerability is considered static, as it originates from the initial analysis and does not change or evolve in the future, except for external resource links. On the other hand, dynamic or non-static data is collected after the publication date from various sources, including vendor-specific advisories, internet forums, ExploitDB, GitHub, and other general sources.

The static data is stored in our database as one record per vulnerability, while the dynamic data is stored as one record per event. Extracting data from NVD is relatively straightforward since they offer an API that provides structured

JSON format. However, obtaining data from other sources, such as publication dates of events, can present challenges, highlighting another obstacle in the data acquisition process, as further detailed next.

2.3 How to Determine Dates of Events?

Challenge: How can we accurately determine the dates when vulnerabilities are weaponized, as well as the dates for other types of events, in order to create a comprehensive temporal series?

Insight: When dates are available from external resources, utilize those dates. If such dates are not provided, employ heuristics to estimate the dates.

Details: To address this challenge, we employ multiple levels of data extraction techniques. Initially, we strive for precision by utilizing crawlers and parsers. We extract information from references provided in NVD and identify the top 20 most frequently used domains. Subsequently, we create tailored crawlers for each domain, taking into account the variations that exist within each one. Some domains may offer APIs, while others necessitate careful parsing of the web page itself.

The final level of extraction pertains to NVD itself. We delve into the log of changes maintained by NVD for each vulnerability. This log enables us to determine when a reference associated with an event is added to the vulnerability web page. By examining these changes, we can deduce relevant dates for the events.

By employing this multi-level approach, we aim to capture accurate dates for weaponization and other events, ensuring the creation of a comprehensive temporal series.

2.4 How to Leverage Dynamic Information about Events to Assess Time to Weaponization?

Challenge: Continuous reporting of events related to vulnerabilities impacts the time it takes for weaponization. The challenge lies in effectively utilizing these events.

Insight: Consider the information associated with each event and take into account previous events. Construct a dataset where each row corresponds to an event, and each column represents features of that event, including cumulative information about prior events.

Details: Events associated with vulnerabilities serve as the fundamental units of analysis. These events encompass various occurrences, including weaponization events. The primary objective is to develop a model that predicts the date of the first weaponization event, utilizing static data about the vulnerability and potentially dynamic data. Furthermore, the aim is to obtain the complete distribution of dates until the first weapon release, including their corresponding probabilities, rather than solely focusing on the most probable date.

To achieve this, we leverage the multiple events linked to a vulnerability through event-driven training and inference. In our training set, we organize

events into tables grouped on a daily basis. Each row in our dataset corresponds to a day when an event occurred, in contrast to previous approaches where rows corresponded to vulnerabilities rather than individual events. To handle dynamic features associated with each event day, we incorporate the count of previous events associated with the given vulnerability. Additionally, we retain the last date of significant events. The objective is to capture the notion that as the number of events increases, the likelihood of an imminent weapon release also rises.

Other studies, like EPSS, also employ a count of the number of citations to a specific CVE, resulting in 17 features that encompass various reference types. However, in the training set of EPSS, they have a single static observation per CVE, whereas our training set comprises multiple observations per CVE, with each observation representing an event. This enables us to capture temporal trends over time.

2.5 How to Infer Distributions of Time to Weaponization using Predictions from Classifiers?

Challenge: When considering the time to weaponization, it is important to establish a distribution for each point in time. The challenge lies in determining how to generate a distribution from the class predictions, given a vulnerability and a timestamp.

Insight: Construct a conditional cumulative distribution function (CDF) for the time to weaponization for each class. Combine these CDFs using the weights derived from the multiple classifier outputs.

Details: We individually classify events into multiple classes. Each class represents a specific range of times to weaponization, allowing us to create a CDF for each range. Although these classes are mutually exclusive, when classifying an event, there is a probability associated with each class. By utilizing the probabilities of these classes, we can merge the conditional CDFs into a single CDF.

While the shapes of the individual class CDFs remain unchanged after training, we assign weights to each of them and combine them to create a highly customized final combined CDF. Additionally, classes that are closer to the event's origin date possess a higher level of resolution compared to those further away from the origin. This means that the final CDF provides greater detail near the event date while gradually losing precision as it moves further away from the origin. This approach allows for the creation of unique CDFs that offer enhanced detail where it is most crucial.

3 Methodology

In our methodology, we utilize XGBoost to predict the time it takes for weaponization to occur, where time is measured in intervals given by the 26 classes described above. Each class corresponds to a conditional CDF of time to first

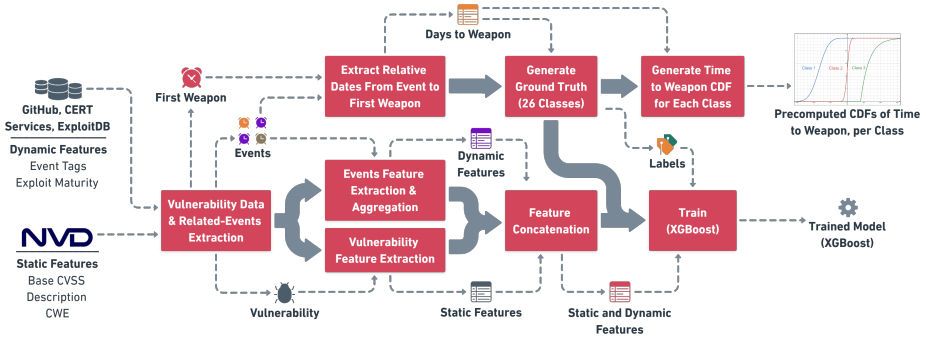


Fig. 1. Proposed training pipeline.

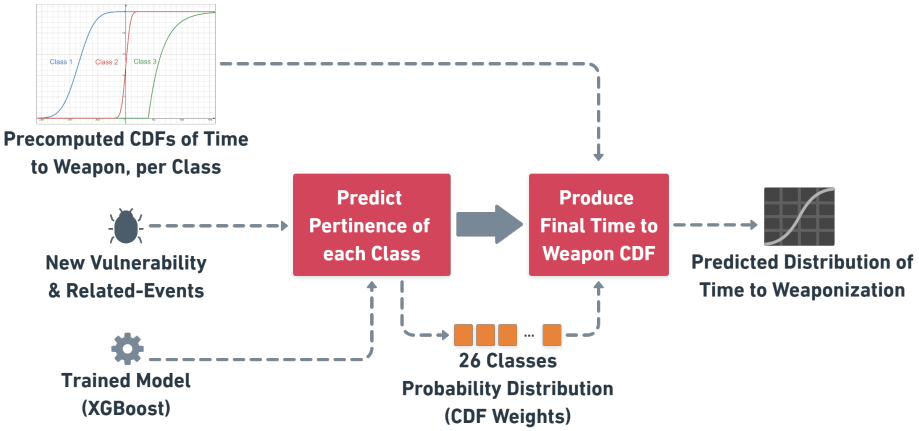


Fig. 2. Inference of CDF of time to weaponization.

weaponization. By combining the results from XGBoost with the conditional CDFs, we generate an unconditional CDF that provides insights into the probability of the first weapon being released within a given timeframe after a vulnerability-related event. Figure 1 provides an overview of the entire methodology.

3.1 Ground Truth

The ground truth consists of 26 distinct classes. Each of the events in our dataset is assigned to a specific class. Such mapping is encoded using one-hot encoding. The distribution of vulnerability-related events across these classes can be observed in Figure 3. Among the classes, there are 12 negative classes denoting weapons that are released prior to the designated event. Additionally, there is one class specifically for day-0 events, signifying that the weaponization event coincides with the considered event. Furthermore, there are 12 positive classes

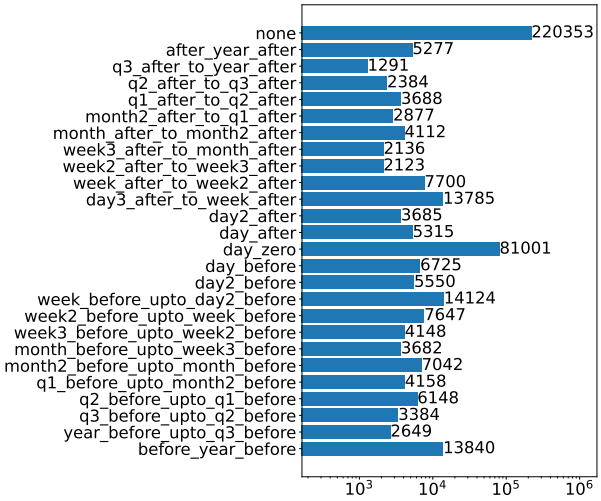


Fig. 3. Distribution of the vulnerability events over the classes of the ground-truth

indicating that the exploit is released after the event date. Finally, there is one class dedicated to events of vulnerabilities that do not have any associated exploits.

The data in our dataset is distributed across multiple classes, as depicted in Figure 3. This distribution can serve as a prior distribution for our data across the classes. Moreover, we possess information regarding the internal distribution of data within each class. We organize this information into conditional cumulative distribution functions (CDFs), with one CDF dedicated to each class. Subsequently, we employ XGBoost to combine these conditional CDFs and generate a final CDF, as elaborated further in the following sections.

3.2 XGBoost for Classification

The XGBoost model is trained to classify dates surrounding the event related to the vulnerability into 26 distinct classes. The distribution of days in each class varies based on their proximity to the event date. Classes representing dates closer to the event have a smaller range of days, while those further away have a larger range. This design allows the model to exhibit higher precision in predicting dates near the event and gradually decrease precision as the prediction moves further from the event. Additionally, there is a specific class that represents a date infinitely into the future, indicating that no exploit will occur following the given event.

In the prediction array, each of the 26 classes is individually represented. This corresponds to the one-hot encoding scheme used in the ground truth. Moreover, when applying the XGBoost model, weights are generated for each class, reflecting their pertinence in the predictions.

3.3 Normalization of Class Pertinences

Once we obtain the prediction array, we proceed to normalize the values within the array to ensure their sum equals one. Since our training data consists solely of zeros and ones (one-hot encoding) the majority of predictions generated by XGBoost are also between zero and one. In the rare instances where XGBoost produces negative values, we replace them with zeros, as our experiments reveal that their absolute values never exceed 0.01.

Note that when provided with an event associated with a vulnerability, the proposed predictor will present multiple weighted classes of date ranges in which weaponization is likely to occur, rather than a single class denoting the most probable outcome.

3.4 Building a CDF Per Class

Next, we proceed to construct the Cumulative Distribution Function (CDF). Each class consists of a distinct set of events that truly belong to that class. For example, the zero-day class encompasses all events that occurred on the day of weaponization. In these events, the ground-truth value for the weaponization day is 0. We refer to these events as zero-day events, which belong to class 0. It is important to note that each class corresponds to a specific set of events, and for each event, we have its corresponding relative date of weaponization, even if the date is infinite.

Each set of events that belong to a class can be utilized to derive a CDF representing the time to weaponization. In fact, each class corresponds to a specific time window, and each event that falls within that window is represented by an integer denoting the number of days until weaponization within that window. The support of the CDF encompasses the range of the time window, while its shape captures the distribution of events within that window.

3.5 Combining the CDFs to Produce a Final Time to Event CDF

We aggregate the individual CDFs described above to generate a unified CDF that represents the complete distribution of the probability of the first weaponization. Let $F(x)$ denote the combined CDF. We express $F(x)$ as a weighted sum:

$$F(x) = \sum_{i=1}^{I} w_i F_i(x) + b_i, \tag{1}$$

where w_i represents the weight assigned to the i-th CDF. Additionally, we define b_i as:

$$b_i = \sum_{j=1}^{i-1} w_j. \tag{2}$$

It should be noted that w_I corresponds to the probability that the vulnerability will never undergo weaponization, while b_I represents its complement, with $w_I = 1 - b_I$.

3.6 Summary

In summary, our approach involves constructing a composite CDF by combining multiple conditional CDFs, using the weights generated by XGBoost. This process can be visualized as multiplying the height of each conditional CDF by the predicted probability for its corresponding class and stacking them together.

In practical terms, when an event occurs, our methodology generates a CDF that represents the distribution of the time to weaponization. This is achieved by leveraging two main components: 1) the XGBoost predictions, which determine the vector of weights w_i, and 2) the pre-computed shapes $F_i(x)$ that were obtained during the training phase.

4 Evaluation

4.1 Model Training

To traing XGboost we (1) separate data in training and test sets with stratification on the target class, 25% for training, and 75% for test; (2) tune hyperparameters using a grid-search method; (3) use XGBoost build-in cross-validation with 7 folds and (4) select best model parameters and collect prediction metrics on test set

We determined the number of trees by utilizing the "early stopping rounds" parameter of XGBoost. This parameter specifies that if the validation errors start increasing for a certain number of rounds, the model should stop generating more decision trees.

4.2 Transfer Learning: Bridging Time to Weaponization and Time to Exploitation

To evaluate the proposed model in a concrete setting, we explore the concept of transfer learning, specifically examining the relationship between time to weaponization and time to exploitation.

In real-world scenarios, exploitation relies on the availability of an exploit that can be utilized as a weapon. Therefore, any prediction of the first weaponization also serves as a prediction for the earliest possible date an attack could occur.

Our findings indicate a strong correlation between our predictions and the EPSS (Exploit Prediction Scoring System) scores of significant vulnerabilities. This correlation is expected since EPSS evaluates the likelihood of exploitation in the wild, indicating the probable existence of an exploit. Consequently, dates associated with high EPSS probability values are likely to exhibit a substantial increase in the probability of exploit existence.

Figure 4 illustrates a comparison between the CDFs of multiple events for CVE-2021-44228 and the corresponding evolution of EPSS scores. This vulnerability, associated with Log4j, enables attackers to take control of the affected device.

The initial event, which occurred on day 0, corresponds to the publication of the vulnerability in the National Vulnerability Database (NVD). According to our model, at this stage, there is a 50% probability that the vulnerability has already been weaponized. Subsequently, for each subsequent event, our proposed model is utilized to generate an updated cumulative distribution function (CDF) of the time to weaponization.

As depicted in Figure 4, we observe a gradual convergence of the CDFs associated with additional events towards the EPSS curve. The EPSS curve itself is derived from multiple point estimates of the EPSS collected over time. Our findings suggest that this collection of EPSS values, obtained from a black box model, could potentially serve as a limiting CDF for the time to weaponization, which is derived from public events gathered from NVD, as the number of events increases. Conversely, by utilizing public data acquired from NVD, one could infer the limiting behavior of the EPSS curve given a sufficient number of events associated with the vulnerability.

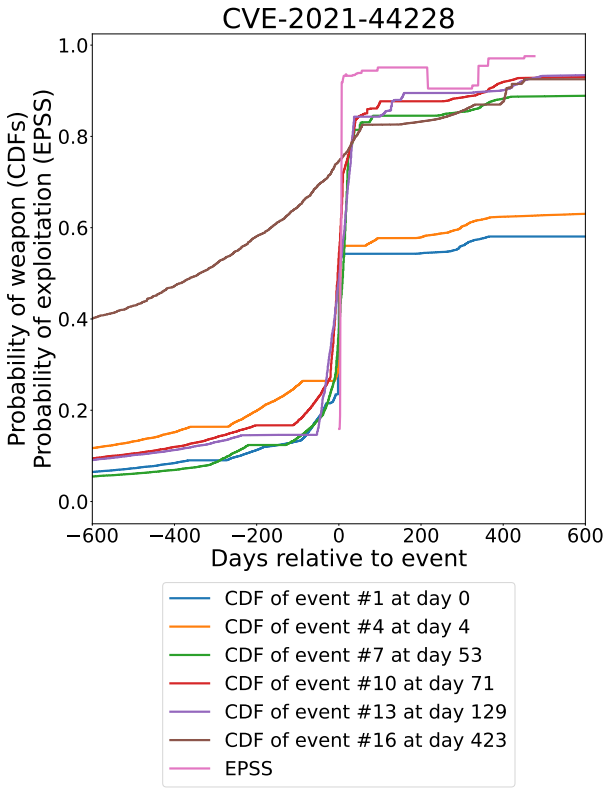


Fig. 4. CDF of time to weaponization contrasted against EPSS

References

1. Allen D Householder, Jeff Chrabaszcz, Trent Novelly, David Warren, and Jonathan M Spring. Historical analysis of exploit availability timelines. In *CSET@ USENIX Security Symposium*, 2020.
2. Jay Jacobs, Sasha Romanosky, Benjamin Edwards, Idris Adjerid, and Michael Roytman. Exploit prediction scoring system (EPSS). *Digital Threats: Research and Practice*, 2(3):1–17, 2021.
3. Jay Jacobs, Sasha Romanosky, Octavian Suciuo, Benjamin Edwards, and Armin Sarabi. Enhancing vulnerability prioritization: Data-driven exploit predictions with community-driven insights. *arXiv preprint arXiv:2302.14172*, 2023.
4. Octavian Suciu, Connor Nelson, Zhuoer Lyu, Tiffany Bao, and Tudor Dumitraș. Expected exploitability: Predicting the development of functional vulnerability exploits. In *31st USENIX Security Symposium (USENIX Security 22)*, pages 377–394, 2022.
5. Jiao Yin, MingJian Tang, Jinli Cao, Hua Wang, Mingshan You, and Yongzheng Lin. Vulnerability exploitation time prediction: an integrated framework for dynamic imbalanced learning. *World Wide Web*, pages 1–23, 2022.

Avatar PLC/SCADA: Cloud Half-Twin for Industrial Control Systems

(Extended Abstract)

Alon Dankner, Shlomi Dolev, and Ehud Gudes

Department of Computer Science
Ben-Gurion University of the Negev
Beer Sheva, Israel
{danknera,dolev,ehud}@cs.bgu.ac.il

Abstract. Industry 4.0 drives the ICS and SCADA systems to utilize the cloud for its significant benefits. New models, designs, and applications of the traditional ICS systems incorporate the cloud to contribute to and optimize the industrial process. In this paper, we propose a novel model that uses avatar processes for devices in the ICS system, like the PLCs. The avatar runs on the cloud and executes PLC tasks. The avatar can perform heavy computations that were not possible before, especially for legacy devices that are common in these systems. We also describe a security architecture for our system that thwarts cyber security threats that may arise from attackers on the cloud. Furthermore, we introduce a new decentralized architecture that combines blockchain with avatars to secure our ecosystem. In total, our system improves the industrial process and allows engineers to easily add complex logic to their (possibly heterogeneous and legacy) systems.

1 Introduction

An industrial control system (ICS) is a hardware and software system with network connectivity that automates an industrial process. These systems are very common in factories and critical infrastructure facilities such as power plants, water treatment, oil and gas processing, and transportation systems. They monitor and control the industrial processes and they are vital to modern life.

According to the Purdue model, a traditional ICS system consists of four levels, as can be seen in Fig. 1. The model consists of four layers:

1. The Field Layer contains the industrial equipment, e.g., turbines, generators, water pumps, centrifuges, and more.
2. The industrial equipment is controlled by the PLCs that reside in the Control Layer.
3. The Supervisory Layer consists of SCADA HMIs and engineering stations. The engineering stations allow the engineers to program and configure the PLCs, while the SCADA HMIs allow the operators to monitor and control the industrial process.

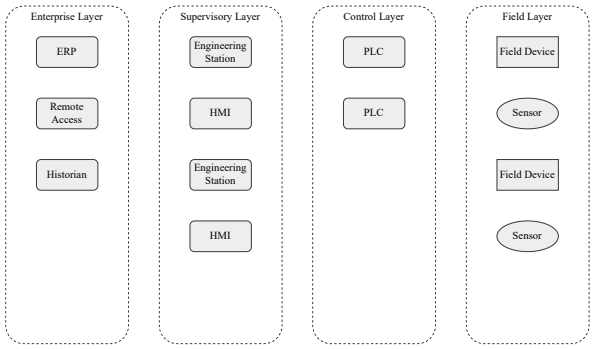


Fig. 1. Purdue Model

4. The Enterprise Layer is a typical IT network for business systems such as ERP and SAP. It also contains the historian, which stores ICS data.

Note that in most organizations, firewalls (that are not illustrated in Fig. 1) forcibly separate between the layers, and certainly between the Enterprise Layer and the Internet.

Nowadays, however, some express reservations about the traditional model. As part of the fourth industrial revolution, a.k.a. Industry 4.0, new technologies for the system are moving to the cloud [5]. This allows organizations to improve the industrial process, be more effective, and keep up with the pace of the industry. Enhancing the computation and functionality capabilities, rather than replacement of the legacy controlled devices and controllers that already exist on the factory floor.

Several models were proposed to define the interactions between the cloud and the ICS system [10]. Some systems, which are called open-loop systems, store ICS data on the cloud (e.g., the historian is moved to the cloud). The more interesting models, which are called closed-loop systems, allow the cloud to make decisions that affect the industrial process. We define a novel approach for a closed-loop system that allows the PLC itself (or, parts of it) to run on the cloud.

We propose a new framework for ICS systems that uses avatar process [3,6] – processes that represent each PLC (and possibly each controlled device on the factory floor in the cloud), in the cloud. The avatar communicates with the device it represents and may tune or even replace the program running on that device.

We focus on separating the real-time portions of programs to run on the PLCs that reside on the industrial site, while migrating the non-real-time aspects to avatars that run on the cloud. The cloud avatar can be sophisticated (having the cloud computation and communication resources) and may implement complex algorithms and expensive security layers. Avatars can also be created for field devices and sensors (those that use TCP/IP). The avatar of a sensor, for example,

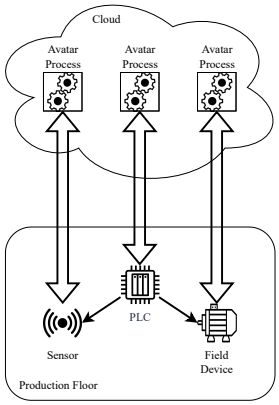


Fig. 2. Avatar PLCs in the Cloud

may record all the readings of the sensor and stop its operation if it is faulty and alike.

This architecture is also useful to enhance computing devices of cars or IoT devices. As the devices have limited computation power and/or become obsolete, they can be enhanced by offloading tasks to the avatar in the cloud, leaving only the real-time portions on the devices.

The cloud provides substantial benefits for the industry. It contains redundant, expandable hardware and software components that are well connected and can be programmed to fit changing situations (e.g., heterogeneous devices, controllers, and extreme conditions such as disaster recovery) and have less energy and other limitations that devices and controllers on the factory floor may have. These resources are maintained by the cloud provider. The use of the cloud reduces the expenses of the organization in many aspects.

Another advantage of the architecture is that avatars may communicate with each other in the cloud to form online distributed decisions and adaptations.

Another notable technology is the digital twin [13]. The digital twin is a digital representation of a device, system, or process. It allows organizations to analyze the system, run simulations, and get predictions about the system. The digital twin is inherently different than our avatar PLC, as we do not replicate the entire ICS network or PLC, we move some of the PLC logic to the cloud to achieve better performance and other useful features. We keep the real-time restricted operations done on the production floor and let the cloud assist with system tuning/programming to achieve better performance, security, and more.

2 Avatar PLC

In our new architecture, the local PLCs in the plant are accompanied by avatar PLCs that run in the cloud. Fig. 2 illustrates the model. The avatar PLC is an

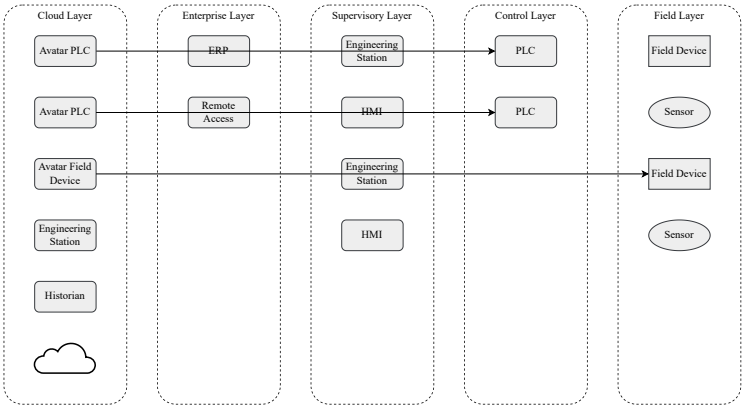


Fig. 3. ICS Topology with Avatar PLCs

avatar process [3,6] that communicates with the local PLC.

As can be seen in Fig. 3, a new layer is added to the traditional model, the Cloud Layer. This layer crosses all the layers up to the field layer and may affect each of them. The Cloud Layer contains the following entities:

- The avatar processes for the ICS devices. In this paper, we focus on avatars for the PLCs.
- An application of an engineering station or a SCADA HMI may be added to the cloud if access is needed at any time from anywhere on Earth. Alternatively, it is also possible to program the PLCs and their avatar solely from the plant.
- Additionally, the historian is moved to the cloud to store historical ICS data in a massive amount.

We split the programmable logic of the PLC into two parts. One part consists of critical code snippets that must be executed in real-time, and the other part consists of code snippets that are not critical.

In many cases, it is crucial that parts of the programmable logic run locally at the PLC. These snippets cannot tolerate network latency. Therefore, we manage to provide offloading of PLC tasks to the cloud.

An example of such logic is predictive maintenance, which typically uses machine learning techniques. Suppose that the PLC should analyze the data history, and at runtime, it needs to use the output of a machine learning algorithm. The part of the program that implements the machine learning algorithm can run as an avatar process and transmit the output to the PLC, there is no need for this code to run locally at the PLC. An example of the use of machine learning in autonomous cars is the implementation of the vision and object identification component appear in, e.g., [2].

A major advantage of the avatar PLC is that it can perform heavy computations, which are not necessarily possible on the PLC. Examples include machine

learning algorithms, simulations, and intrusion detection. PLCs typically do not have the ability to perform complicated calculations. To make matters worse, low replacement and update frequency characterize existing ICS deployments. PLCs have a lifetime that can reach decades (unlike devices in standard IT networks). As a result, there are many legacy PLCs with limited computation power.

Another advantage of avatar PLC is the cost reduction. Many places offer a discounted rate for electricity in certain time frames. The avatar PLC can plan when the production costs the least and operate the system at that time.

Our avatar actually contains a certain percentage of the PLC code. Notice that in the case of 100% of the code, the PLC is basically a relay for the field devices and it is unnecessary. We introduce a new model of PLC as a Service [8] which is realized through the avatars. It is important to note that in this case, we compromise the real-time requirement since all operations come from the cloud.

In this architecture, the engineering stations and SCADA HMIs are accessible from the cloud as well. The PLCs communicate directly with the field devices using the same protocols as before.

Notice that communication with the cloud depends on the ISP, but the industrial process must not be affected by failures such as loss of communication. For this purpose, the engineer can implement an emergency program that runs in case of a communication failure with the cloud.

2.1 The Role of the Engineering Station

The engineering station allows the user, or the engineer, to implement a control logic program, load it to the PLC, and send a variety of other commands to the PLC. Since parts of the control program run on the cloud in our new system, the role of the engineering station must be adapted and extended accordingly.

The engineering station has a special task during the installation of the avatar, securely coupling the PLC/device to its avatar. Later, the avatar and its coupled PLC/device communicate directly (using an encrypted channel with private key(s) unrevealed to the engineering station). Thus, following installation, the engineering station communicates only with the avatar, which in turn allows the usage of commands and reports in a high level of abstraction.

Suppose that the engineer implements a control logic program for the PLC and the avatar. The engineering station prepares the running environment for the program as follows:

- Creates a new avatar process if one does not already exist.
- Gives the identification and security information of the PLC to the avatar.
- Loads the program to the avatar.
- Gives the identification and security information of the avatar to the PLC.
- Loads the program to the PLC.

The PLC can then start running the loaded program and communicating with its avatar process.

Communication with the cloud, however, holds various security risks. In the following section, we address these risks and provide a holistic security solution for our new system.

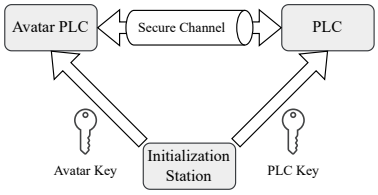


Fig. 4. Key Distribution in High Level

3 Security Architecture

ICS systems are an attractive target for cyber attacks due to their importance and criticality. Over the years, many attacks against ICS systems were publicly disclosed by the media [7,4,1]. In this section, we describe a security architecture for our new system that allows the devices in the plant to securely communicate with their matching avatars in the cloud.

A major difference in our new architecture, compared to the traditional architecture, is the inbound traffic to the industrial network. In the traditional architecture, data is only sent out of the network. In this new architecture, messages and commands are coming from outside the industrial network directly to the PLCs, similar to other closed-loop systems [10]. This opens a window for attacks that were not possible before; the threat model is drastically changed.

This change allows for possibly-malicious traffic to impact the industrial networks which are inherently insecure. The Modbus/TCP protocol [12], for example, does not provide integrity or confidentiality for the messages. As a result, an attacker can eavesdrop and manipulate sensitive traffic.

Our goal is to limit the broad attack surface created by the transition of the control logic to the cloud.

The communication between the PLC and the avatar is done over a secure channel, like TLS [11] or IPSec [9]. I.e., there is an authenticated and encrypted tunnel between the PLC and the avatar. The tunnel allows the avatar to securely send commands and control the PLC. For this purpose, we use the engineering station to provide the PLC and its avatar with security information. We design a new protocol at the end of which both parties can establish a secure channel using cryptographic keys.

When the engineering station creates the avatar, it sends a pair of public key and private key both to the PLC and the avatar. The engineering station also sends the identifying details of the PLC and its public key to the avatar and vice versa. This process is illustrated in Fig. 4.

Using the public keys, the PLC and its avatar form a secure channel to communicate securely. When a PLC wants to communicate with its avatar, they use a challenge-response protocol to authenticate their identities. Each entity validates the identity of the other with the public key. If the validation is successful, the entity knows that the other party is the real avatar and not an impostor. Then, they use the public keys to agree on a shared symmetric key. This key is

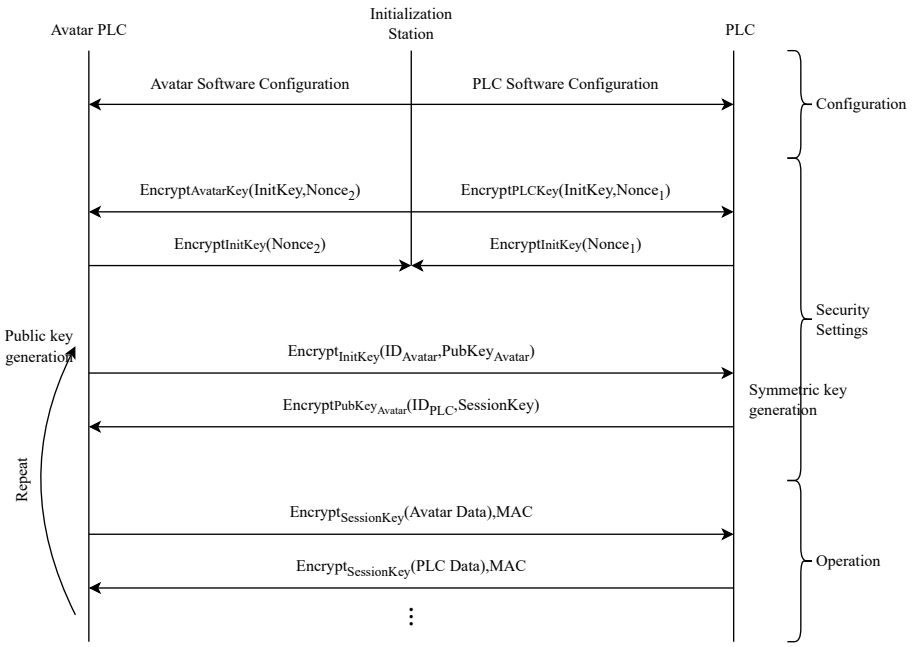


Fig. 5. Key Distribution Protocol

a short-term key that is used for a limited period. If the PLC and the avatar have already agreed on a key in the near past, they continue using it without generating a new key.

A public key can be revoked, e.g., if the private key of the entity is compromised. In such a case, the programmer uses the engineering station to load another public key to the PLC and the avatar.

A benefit of this architecture is that the avatar PLC can run security tools that do not exist today on standard PLCs, such as sophisticated firewalls and anti-virus systems. Nowadays, many attacks against ICS systems use takeovers on the PLCs that do not run such tools. We migrate these security tools to the avatar, and therefore such tools dramatically enhance the security of the whole system.

4 Key Distribution Protocol

The following are details about our new key distribution protocol: a cryptographic protocol between the avatar, its PLC, and the initialization station (which is implemented as part of the engineering station). The protocol is outlined in Fig. 5. The initialization station, as its name suggests, initiates the protocol and bootstraps the cryptographic keys of the parties.

We assume that both the PLC and the avatar have initial keys installed. These keys are stored outside the framework of the system, e.g., by the PLC vendor.

Configuration. The protocol starts when the initialization station sends a new configuration for the avatar and the PLC, such as a new logic.

Bootstrapping. The initialization station generates a new symmetric key, $InitKey$, and sends it to the avatar and the PLC. The new key is encrypted together with a random nonce, which is used as a challenge, under the receiver's key. The receiver decrypts $InitKey$ and its nonce from the message. Then, it encrypts the nonce under $InitKey$ and sends it to the initialization station. The last job of the initialization station is to make sure that the nonce is properly returned.

Creating a new key. The avatar randomly generates an ephemeral public key pair. It uses $InitKey$ to encrypt the public key with its ID and sends the result to the PLC. The PLC decrypts the received ciphertext and validates the ID of the avatar. It then creates a symmetric session key, $SessionKey$, and uses the public key to encrypt it with its ID using quantum-safe encryption. Upon receipt, the avatar decrypts the ciphertext with the private key and validates the ID of the PLC. Moreover, the ephemeral public key pair is removed from the memory of the avatar.

Operation. After the message exchange is completed, the avatar and the PLC share a symmetric key that is used to secure the session. The parties use this session key to form a secure channel, over which they can send messages securely. Messages are protected using encryption and authentication.

Our security architecture provides various security features:

- **Authentication** Each party knows how to verify the other party's identity. To communicate, each party needs to decrypt $InitKey$, which is encrypted with its secret key. The parties receive it in a message from the initialization station, and they only have to decrypt the message to obtain the key. In the next message exchange between the parties, they both use $InitKey$, so they identify a rogue machine if it exists.
- **Integrity and confidentiality** Secure the transmitted data from malicious modification and reading (respectively) by an attacker by the secure channel that encrypts and authenticates the data.
- **Forward secrecy** The protocol provides forward secrecy because if either of the long-term keys is exposed, $AvatarKey$ or $PLCKey$, it will lead to exposure of $InitKey$, because the attacker can decrypt it from the traffic. Still, the attacker will not be able to decrypt the avatar–PLC data. The ephemeral key, $PubKey_{PLC}$, protects against such an attack because the private key remains secret, even if long-term keys are compromised, as it is not sent over the channel. Therefore, the session key also remains secret and the traffic is protected.
- **Backward secrecy** In the opposite direction, the protocol provides backward secrecy because the parties replace the session key every once in a

while. Therefore, even if long-term keys are compromised, an attacker still fails to decrypt future traffic.
- **Post-quantum safety** The protocol is post-quantum safe because we use post-quantum cryptography. Examples of such primitives are AES-256 for symmetric encryption and lattice-based cryptography for asymmetric encryption.

5 Implementation Concepts for the Avatar PLC/SCADA

Several implementation aspects that their details are omitted from this extended abstract.

- **Real-Time-Oriented Programming** Extensions to common programming languages and compilers that allow the programmer to mark parts of the code as real-time code. We provide an interface for the programmer, to tell the PLC and the avatar which parts should run locally on the PLC since they are critical, and which parts may take longer to execute and should be offloaded to the cloud.
- **Real-Time-Oriented Compiler** A compiler that automatically turns a generic program into a real-time-oriented program. The compiler decides which parts run locally and which parts run on the cloud based on program analysis.
- **Adaptive Real-Time Partition** A program that automatically splits a program into a real-time-oriented program. The PLC and the avatar learn which parts must run with real-time restrictions and should be executed locally and which parts would take longer.
- **Blockchain** Investigate the use of blockchain, a decentralized system, in the environment of ICS systems. Using smart contracts in blockchain to act as distributively implemented avatars. Blockchain provides various security features that allow to securely store information on a ledger.
- **Machine Learning Models** Use machine learning algorithms on the data the avatars collect from the sensors. Constantly learning and tuning operations according to the collected data from the controlled production floor, from the PLCs, sensors, activators, cameras, and alike.
- **Distributed Computations** Our architecture allows the PLCs to work together and perform computations in a way that is not possible before, via their avatars.

References

1. Eli Biham, Sara Bitan, Aviad Carmel, Alon Dankner, Uriel Malin, and Avishai Wool. Rogue7: Rogue engineering-station attacks on s7 simatic plcs. *Black Hat USA*, 2019, 2019.
2. Alexey Bochkovskiy, Chien-Yao Wang, and Hong-Yuan Mark Liao. Yolov4: Optimal speed and accuracy of object detection. *arXiv preprint arXiv:2004.10934*, 2020.

3. Cristian Borcea, Xiaoning Ding, Narain Gehani, Reza Curtmola, Mohammad A Khan, and Hillol Debnath. Avatar: Mobile distributed computing in the cloud. In *2015 3rd IEEE International Conference on Mobile Cloud Computing, Services, and Engineering*, pages 151–156. IEEE, 2015.
4. Defense Use Case. Analysis of the cyber attack on the ukrainian power grid. *Electricity Information Sharing and Analysis Center (E-ISAC)*, 388:1–29, 2016.
5. Philip Church, Harald Mueller, Caspar Ryan, Spyridon V Gogouvitis, Andrzej Goscinski, Houssam Haitof, and Zahir Tari. SCADA systems in the cloud. In *Handbook of Big Data Technologies*, pages 691–718. Springer, 2017.
6. Shlomi Dolev, Marina Kopeetsky, and Dudu Mimran. Avatar process for mobile devices, February 2014. EP2696608 A2.
7. Nicolas Falliere, Liam O Murchu, and Eric Chien. W32. stuxnet dossier. *White paper, symantec corp., security response*, 5(6):29, 2011.
8. Omid Givehchi, Jahanzaib Imtiaz, Henning Trsek, and Juergen Jasperneite. Control-as-a-service from the cloud: A case study for using virtualized PLCs. In *2014 10th IEEE Workshop on Factory Communication Systems (WFCS 2014)*, pages 1–4. IEEE, 2014.
9. Stephen Kent and Karen Seo. Security architecture for the internet protocol. RFC 4301, 2005.
10. Dale Peterson. Securing closed loop ICS cloud services. `https://dale-peterson.com/2020/02/13/securing-closed-loop-ics-cloud-services/`, 2020.
11. Eric Rescorla. The transport layer security (TLS) protocol version 1.3. RFC 8446, 2018.
12. Andy Swales et al. Open modbus/TCP specification. *Schneider Electric*, 29:3–19, 1999.
13. Fei Tao, He Zhang, Ang Liu, and Andrew YC Nee. Digital twin in industry: State-of-the-art. *IEEE Transactions on industrial informatics*, 15(4):2405–2415, 2018.

Waves Interference for Perfect Output VES (Preliminary Version)

Shlomi Dolev, Alexander Fok, and Michael Segal

Ben Gurion University of the Negev, Beer Sheba, Israel

Abstract. Known Visual Encryption Scheme (VES) schemes encode the secret image pixels into subpixel maps of size $m \times m$, where m is a parameter of the scheme. The pixel encoding is usually based on pixel visual property, for example - transparency. The resulting pixel maps contain black and white pixels and look like random collection of black and white pixels, such that it is impossible to reconstruct the original pixel. To reconstruct the original secret image, more than T shares should be stacked. The reconstructed image will look grey with darker or whiter pixels. It happens because the stacked subpixel maps represent a different (implementing or logical operation) levels of grey - from very black to some level of grey. In this work we introduce an optical VES solution that leverages a physical model of waves interference. The image reconstructed with the proposed VES consists of pure white and black pixels, while preserving traditional VES computational efficiency. The proposed VES is perfect information theoretical secure.

Keywords: Visual Secret Sharing · Visual Encryption Scheme · Image Encryption · Information-Theoretic Secure Solution

1 Introduction

All known VES schemes encode the secret image pixels into subpixel maps of size $m \times m$, where m is a parameter of the scheme. The pixel encoding is usually based on some pixel visual property, for example - transparency, as in [2] or light intensity, as in [1]. The resulting pixel maps contain black and white pixels, as shown in Figure 1. To reconstruct the original secret image, the shares are stacked. If we put source of light under the stacked shares (created from pixel transparencies), the reconstructed image will look grey with darker or whiter pixels, namely, there will not be a totally white pixel. It happens because the stacked subpixel maps represent different levels of grey - from very black to some level of grey. More precisely, stacked subpixel maps representing black pixels will have more than T black subpixels in the same positions of $m \times m$ maps, and stacked subpixel maps representing white pixels will have less than T black subpixels in the same positions of $m \times m$ maps, as shown in Figure: 1. Some of the existing VES schemes, including the Naor and Shamir original scheme, can be extended to deal with the concealing of the secret image. There are various methods to generate shares from the secret image, but all known

methods generate grey scale shares that do not effectively hide the fact that there is secret communication. For example, using Naor and Shamir proposed extension, we can use innocent images of dog and cat to hide image of house. For that, we create two shares - D that represents the dog and C that represents the cat. When D and C are stacked, they reveal H - secret image of house. The problem here is that all images participating in calculation, are grey images. It includes shares D and C, and the reconstructed image H.

2 Related Work

2.1 Visual Encryption Schemes (VES) Background

Various **Visual Encryption Schemes (VES)** are proposed by researchers. Naor and Shamir in [2] proposed a VES based on a pixel transparency sharing. They define k out of n scheme that works as follows. The secret image is encoded into n slides. At least k slides are required to recover the original image. The image slide (or share) consists of the pixel subpixels (shares) that are populated as following. Secret image T consists of black and white pixels. Every image pixel is treated and encoded separately. There are two collections of $n \times m$ matrices C_0 and C_1. To encode a white pixel, a matrix from C_0 is randomly drawn. To encode a black pixel, a matrix from C_1 is randomly drawn. Every row from the chosen matrix is the pixel share. It defines the color of m pixels in each one of the n shares. Generally, it can be described as Boolean matrix $S = [s_{ij}]$, where $s_{ij} = 1$ iff the jth subpixel in the ith slide is black. When the slides are stacked together, we see a combined share whose black subpixels are represented by Boolean "or" of rows $Si_1, i_2, \ldots i_r$ in S.

Example: $k = 2$**.** Let C_0 and C_1 be two sets of all matrices obtained by permuting the columns of matrices as described below:

$$C_0 = \left\{ \begin{array}{c} \text{Matrices obtained by permuting of the columns of} \\ \begin{bmatrix} 1 & 0 \ldots 0 \\ 1 & 0 \ldots 0 \\ \ldots & \\ 1 & 0 \ldots 0 \end{bmatrix} \end{array} \right\}$$

$$C_1 = \left\{ \begin{array}{c} \text{Matrices obtained by permuting of the columns of} \\ \begin{bmatrix} 1 & 0 \ldots 0 \\ 0 & 1 \ldots 0 \\ \ldots & \\ 0 & 0 \ldots 1 \end{bmatrix} \end{array} \right\}$$

Any single share in either C_0 *or* C_1 is a random collection of one black and $n-1$ white pixels. Any two shares of a white pixel have a combined Hamming weight of 1. Any two shares of a black pixel have a combined Hamming weight of 2. To avoid a distortion of the aspect ratio of the secret image, it is convenient to represent the shares as two dimensional arrays $d \times d$.

An example of VES scheme pixel share options is shown in Figure 1, that fits 2 out of 2 VES scheme. This scheme does not keep a visual aspect ratio of the VES shares (avoiding images distortion), like authors are doing in original VES paper [2], but still is a valid VES example. Every target image pixel is randomly given a value from the left or the right table. For the white pixel, the same share is chosen. For the black pixel, a mirror share is chosen.

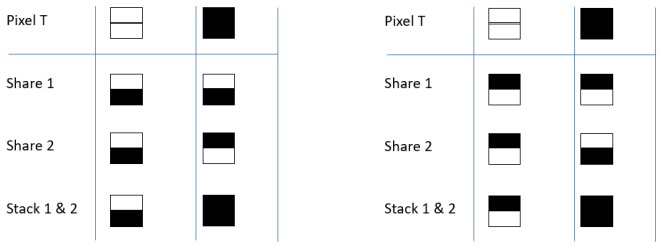


Fig. 1 Pixel Share Options

2.2 VES Perfect Color Use Limitation

In the VES scheme as described above, the visual effect of a black subpixel can not be undone by the color of the same subpixel in other shares, which are laid on it. Naor and Shamir call this property **monotonicity**. Security wise, it means that if one of the VES shares is perfectly black (or perfectly white), the stacked result is known in advance - if one of the shares is black, the stacked result is black; if one of the shares is white - the stacked result is equal to the second share value. This limitation example is shown in Figure 2 for 2 out of 2 VES. The solution for the monotonicity used by the existing VESs is to represent every subpixel in the share as a subpixel maps of size $m \times m$. The stacked black or white subpixels maps visually result in different grey level pixels. VES introduce threshold d to distinguish black and white colors.

3 Our Solution

3.1 Waves Interference for Perfect Output Visual Encryption Scheme

Our optical solution is based on waves interference and achieves Visual Encryption Scheme for perfect black and white images [1].

[1] One may consider RGB images by handling, the matrix corresponding to R, with values 0 when the pixel has no red component and 1 otherwise. Similarly, to green

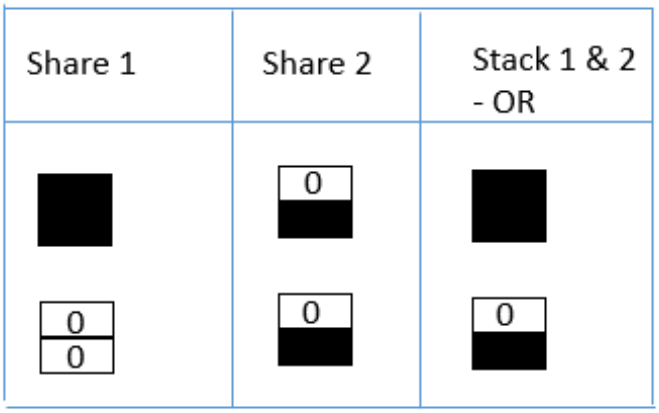


Fig. 2 VES Security Problem

We propose to use an existing physical model of waves interference to extend the VES computational model. The waves interference is a physical phenomenon in which, when waves meet in the space and time at the same point, their interaction creates an aggregated wave that can be of the same, greater or lower altitude. Constructive interference means that resulting wave has higher altitude, and destructive interference means that resulting wave has lower altitude. To maintain consistent and permanent waves interference, we have to ensure that the interacting waves have the same **length**.

We represent each VES share as wave signal. We introduce two additional parameters - **phase** and **amplitude** to control the waves. These additional parameters allow us to control the waves interaction, so it can result in either constructive or destructive signal. With the constructive signal we implement VES shares addition operator. The destructive signal allows us to implement VES shares subtraction operator. In the VES scheme based on addition and subtraction operators, we can achieve pixels of perfect white and black colors in both the secret image shares and the reconstructed image.

In Waves Interference VES, each pixel in the secret image is represented as a light beam (wave). The black pixel wave has an amplitude 1 and the white pixel has an amplitude 0. The original pixel is encoded into n shares. Each share is a light beam (wave) with amplitude A_i. Reconstructing the secret image from n shares is calculated according to the principle of superposition - $A = \sum_1^n A_i$. We are going to show that using waves constructive and destructive interference, A_i can be assigned in the way that the above calculation results in 1 or 0 values.

---

and blue components of the pixel, thus matching three binary matrices instead of one. More sophisticated schemes for fine tuned colors can be supported by adding more matrices.

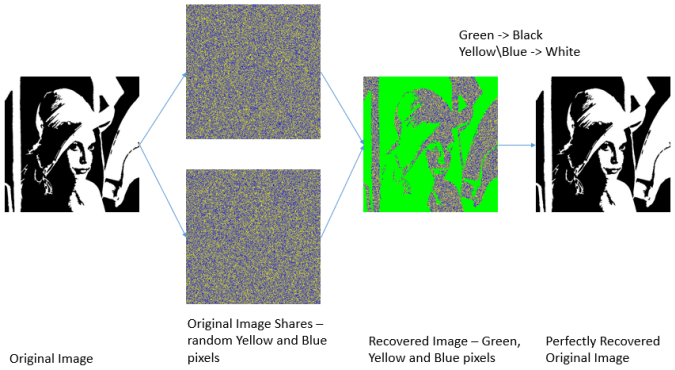
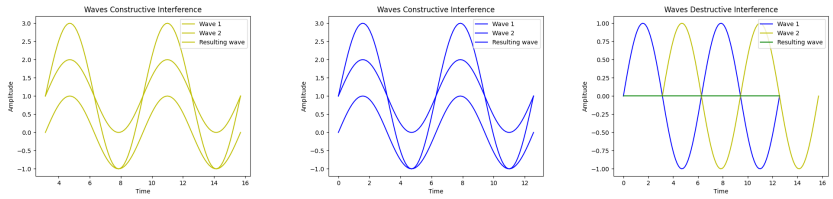


Fig. 3 VES Optical Model



(a) Constructive - Yellow Pixel (b) Constructive - Blue Pixel (c) Destructive

Fig. 4 Waves Interference Types

Lets look at the 2 out of 2 VES scheme. The possible pixel values and their superposition are shown in Figure 5. We denote the different amplitude waves with *yellow* and *blue* colors. The resulting superposition wave amplitude can be either *yellow* or *blue* when the shares' colors are the same; or *green* when the shares' colors are different. The *green* color represents constructive waves interference, while the *yellow* and *blue* colors represent destructive waves interference. The waves interference types are shown in Figure 4 - constructive for *yellow* pixel share, constructive for *blue* pixel share, and destructive for different pixel colors. The waves superposition represents Boolean *NOT XOR* operator, as shown in Figure 5 It should be noted that we use colorful pixel to describe the logical waves interference model. We can not use real optical wave colors since the interfering waves should have the same wave lengths for stable waves interference. The complete Optical VES Model is shown in Figure 3. First, we convert the original black and white secret image to two shares of randomly selected *blue* or *yellow* pixels as following - first share is randomly drawn from the first column in the table in Figure 5. If the original pixel is black, the second share pixel value is taken from lines 2 (if the first share was *blue*) or line

3 (if the first share was *yellow*). If the original pixel is white, the second share pixel value is taken from lines 1 or 4 - resulting in the same color for the both shares - either *blue* or *yellow*. The result is two shares of uniformly distributed, randomly selected *blue* or *yellow* pixels. An adversary that has access to one of the shares, has not enough information to recover the original pixel, thus the proposed VES is perfect information theoretical secure. To recover the original image, the shares are stacked. It results in the three colors pixel image - *blue* or *yellow* for the white pixels, and *green* for the black. The original image can be perfectly recovered by simple pixels mapping - the *blue* and the *yellow* pixels are mapped to the white pixels, and the *green* pixels are mapped to the black pixels, as shown in Figure 5.

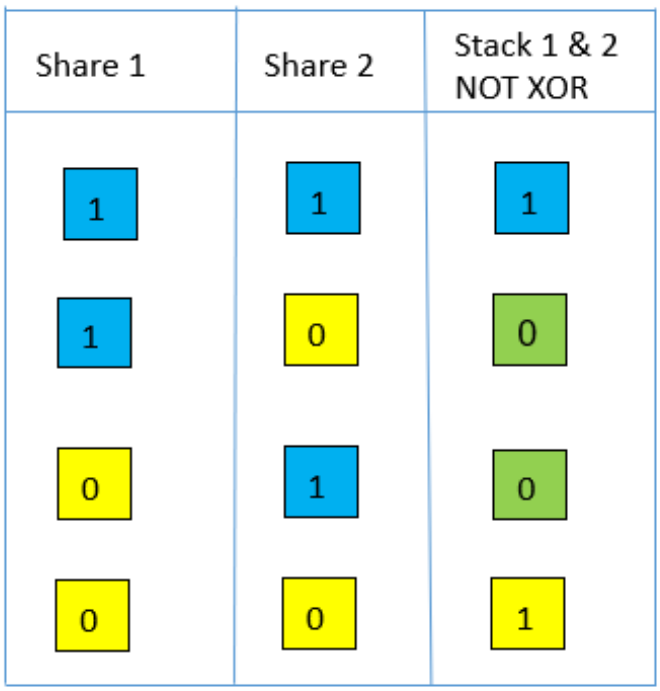


Fig. 5 Waves Interference Pixel Matrix

References

1. Jiao, S., Feng, J., Gao, Y., Lei, T., Yuan, X.: Visual cryptography in single-pixel imaging. Opt. Express **28**(5), 7301–7313 (Mar 2020).

https://doi.org/10.1364/OE.383240, http://opg.optica.org/oe/abstract.cfm?URI=oe-28-5-7301

2. Naor, M., Shamir, A.: Visual cryptography. In: Workshop on the Theory and Application of of Cryptographic Techniques. pp. 1–12. Springer (1994)

Scalable Video Coding for Satelite Video Multicast*
(Preliminary Version)

Alex Binun[1], Yefim Dinitz[1], Shlomi Dolev[1], Ofer Hadar[2], Adnan Jaber[1], and Shevach Riabtsev[2]

[1]Department of Computer Science, Ben-Gurion University of the Negev, Israel
[2]School of Electrical and Computer Engineering, Ben-Gurion University of the Negev, Israel

June 28, 2023

Abstract

The use of low Earth orbit satellites (LEO) for communication has become a reality (e.g., SpaceX). The usage of Internet communication communicating videos is very significant. We propose a scheme based on Scalable Video Coding (SVC) that fits multicat to users with heterogenous resolution demands (e.g., mobile phones, computer screens, and HD televisions). We build a hierarchy of optimal Steiner trees that are used to communicate layers of the SVC. The first Steiner tree spans all the terminals and is used to convey the first layer of the SVC, and the second spans the terminals that require more resolution than the basic resolution. The third Steiner tree spans the terminals that require even more resolution, and so forth for the following Steiner trees and SVC layers.

We suggest a new algorithm for finding the Steiner trees in the hierarchy, such that they are all optimal and prefer edges not used by the Steiner trees that are used for previous layers. Thus, distributing the communication without sacrificing optimality.

1 Introduction

Broadband satellite multimedia (BSM) systems will be an integral part of the global information infrastructure as one of the major technologies providing both broadband access and broadcast services [ST02]. Recent commercial deployments show that users not only would like to have access to value-added services (e.g., mobile Internet, multimedia streaming, etc.) but are also willing to pay more for them and, in particular for video services (Sattler). The introduction of video coding technology in the satellite application space opens up new and challenging topics; digital video applications have to face potentially harsher transmission environments than the ones they were originally designed to work with (e.g., HDTV, Mobile TV), especially as regards traversing packet networks with the presence of satellite links. Towards approaching the satellite multimedia application delivery needs, H.264/MPEG4 Advanced Video Coding (AVC) [OW04] as the latest entry of international video coding standards has demonstrated significantly improved coding efficiency, substantially enhanced error robustness, and increased flexibility.

Video Transmission. Media and its transmission have undergone significant transformation in recent years. Just a few decades ago, it was unimaginable that people could launch their own media and broadcast it to a wide audience. However, with that media transition comes a revolution in screen qualities (full hd, hd, etc..), and since users are highly diverse, it is hard to meet all of their requirements. One of the main strategies to fulfill users' demands is simulcast, where videos are sent in different resolutions (1080p, 720p, 360p). After receiving the three streams, the SFU selects one based on the terminal performance and forwards the streams to the terminal.

*Research supported by SATELLITE BGU-GILAT magneton grant, contact author: Adnan Jaber, adnanjaber89@gmail.com.

Another strategy out there is SVC (Scalable Video Coding). SVC deals with the problem in video multicast caused by user heterogeneity and attracted more and more attention in recent years. Specifically, SVC encodes a video stream into one base layer and multiple enhancement layers. The base layer carries the essential information and provides a minimum quality of the video, and the enhancement layers represent the same video with gradually increasing quality. The decoding of a higher enhancement layer is based on the base layer and all its lower enhancement layers [KPH15]. By multicasting an SVC-based video, users with higher channel quality can decode more layers to retrieve a video with higher quality, and users with worse channel quality can decode fewer layers to retrieve a video with a lower quality.

Scalable Video Coding. In general, a video bit stream is called scalable when parts of the stream can be removed in a way that the resulting substream forms another valid bit stream for some target decoder, and the sub-stream represents the source content with a reconstruction quality that is less than that of the complete original bitstream but is high when considering the lower quantity of remaining data. Bitstreams that do not provide this property are referred to as single-layer bit streams. The usual modes of scalability are temporal, spatial, and quality scalability.

Spatial scalability and temporal scalability describe cases in which subsets of the bit stream represent the source content with a reduced picture size (spatial resolution) or frame rate (temporal resolution), respectively. With quality scalability, the substream provides the same spatiotemporal resolution as the a complete bit stream, but with a lower fidelity – where fidelity is often informally referred to as signal-to-noise ratio (SNR). Quality scalability is also commonly referred to as fidelity or SNR scalability. The different types of scalability can also be combined so that a multitude of representations with different spatiotemporal resolutions and bit rates can be supported within a single scalable bit stream.

Efficient, scalable video coding provides several benefits in terms of applications a few of which will be briefly discussed in the following. Consider, for instance, the scenario of a video transmission service with heterogeneous clients, where multiple bit streams of the same source content differing in coded picture size, frame rate, and bit rate should be provided simultaneously. With the application of a properly configured scalable video coding scheme, the source content has to be encoded only once – for the highest required resolution and bit rate, resulting in a scalable bit stream from which representations with lower resolution and/or quality can be obtained by discarding selected data. For instance, a client with restricted resources (display resolution, processing power, or battery power) needs to decode only a part of the delivered bit stream. Similarly, in a multicast scenario, terminals with different capabilities can be served by a single scalable bit stream. Another benefit of scalable video coding is that a scalable bitstream usually contains parts with different importance in terms of decoded video quality. This property, in conjunction with unequal error protection, is especially useful in any transmission scenario with unpredictable throughput variations and/or relatively high packet loss rates. By using stronger protection of the more critical information, error resilience with graceful degradation can be achieved up to a certain degree of transmission errors.

Media-aware network elements (MANEs), which receives feedback messages about the terminal capabilities and/or channel conditions can remove the nonrequired parts from a scalable bit stream before forwarding it. Thus, the loss of essential transmission units due to congestion can be avoided, and the overall error robustness of the video transmission service can be substantially improved. Scalable video coding is also highly desirable for surveillance applications, in which video sources not only need to be viewed on multiple devices ranging from high-definition monitors to videophones or PDAs, but also need to be stored and archived. With scalable video coding, for instance, high-resolution/ high-quality parts of a bit stream can ordinarily be deleted after some expiration time so that only low-quality copies of the video are kept for long-term archival. The latter approach may also become an interesting feature of personal video recorders and home networking.

Even though scalable video coding schemes offer a variety of valuable functionalities, the scalable profiles of existing standards have rarely been used in the past, mainly because spatial and quality scalability has historically come at the price of increased decoder complexity and significantly decreased coding efficiency. In contrast to that, temporal scalability is often supported, e.g., in H.264/AVC-based applications, but mainly because it comes along with a substantial coding efficiency improvement. H.264/AVC is the most recent international video coding standard. It provides significantly improved coding efficiency in comparison to all standards.

2 Scalable Video Coding Formats

H.264/SVC. Advanced Video Coding (AVC), also referred to as H.264 or MPEG-4 Part 10, is a video compression standard based on block-oriented, motion-compensated coding. It is by far the most commonly used format for recording, compressing, and distributing video content, used by 91% of video industry developers as of September 2019. It supports resolutions up to and including 8K UHD. The intent of the H.264/AVC project was to create a standard capable of providing good video quality at substantially lower bit rates than previous standards (i.e., half or less the bit rate of MPEG-2, H.263, or MPEG-4 Part 2), without increasing the complexity of design so much that it would be impractical or costly to implement.

H.265/HEVC. HEVC was designed to substantially improve coding efficiency compared with H.264/MPEG-4 AVC HP, i.e., to reduce bitrate requirements by half with comparable image quality at the expense of increased computational complexity. HEVC was designed with the goal of allowing video content to have a data compression ratio of up to 1000:1. Depending on the application requirements, HEVC encoders can trade off computational complexity, compression rate, robustness to errors, and encoding delay time. Two of the key features where HEVC was improved compared with H.264/MPEG-4 AVC are the support for higher resolution video and improved parallel processing method.

AV1-VP9. AV1 is a traditional block-based frequency transform format featuring new techniques.

Based on Google's VP9, AV1 incorporates additional techniques that give encoders more coding options to enable better adaptation to different input types.

Types	**MPEG2/H.262 (1995)**	**AVC/H.264(SVC) (2007)**	**HEVC/H.265(SHVC) (2014)**	**AV1/VP9 (2018)**
Scalability	Quality, Spatial	Quality, Spatial, Temporal	Quality, Spatial, Temporal	Quality, Spatial, Temporal
Decoder complexity	Low	Low	High	High
Implementation	Easy	Hard	Easy	Easy
Loop control	one layer	Hybrid	Multi	Multi

Figure 1: SVC Formats

3 Experiment with Scalable Video Coding

In the next experiment, we consider simple spatial scalable coding with three spatial resolutions (720p, 360p, 144p). Thus, the base layer (layer 0) represents a 144p sequence with a frame rate of 25 Hz. In the first and second enhancement layers, a 360p and 720p sequence with a frame rate of 25 Hz is coded. The hierarchical coding structure with two spatial layers is illustrated in Figure 2.

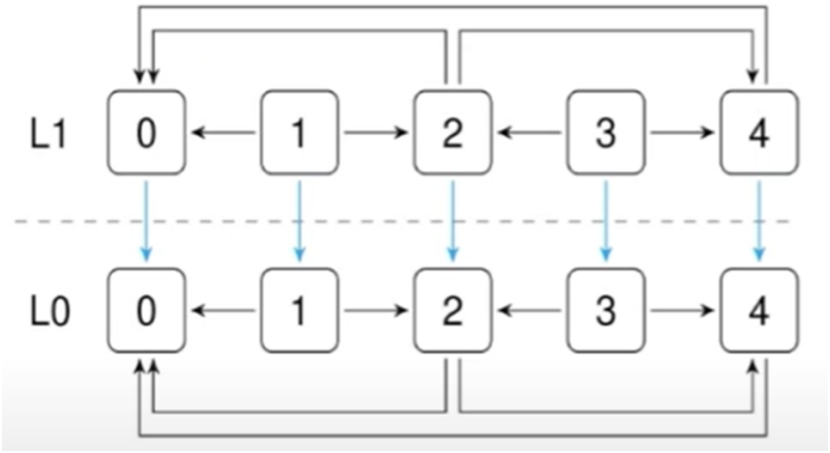


Figure 2: structure of two spatial layers

In the first step, the resampling tool is used for generating spatially and temporally downsampled sequences; at this next step, a configuration file that contains information about each layer and a main configuration file that contain details about the hierarchy and the strategy are given to the Encoder as input. In the end, the corresponding encoder output is shown, which summarizes the supported spatiotemporal resolutions and bit rates.

```
Contained Layers:
====================

      Layer   Resolution   Framerate   Bitrate MinBitrate      DTQ
        0     256x144       6.2500     144.70      144.70   (0,0,0)
        1     256x144      12.5000     150.70      150.70   (0,1,0)
        2     256x144      25.0000     155.20      155.20   (0,2,0)
        3     640x368       6.2500     761.70      761.70   (1,0,0)
        4     640x368      12.5000     798.30      798.30   (1,1,0)
        5     640x368      25.0000     830.00      830.00   (1,2,0)
        6     1280x720      6.2500    3016.00     3016.00   (2,0,0)
        7     1280x720     12.5000    3170.00     3170.00   (2,1,0)
        8     1280x720     25.0000    3300.00     3300.00   (2,2,0)
```

Figure 3: Spatiotemporal Layers stack

Note that each layer depends on and contains all of the previous layers, so as a next step, Demultiplexer for SVC-DASH is used to Splits the SVC bitstream into chunks, one per layer [GTH+13]. At this step's end, each layer can be sent independently to the next end. at the client Re-Multiplexer for SVC-DASH. Reorders SVC layer chunks into a single SVC bitstream.

The next table shows the bandwidth of the original file and each of the spatial layers.

file	size	frame rate
original	2.13MB	25
layer 0	140KB	25
layer 1	738KB	25
layer 2	2933KB	25

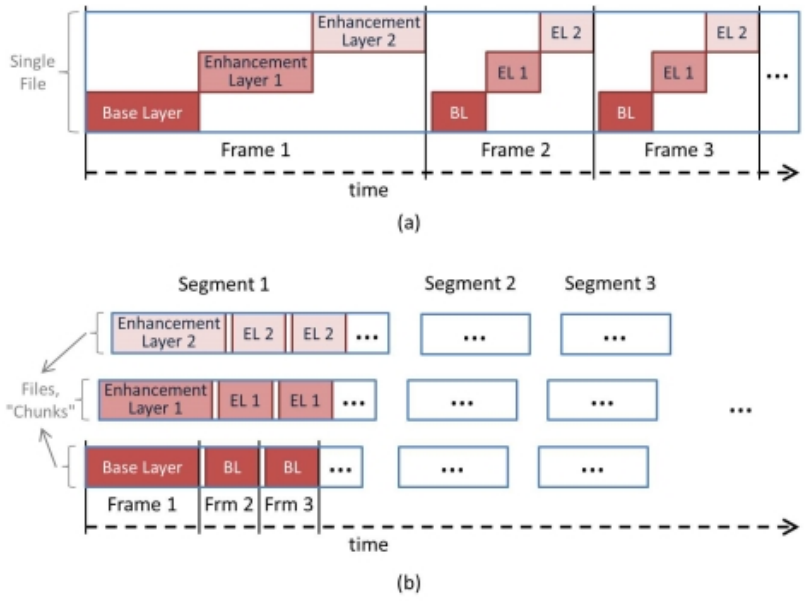


Figure 4: Btstream-Reorder

4 Hierarchical Optimal Steiner Trees, Avoiding Tree Edges Overlaps

The Steiner tree problem is one of the best-studied problems in Computer Science [RK23]. Given a connected graph $G = (V, E)$ on $n = |V|$ nodes, edge weights $w : E- \to R+$, in fact, we restrict the weights here to be positive integers, and a set $S \subseteq V$ of k terminals, the objective is finding a subtree ST of G spanning S such that the weight $w(ST) := \sum_{e \in ST} w(e)$ of ST is minimized. In the cardinality version of the problem, all edge weights are one.

Steiner tree problem involves finding the minimum-cost tree that connects a given set of points in a graph. In this problem, we have a set of "terminal" points that we must connect, and we can also introduce additional "Steiner" points to minimize the total length or cost of the tree.

The objective of the Steiner tree problem is to find the optimal set of edges that connect all the terminal points while minimizing the total length or cost. The tree can be constructed by adding extra Steiner points to the graph, allowing for more flexibility in connecting the terminals.

The Steiner tree problem has numerous applications, such as in network design, VLSI circuit routing, wireless sensor networks, and geographic information systems. By finding the minimum-cost tree that connects a given set of points, the Steiner tree problem helps optimize various real-world systems where efficient connections are essential.

Finding an exact algorithm for the Steiner tree problem is a daunting task because it falls under the category of NP-hard problems, implying that it is computationally challenging to solve for large instances in a reasonable amount of time. Despite this difficulty, there are two popular exact algorithms for the Steiner tree problem, Integer Linear Programming (ILP) and Branch and Bound.

The branch and bound algorithm explores the solution space by systematically branching on edges or potential Steiner points, creating subproblems. It evaluates the feasibility and cost of each subproblem, considering connectivity requirements and calculating the total length of the tree. Through pruning techniques, it discards subproblems that are deemed inferior based on their evaluated cost, avoiding unnecessary computations. The algorithm continues branching and evaluating subproblems, iteratively narrowing down the search space until an optimal solution is found.

ILP formulates the Steiner tree problem as an optimization problem using binary variables to represent the presence or absence of edges in the tree. The objective function aims to minimize the total length or cost of the Steiner tree, subject to constraints that ensure connectivity and the inclusion of necessary Steiner points. ILP solvers, such as CPLEX or Gurobi, can be utilized to solve the formulated problem by employing advanced optimization techniques. ILP provides a rigorous and

exact approach to finding the optimal Steiner tree solution, but its computational complexity increases significantly with larger problem instances.

reduction techniques are the most important ingredient in a state-of-the-art SPG solver. Using linear programming and branch-and-bound SCIP-JACK [RK23] implement the branch-and-cut approach. Branch-and-cut combines the strengths of branch and bound for exploring the solution space and linear programming relaxation for obtaining lower bounds and cutting planes.

building on that, we suggest a new algorithm for finding the Steiner trees in the hierarchy, such that they are all optimal and prefer edges not used by the Steiner trees that are used for previous layers.

Algorithm 1 Herarchical Steiner Tree (HST) Algorithm (finding ST_1, ST_2, ST_3)

Require: $S_3 \subseteq S_2 \subseteq S_1$ Sets of Terminals, in fact, we can continue with more subsets
Ensure: find optimal Steiner tree ST_1 where $G = (V, E)$ and $S_1 \subseteq V$ is a finite set of terminals.
  for each $(E \in G)$ **do**
    $W(E_1) \leftarrow W(E) \times |V|$
  end for
  for each $(E \in ST_1)$ **do**
    $W(E_1) \leftarrow W(E) + 1$
  end for
Ensure: find optimal Steiner tree ST_2 where $G = (V, E_1)$ and $S_2 \subseteq S_1$
  for each $(E \in ST_2)$ **do**
    if $E \in ST_2$ AND $E \notin ST_1$ **then**
      $W(E_2) \leftarrow W(E) + 1$
    end if
  end for
Ensure: find optimal Steiner tree ST_3 where $G = (V, E_2)$ and $S_3 \subseteq S_2$
  return (ST_1, ST_2, ST_3)

To deal with ST_i where i is greater than 3, we repeat the actions made to the previous ones making sure that all edges used in previous STs are with a weight that is 1 more than the $|V|$ times their original weight in E. Then we apply the optimal Steiner tree finder.

In this way, the optimal Steiner tree finder finds in each iteration i an optimal Steiner tree for S_i over G as it prefers an optimal Steiner tree that chooses already used edges that may have the weight of $|V| - 1$ additional 1 greater than the optimal weight multiplied by $(|V|)$ (optimal weight multiplicated, OWM) over a non-optimal tree that has at least $|V|$ more weight than OWM, as one non-optimal edge implies the addition of at least $|V|$ to the tree weight.

4.1 Implemntaion and results

We have implemented Hierarchical Stainer Tree (HST) algorithms that search for N hierarchial Steiner trees in a weighted graph G. Giving a weighted graph G and a set of terminals $S_1, S_2, S_3 ... S_n$ where $S_n \subseteq S_n - 1 .... \subseteq S_2 \subseteq S1$ HST finds efficiently N optimal Steiner trees and prefers edges that are not used by the previous trees. We base our solution on the SCIP-Jack [GKM+17] solver.

HST was tested on different graph scales and various terminals number. It comes out that for a 50x50 mesh grid, we can find at least 4 hierarchical optimal Steiner trees in a good performance, only in one second. worse case performance finding optimal Steiner tree in different mesh grid graph scales shown in Figure 5, HST has better performance on small-scale graphs.

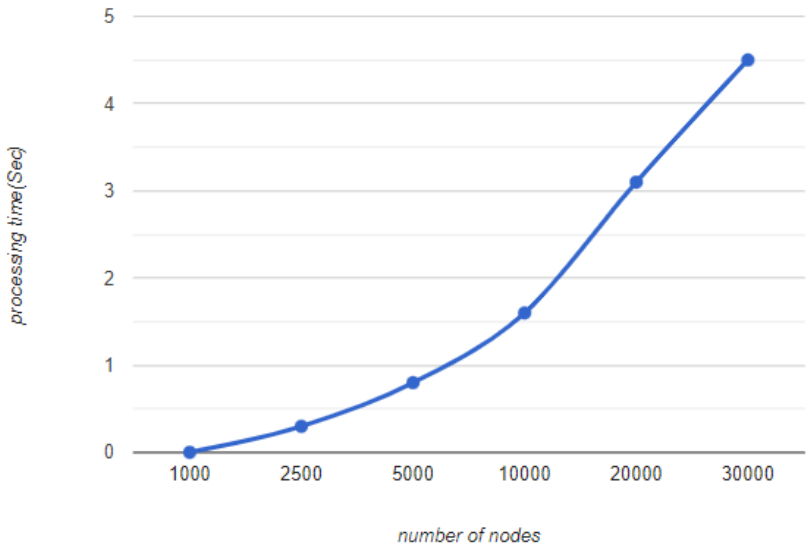


Figure 5: Prosscing time in seconds

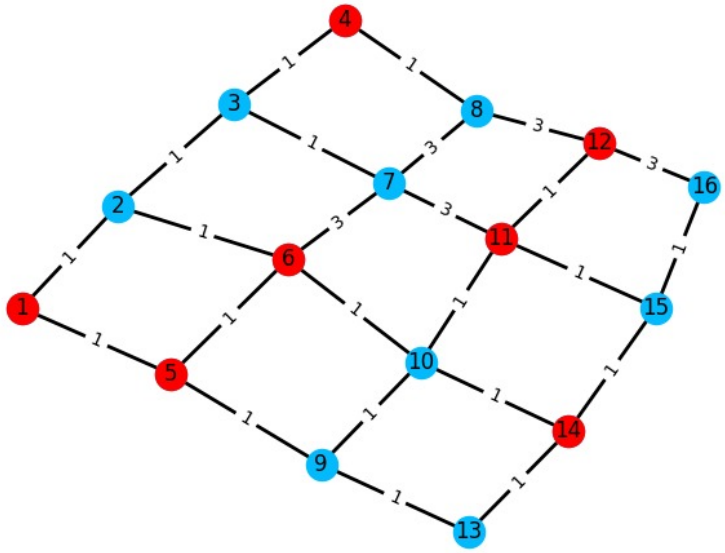


Figure 6: Orginal Graph: 4x4 Mesh-Grid

Given a 4×4 weighted graph and a set of terminals marked in red Figure 6 and Figure 7 respectively S_1={1, 4, 5, 6, 11, 12, 14} and S_2={1, 5, 11, 14}. The output was two optimal Steiner trees Figure 8 is the first hierarchical Steiner tree weighted 9 and Figure 10 is the second weighted 5.

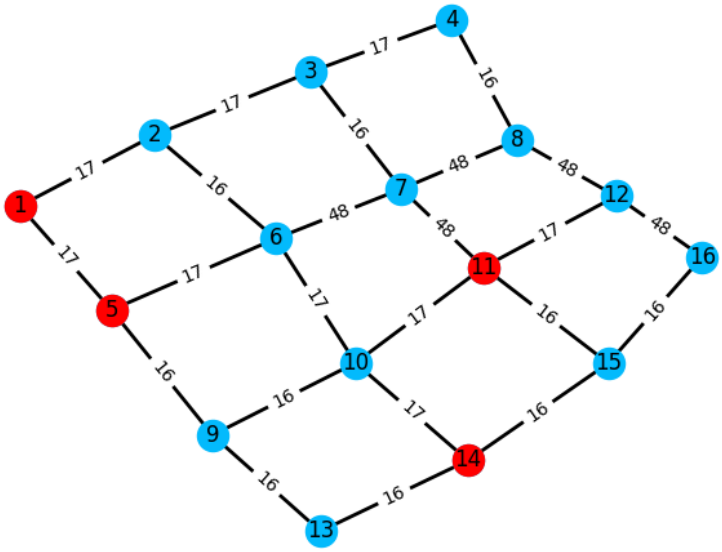


Figure 7: Graph weights after applying algorithm

In the first step of the algorithm, we search for the first optimal Steiner tree giving original graph weights and terminals set S_1. while in the next step, we change the original graph weight Figure 7 to avoid using the same edges of the first tree. the weights of the graph changed as follows each edge in the graph multiplied by the nodes number of the graph, in our case 16. and each edge used in the previous tree increased by one.
Applying HST on the graph resulted in having an optimal Steiner tree that used two odd edges, highlighted in green 9. while all the edges of the optimal tree without applying HST 10 have been used by the previous tree 8. We note that we can search for several optimal Steiner trees in the first layer, if there are two such optima Steiner trees then we can use E_1 and the same terminal, to check whether another optimal tree exists, then we can continue with each of them to further explore next layers. We defer more exposition of this approach to the full version.

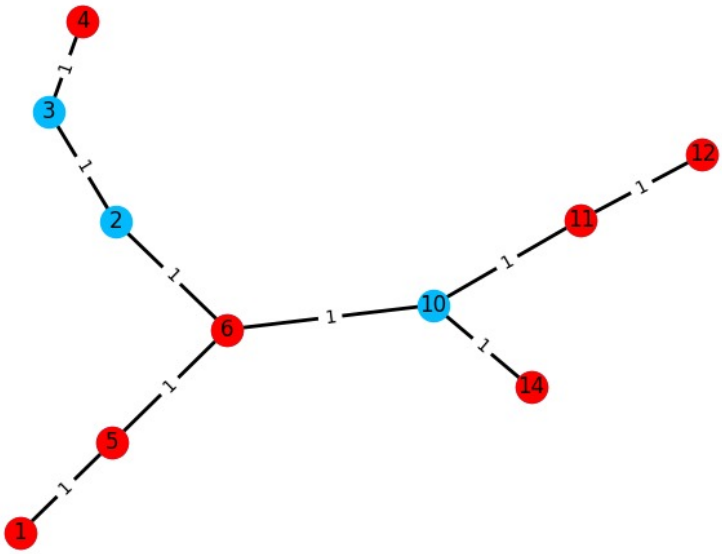


Figure 8: first hierarchical Steiner Tree

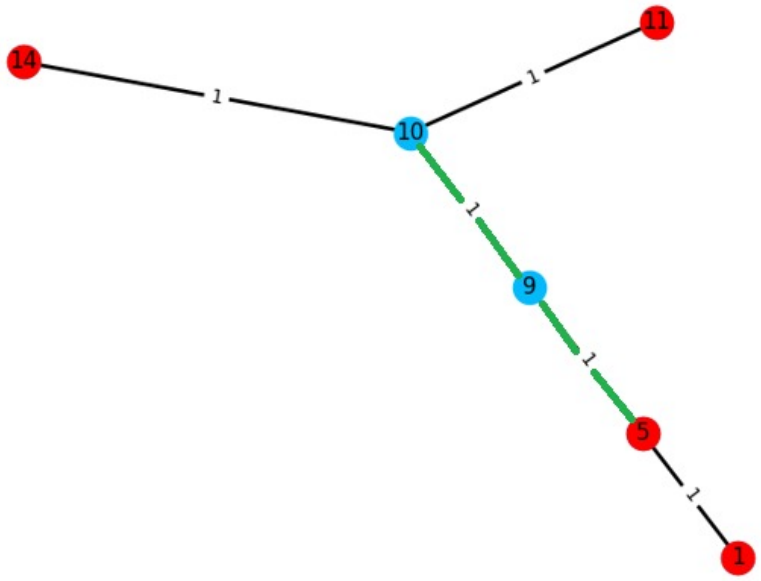


Figure 9: The second hierarchical Steiner Tree using HST

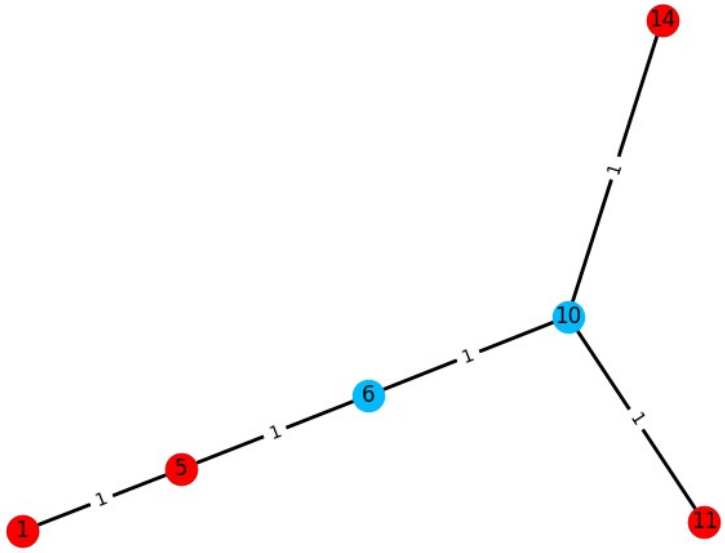


Figure 10: The second hierarchical Steiner Tree

References

[GKM+17] Gerald Gamrath, Thorsten Koch, Stephen J. Maher, Daniel Rehfeldt, and Yuji Shinano. Scip-jack - a solver for STP and variants with parallelization extensions. *Math. Program. Comput.*, 9(2):231–296, 2017.

[GTH+13] Michael Grafl, Christian Timmerer, Hermann Hellwagner, Wael Chérif, and Adlen Ksentini. Evaluation of hybrid scalable video coding for http-based adaptive media streaming with high-definition content. In *IEEE 14th International Symposium on "A World of*

Wireless, Mobile and Multimedia Networks", WoWMoM 2013, Madrid, Spain, June 4-7, 2013, pages 1–7. IEEE Computer Society, 2013.

[KPH15] Christian Kreuzberger, Daniel Posch, and Hermann Hellwagner. A scalable video coding dataset and toolchain for dynamic adaptive streaming over http. In Tsang Ooi Wei, editor, *Proceedings of the 6th ACM Multimedia Systems Conference*, MMSys '15, pages 213–218, New York, NY, USA, mar 2015. ACM.

[OW04] Jörn Ostermann and Axel Weissenfeld. Talking faces - technologies and applications. In *17th International Conference on Pattern Recognition, ICPR 2004, Cambridge, UK, August 23-26, 2004*, pages 826–833. IEEE Computer Society, 2004.

[RK23] Daniel Rehfeldt and Thorsten Koch. Implications, conflicts, and reductions for steiner trees. *Math. Program.*, 197(2):903–966, 2023.

[ST02] Harald Skinnemoen and Hillar Tork. Standardization activities within broadband satellite multimedia. In *IEEE International Conference on Communications, ICC 2002, April 28 - May 2, 2002, New York City, NY, USA*, pages 3010–3014. IEEE, 2002.

Entrepreneurship Pitch Track chaired by: Yonah Alexandre Bronstein

Entrepreneurship Pitch Track chaired by: Yonah Alexandre Bronstein

The Hi-Tech industry and state-of-the-art research are getting ever closer, as shown by the overlap between the PhD track and Entrepreneurship track this year. The goal of the CSCML Pitch Track is to expose researchers to the world of entrepreneurs and vice versa, for the sake of creating mutual value and advancing the economy and society. Seven startups pitched this year focusing mainly on Threat Protection, although the *scope* of the threats was quite broad: Cyber threats in software and in the web, Physical Infrastructure threats, Biological threats (hacked DNA is no longer science-fiction), and Human threats in Health Care or Elder Care! It was heartening, this year as well as in past years, to note that, even in a business focused track, there were entries that could justifiably be considered "for the greater good of the people" – that is, even if they had business motives and priorities, they would still end up benefiting all of us, such as Smart Traffic management and Efficient Polymers in building construction… All these entrepreneurs deserve all the encouragement that we in the community can give them, in whatever form is suitable. As was the case last year, the Entrepreneurship Pitch track at CSCML 2023 did an excellent job of fulfilling this objective and consequently was a great success. It received endorsement from leading VCs and corporations (IBM, Microsoft, Checkpoint…).

Out of the seven start-ups pitched during CSCML 2023, three were selected as finalists: "Xplorisk - Automated Web3 Risk Management" led by Shiran Kleiderman and Snir Levi, was selected by the Entrepreneurship Pitch Track Committee and the audience as the leading entry and won the $500 prize. "Amaze," presented by Hannah Yair – which some pitch-in help from Prof. Dolev - which aims to improve traffic efficiency at intersections, received second place. Tied for second place was "AntisepTech," led by Barak Katz, who implemented a nice pivot from the Corona focused pitch in the recent past, to a much older and pervasive issue unfortunately, of human abuse in health care or elder care…

Overall, the quality and value of the start-ups who presented was quite impressive. And I look forward to future CSCML conferences in the years to come, still on Zoom as IMHO, it makes it so much easier and cheaper for people around the world to present (and attend ☺).

Best regards,

Yonah Alexandre Bronstein

Entrepreneurship Pitch Track Chair

DETEXT Team

Security Layer for privacy and confidentiality

Co-Founder / CTO

Yuri Shterenberg

B.sc in Software Engineering
DevOps Architect/Team lead/
CyberSecurity specialist in
Cybersecurity company

Co-Founder / CEO

Idan Proshtisky

B.A in Information System Management
Project Manager/Service Team lead

Head of Research

Dr. Nadav Voloch

CyberSecurity Researcher at IMT School of advanced studies, Lucca, Italy
Co-founder of BrainPM.com- AI project management
Cyber and Software lecturer at Ruppin Academic Center

The Problem

Sensitive business information is any data that would pose a risk to the company if exposed to unauthorized factors

Personal Information

ID number
Name
Residential address
Email address
Any demographic details

Sensitive Information

Medical history
Political opinions
Genetic or biometric information
Personal equity
Communication data

Confidential Information

Any kind of textual data that can be classified as secret.
Documents and README files containing passwords, users, urls and access keys.

The Problem

Today, over half of the companies' digital textual information is exposed to **unauthorized** factors. This is due to the fact that there is no solution combining user **access control** and sensitive **data detection**.

Corporate internal information

Email:david@company.com
Phone: (555) 555-1234
Card:1234-5678-1234-1234
Despite earning $12.5B in revenue

Secrets in files

```
$db = new Db();

// CHECK FOR BYPASS
if($email == '123@spe.com' && $password == '123123') {
    $this->bypass($email, $password);
}

// check for valid user
$user = new User();
$uid = $user->get_uid($email, $password);
if(empty($uid)) {
    $this->error('invalid login', '03');
    exit;
}
```

Multiple CMS Systems

Absence of authentication

Anyone at any time can access sensitive information if it is at his disposal

Current available solutions

- Online training platform
- Legally binding contract between two or more parties (NDA)
- Whitelisting destination domains (data in transit)

The Solution

detext.io

Identification and protection of sensitive parts of textual information

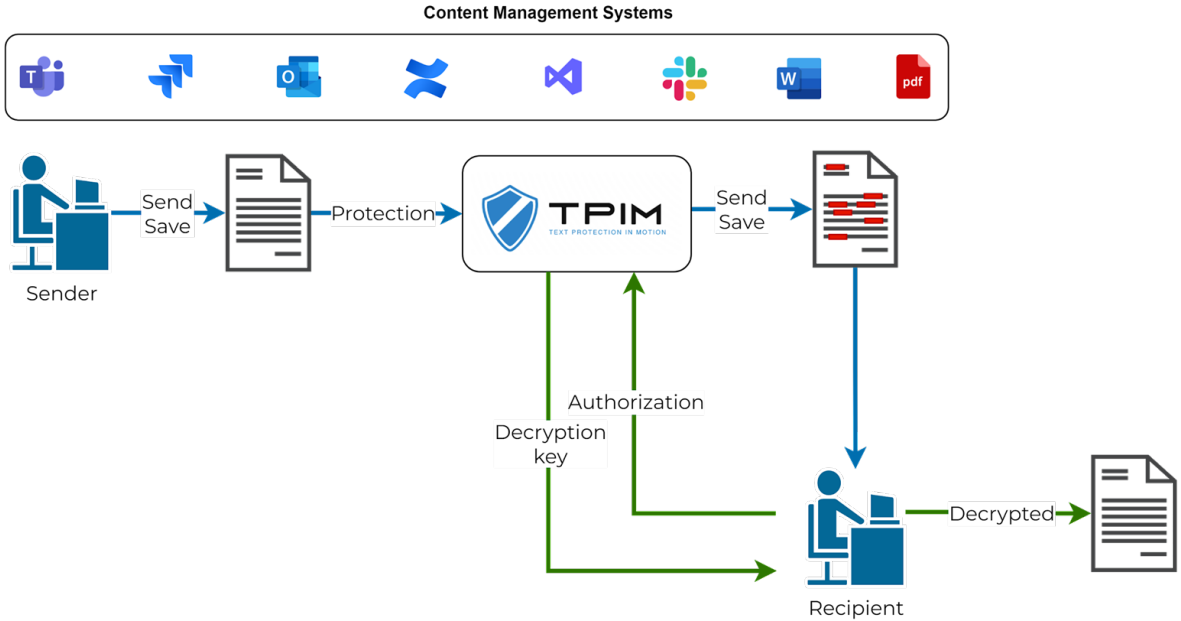
Detection of potential weakness

Protect sensitive data

Access Control

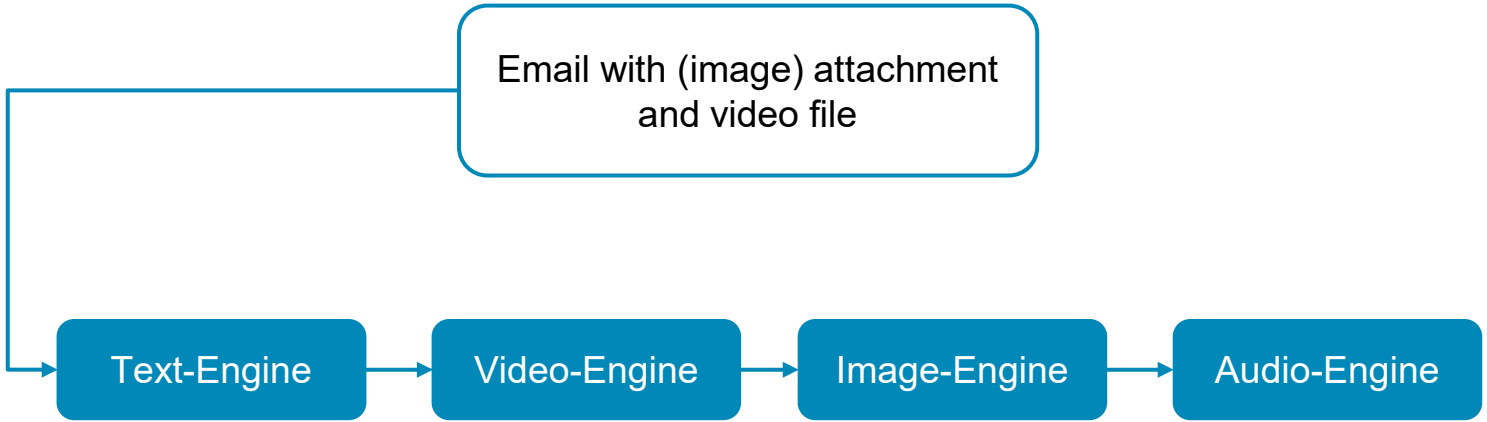
The Product

Our platform provides businesses with an ultimate solution to protect sensitive textual data, by guaranteeing that only authorized users can access it.



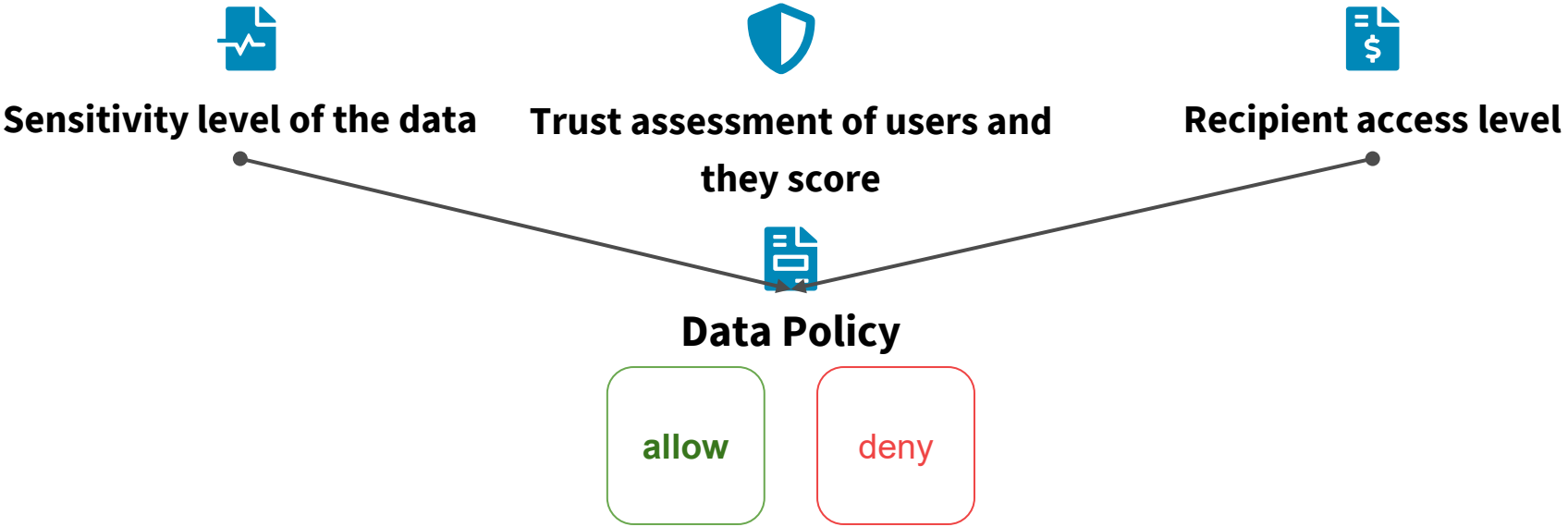
The Innovation

Our unique technique for parsing and extracting sensitive information from multi-format data structures.



The Innovation

Our unique algorithm allows for the secure sharing of information, with access to sensitive data determined by the recipient's pre-defined parameters.



Our Achievements So Far

- We have done validation with 4 companies -link to the validation PP
- Designed Partner- recently we started to run our alpha product with design partner(who took apart in our validation forum)- in this link we can see RTD with all the encrypted data
- A part of the **NVIDIA** acceleration program
- Collaboration with **Ruppin Academic Center**

Private & Confidential

Xplorisk

Automated & Ongoing Web3 Risk Management

June 2023

Xplorisk Team

- Pioneering compliance and security in the Web3 industry
- Proven track record in successfully building security departments, products, and companies

Snir Levi, Co-Founder & CTO

- Head of Security Data Science & R&D, Celsius
- CTO & Risk Manager, Financial Immunities
- CTO & Financial Consultant, Precise
- IDF Commander, Iron Dome Core Team

Shiran Kleiderman, Co-Founder & CEO

- CSO & Head of IT, Celsius
- Founding Team Member & CTO Dark Web Intelligence, BlueVoyant
- CTO Cyber Threat Intelligence, K2 Integrity (acq.)
- Co-Founder, Aegis Bitcoin Wallet
- IDF Commander, 8200 & Matzov

Advisors:
-Eran Reshef (Sanctum/IBM, Skybox/Providence Eq., Collactive/IAI),
-Uri Stav (DCG, Genesis, Crypto Expert),
-Dr. Udi Levi (Prime Minister's Office, Banking Sector)

For Web3 to be widely adopted, it's essential to

Have a forward looking approach for compliance & security

Establish mature risk mitigation frameworks

Improve client asset safeguarding controls

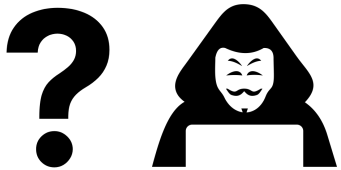
Control your Web3 environment

Your Web3 Environment and Dependencies

Do you know your assets as well as the criminal groups and hackers?

Compliance & Security Officers

LACK CONTROL OF THEIR WEB3 ENVIRONMENT



Additional Sources & Counterparties

Client Mobile/Web Apps + APIs

On & Off Ramp Services

Internal Deployment Wallets/ Network

Custodians

Exchanges

DeFi

Staking

Mining

OTC

Lending

External Asset Managers

Chains, Bridges & "Hops"

Social Media

Domains

GitHub/Code

Nodes

Validators

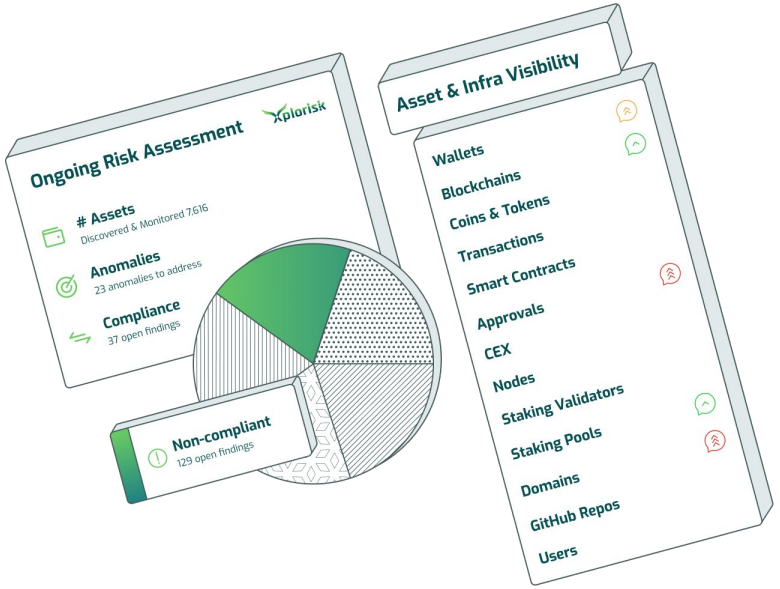
- # The Web3 System of Record

With the Supporting Web 2.0 Components

The Solution

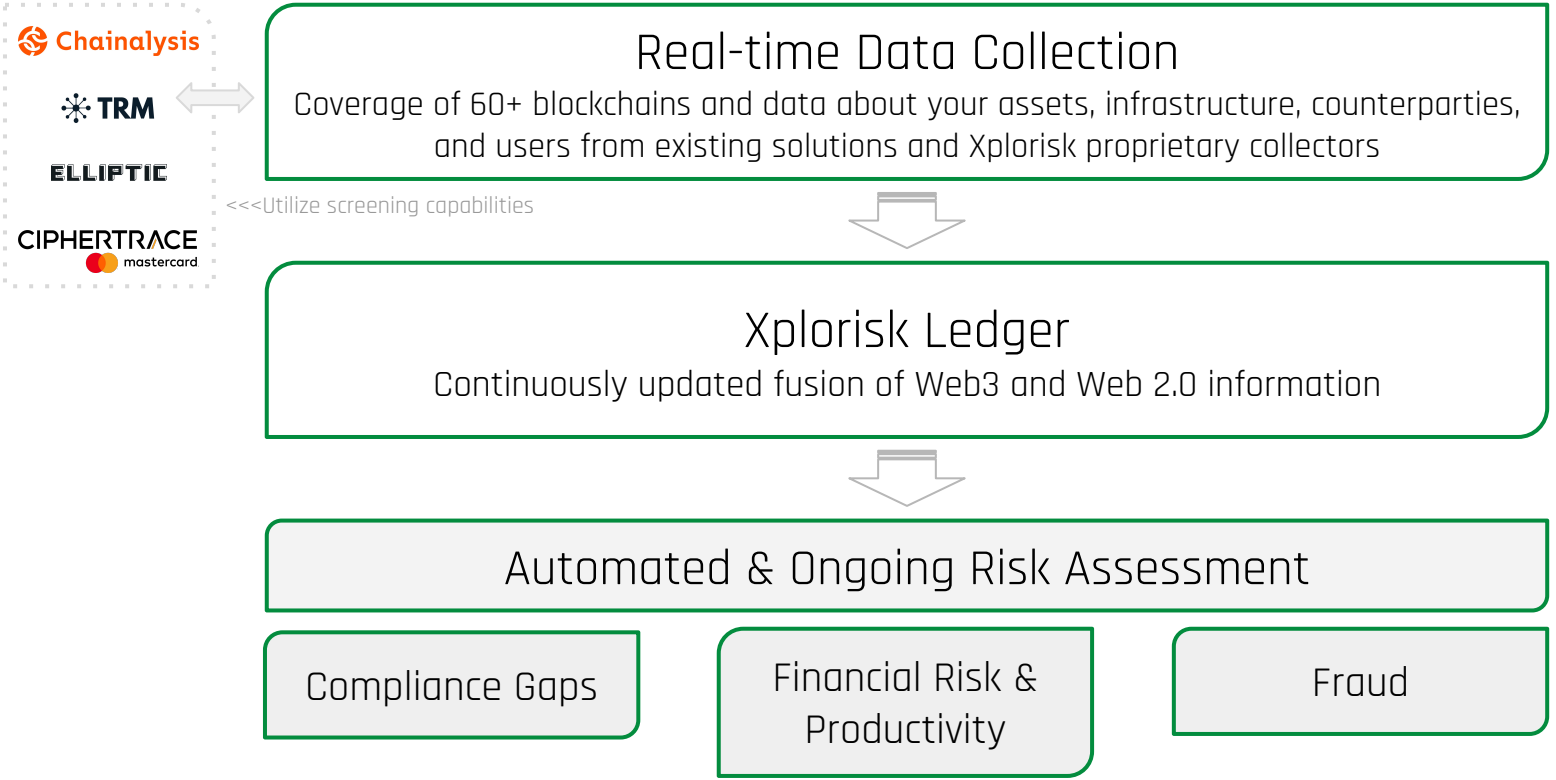
Xplorisk Ledger &
Ongoing Risk Assessment

A single view of your entire
Web3 environment and user base
via our SaaS, API, and Workflow Automation

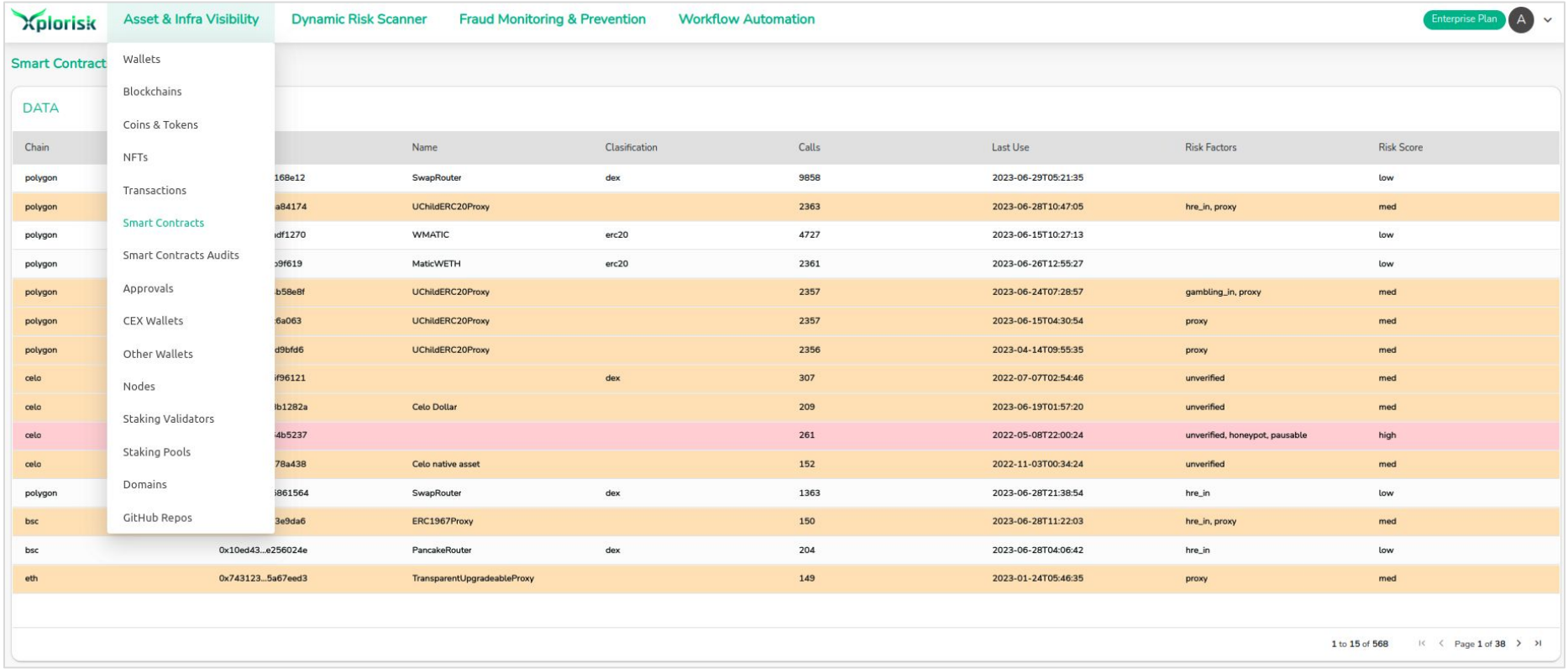


WEB3 ON-CHAIN > WEB3 OFF-CHAIN > WEB 2.0 & FIAT

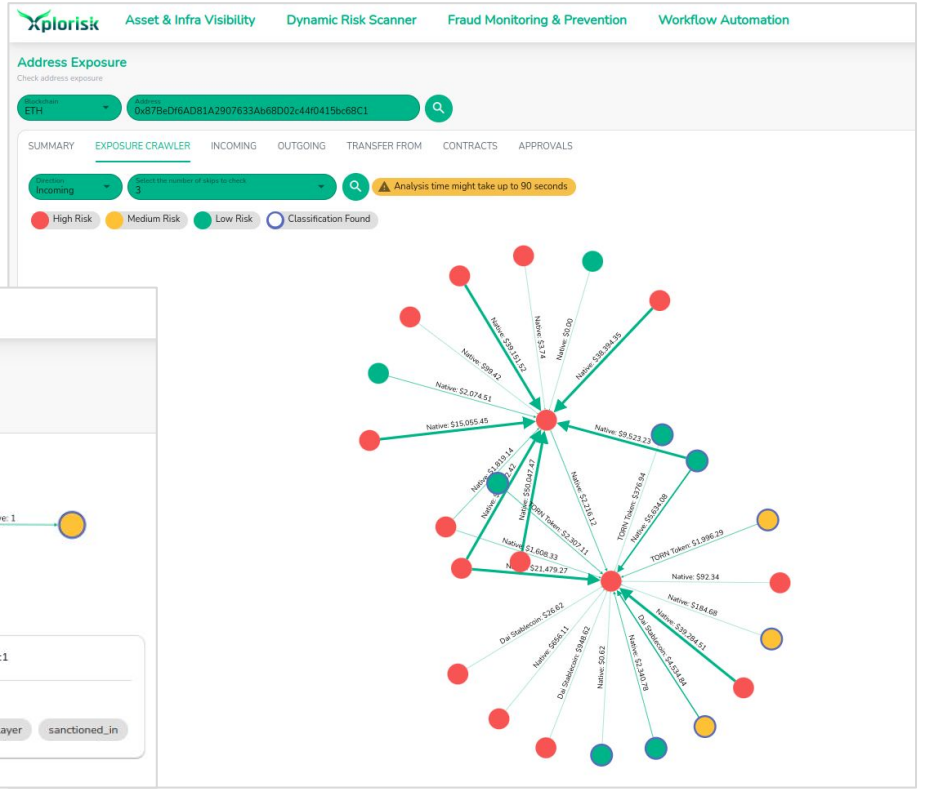
Collaborative Vision & Platform

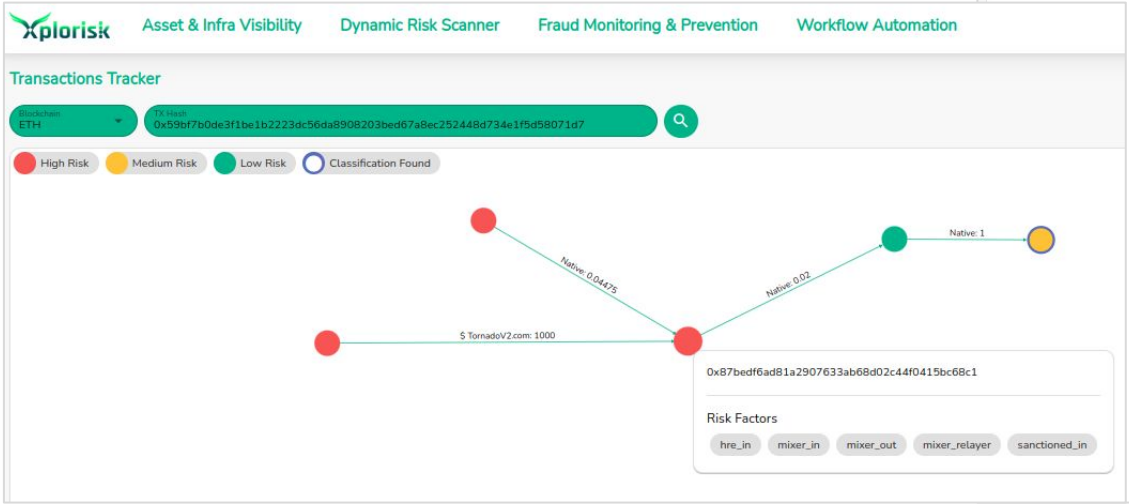


Complete Asset Visibility & Control



Behavioral & Automated Risk Assessment





Target Audience

TradFi & CeFi

- Exchanges & CeFi
- Banking Crypto Arms
- Custodians
- Hedge Funds
- Payment Platforms
- Web 2.0 Companies with Web3 Interaction

DeFi

- Platforms & Protocols
- Bridges
- Staking & Mining
- DAOs

Regulators, Auditors, Government, and Law Enforcement

Stakeholders:

Chief Compliance Officers, Risk Officers, and CISOs

Any entity dealing with Web3, crypto, and digital assets

Not only DeFi

Value Proposition

- Comply with current and future regulations
- Complete visibility and control of your Web3 environment, 24-7, automated, and with full context
- Safeguard client assets

Automated & Ongoing Web3 Risk Management

Complete Asset Visibility and Control

Thank You.

Cyber@BGU,
Software and Information Systems Engineering
Ben-Gurion University of the Negev
Contact: Rami Puzis puzis@bgu.ac.il

Cyber-Biological Threats

Vision: Developing secure coding practices in biohacking and synthetic biology

Mission: BioSOC – security operations center for efficient and robust synthetic DNA order screening

Digital security and biosecurity have similar challenges

- **Code Obfuscation**

- **DNA Obfuscation**

Changing the DNA such that it is difficult to recognize but it can be reconstructed into a functional form.

Biologically proven!

Supply chain attacks!

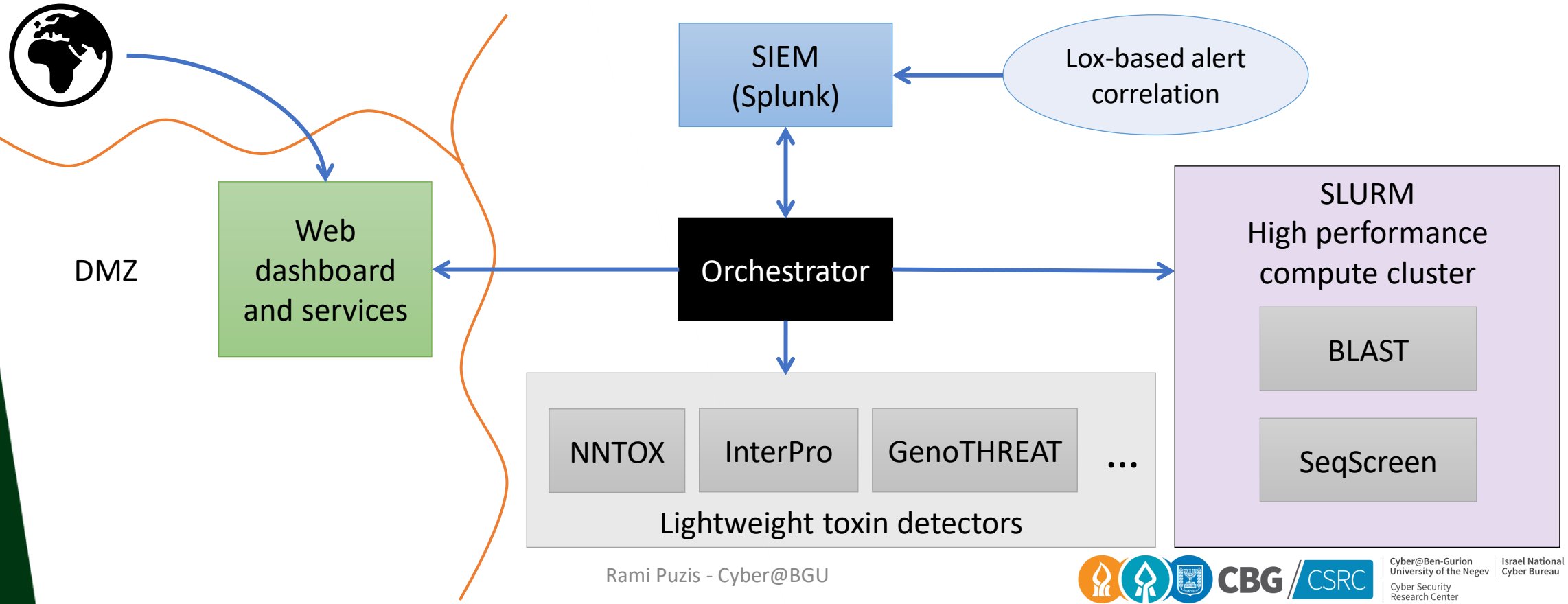
Alert investigation and triaging

False-positive rates

False-negatives		
	Detected	**Not detected**
Toxins	4794	1699
Half-toxin	4479	2014
Toxins+camouflage	502	5991
Cre-Lox obfuscated toxins	129	6364

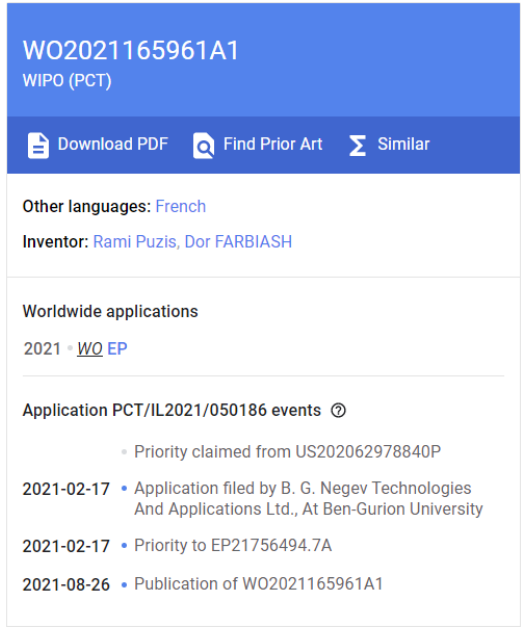
Four decades of digital cyber-security empower the biological cyber-security

- Security Operation Center (SOC) with bioinformatics toolbox
- Detection and triaging of suspicious orders



Gene Edit Distance

Quantify how easy is it to
make something malicious
out of an order rather than checking
if it is already malicious.

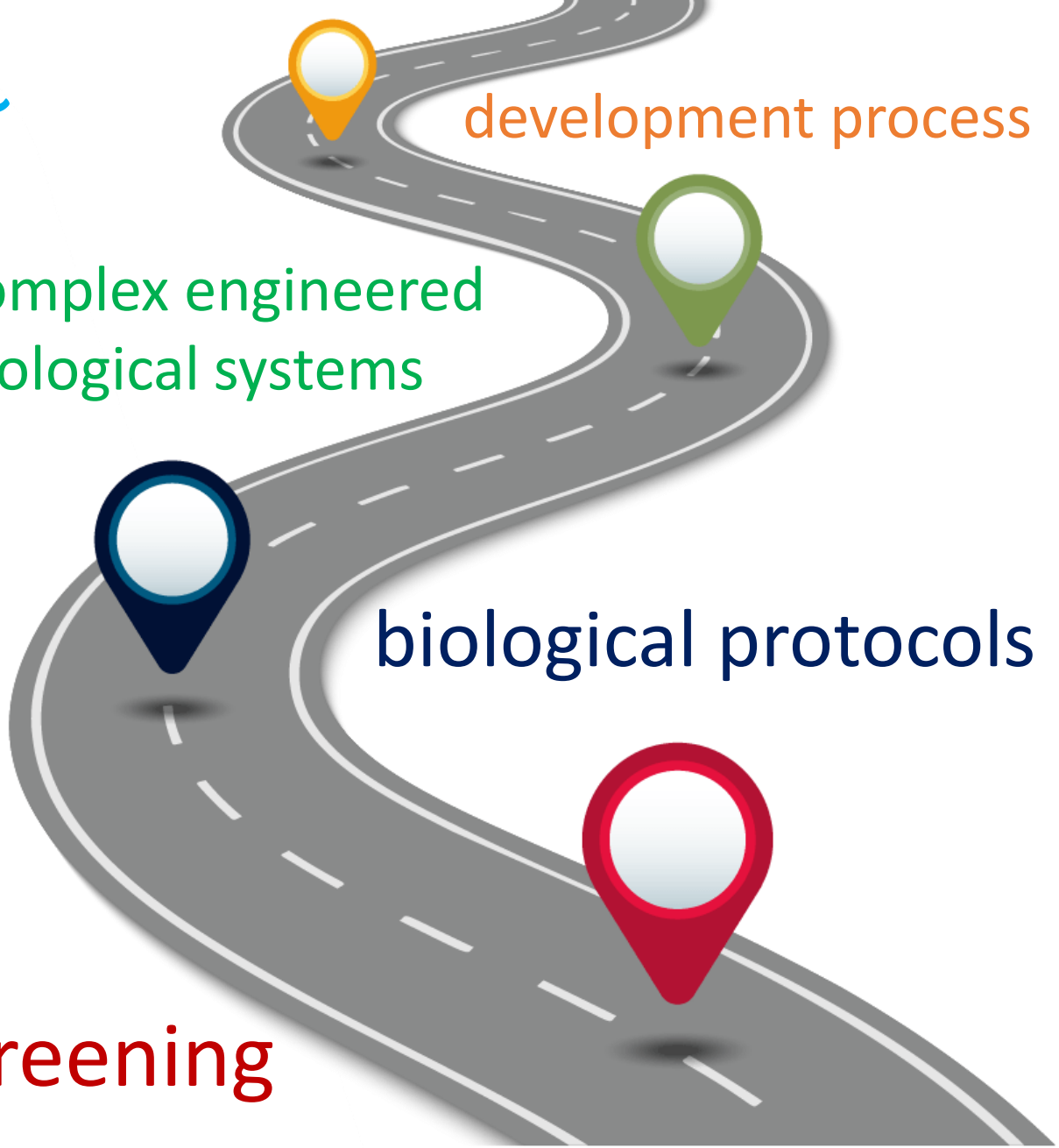


The market

- Competition:
  - DNA screening service providers:
    - BATTELLE
    - ACLID
  - Cyber-biosecurity anti-crime
    - bronic
- Academia:
  - NNTox – toxicity prediction tool; relies on GO terms; easily circumvented
  - SeqScreen – framework for sequence function prediction; easily circumvented
  - InterProScan – … easily circumvented
- Potential clients
  - Members of the IGSC consortium (span 80% of synthetic DNA market)

R&D roadmap

Adversary resilient



SUPER POLYMER

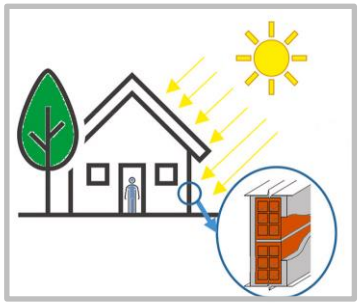
Fully Crystalline Polymers

Reinventing Polymers

Our first market – Net Zero Construction

Significant problems we solve:

Buildings use about 50% of their energy for heating and cooling [1]



Buildings can be cooled and heated with less energy and carbon emissions

More than 25% of all buildings in the US are water-damaged [2]

We can reduce dampness and water damage in buildings

1. Source: European Commission
2. Source: Surviving mold

SUPER POLYMER
Fully Crystalline Polymers

Transforming semi-crystalline polymers into fully crystalline Super Polymers

Input

Raw Material
40-50% crystalline

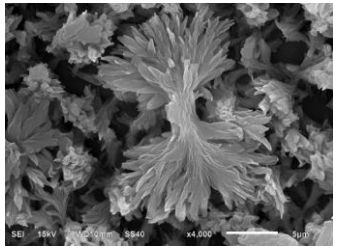
We take commercial grade raw polymers

Process

Apply our proprietary process

Organize the raw polymer differently

Output



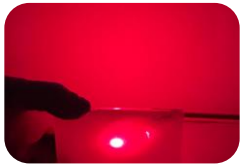
100% crystalline polymer

Provided as powder, emulsion or additive

Based on technology transfer • 3 national phase applications • 1 patent accepted [CN]

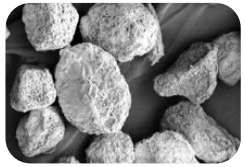
SUPER POLYMER
Fully Crystalline Polymers

Unique morphology leads to new and improved properties



High refractive index

Heat shielding



Air encapsulation

Hydrophobicity

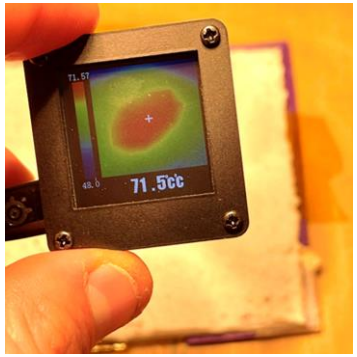
Irradiated heat scattering

Very low heat conductivity

Excellent waterproof

Cost effective

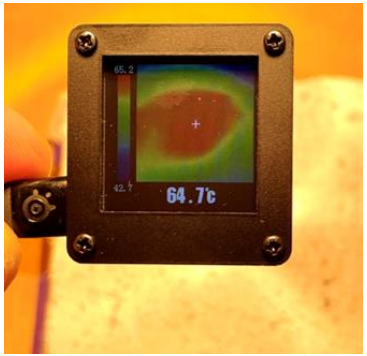
Environmental impact



Plaster sample with no SP

> 10 °C

less heat transfer



Plaster sample with 5% SP

- Lower dry weight by 20%
- 50% less water absorption
- Improved whiteness

A word on using recycled HDPE

Crystallinity of rHDPE improves by 36% in first trials

rHDPE granules [black] and SP layer made from it [white]

Crystalline morphology is indicated by a hydrophobic surface

Traction from first tech demonstrations

8 POs for Sample testing

3 Joint projects are negotiated with strategic partners

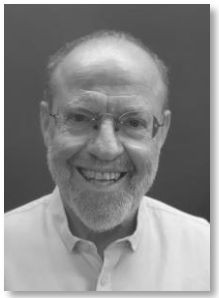
14 Potential strategic partners are in dialogues

- Construction materials
- Chemicals
- Paint & Coating
- Packaging
- Consumer electronics
- Raw materials

Super mission - super team

Hagai Ortner CEO and Co-founder

27 years of management and startup development



Atzmon Amitai Director and Co-founder

35 years as a product growth expert in the chemicals industry

Dr. Evgeniy Mervinetsky Head of R&D

Ph.D. in Chemistry
15 years of research and development experience

Adva Bar-On, Paint development consultant

Chemist, with 35 years of experience in paint & coating formulations

Ask

Pre-seed SAFE round

- Closing a preseed SAFE round of $1M with $300K open to investors
- SAFE terms:
  $5m Cap, 15% discount

Use of proceeds

A year from now:

1. Kg's/month process to supply samples
2. Product validation with strategic partner

SUPER POLYMER

Fully Crystalline Polymers

Thank you!

hagai@super-polymer.com

+972-58-7263197

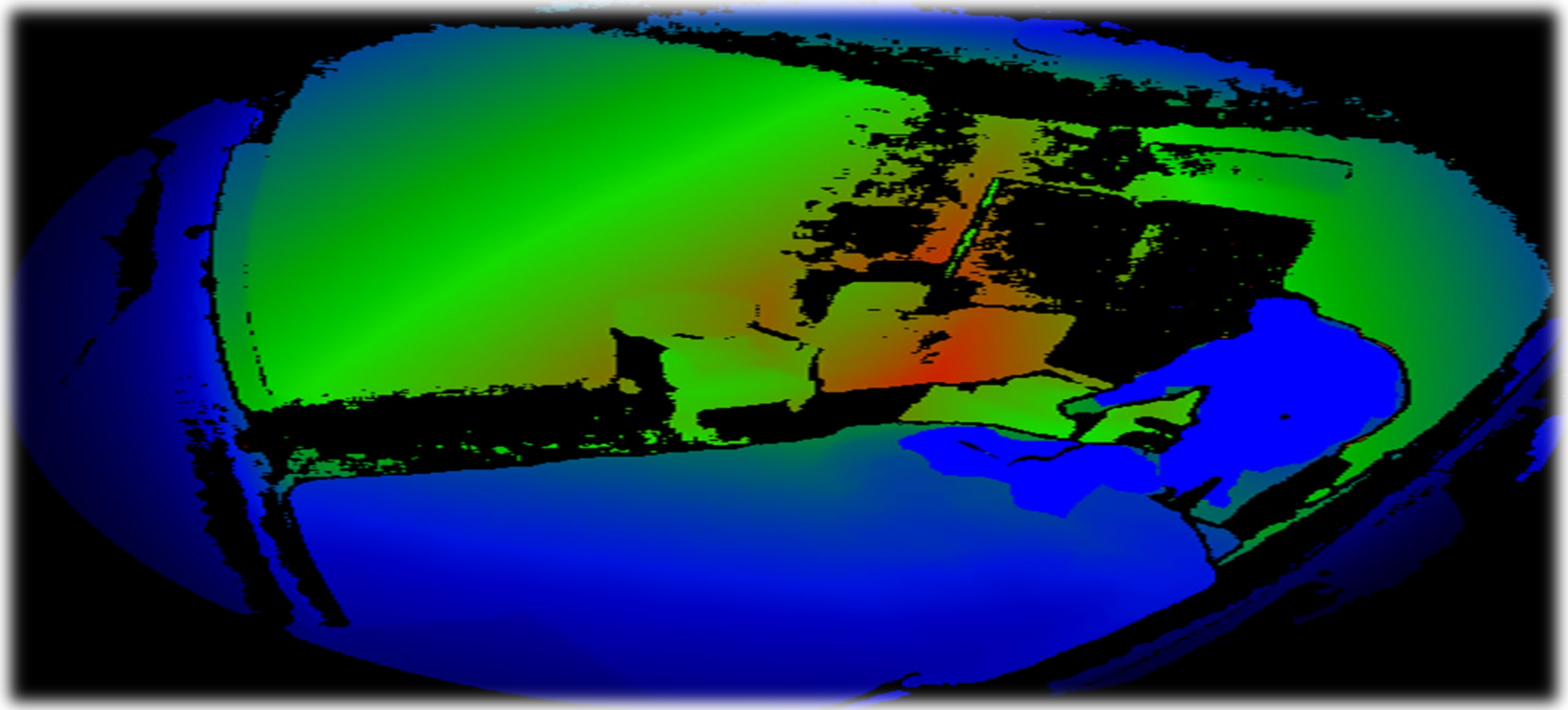
Our site

www.super-polymer.com

Our LinkedIn

AntisepTech

Denied Camera Area Sensing - HealthCare Settings



Barak Katz (PhD) Boaz Tadmor (MD)

barakkatzz@gmail.com

+972-528744474

Summary

Malpractice and improper care are the main causes for infections, abuse, bedsores, and falls, few of the most expensive Never Events in healthcare settings. Current technological solutions fail in automatic detection, understanding and reporting meaningful events and behaviors in private areas, such as patients' rooms, that may prevent and control such Never Events in the healthcare settings.

AntisepTech 3D patented AI solution which comply with HIPPA requirements, addresses several healthcare use cases such as abuse, infections (HAI) control and prevention, bedsores control and prevention, falls control and prevention, restlessness situation etc. Use cases that are relevant both to homecare, hospitals, nursing home, rehab centers and a like

AntisepTech solution enables real time reliable automatic detection of behavioral patterns that may indicate improper care or potential malpractice. Moreover, the ability to automatically and reliably understand such events allows us not only to collect meaningful data and report it, but also allows us to provide real time meaningful alerts, and the ability to collect and stream meaningful anonymous visual data that assist in understanding remotely activity inside private settings that facilitates interventions that may prevent Never Events and malpractice. Such intervention modify behavior and provide patients, stuff and families better hospitalization experience and peace of mind.

The proposed solution is the first and only solution that comply with the Israeli law with the aim of "locating and preventing harm to inpatients in a geriatric hospital through the installation of cameras inside patient room while preserving, as much as possible, the dignity and privacy of the inpatients and the employees in the hospital."

AntisepTech solution is currently installed in clinical settings at a long-term autistic nursing home and protects low functioning autistic adult from abuse and improper care inside his private room. The solution automatically detects proximity events below 50cm, records the event of proximity, records access to patient bed at night, records staff entrance at night with accordance to institutional requirements, while maintaining the dignity and privacy of patient and staff while complying with the Israeli law.

In addition, AntisepTech gain significant traction for its HAI use case from healthcare leaders. Recently AntisepTech, jointly with the Hospitals Division of the Israeli Ministry of Health, Dorot Geriatric & Rehab Center and the Israeli Innovation Authority, won a grant that aims to expend the clinical trial currently conducted in the autistic institution to three different beds in the Dorot Geriatric Center addressing abuse/bedsores/falls control and prevention uses cases.

Monitor Interact Educate Prevent

The Unmet Need

Lack of Monitoring & Prevention at "Camera-Denied" Areas in the HealthCare Settings

Quality-of-Care

HAI

Abuse

What if an interactive AI robotic "observer" is placed **inside** patients' room?

Would it assist with better understanding or predicting abuse? Prevent HAI?
Would safety events, negligence or medical errors may be observed or predicted?
Could we provide a better Quality-of-Care? Provide peace-of-mind to the patient/family/stuff/management?
Could we reduce the massive costs associated with such undesired "never events"?

Monitor Interact Educate Prevent

The Problem

Monitor Interact Educate Prevent

Solution

11110191 PATENTED

10754924 PATENTED

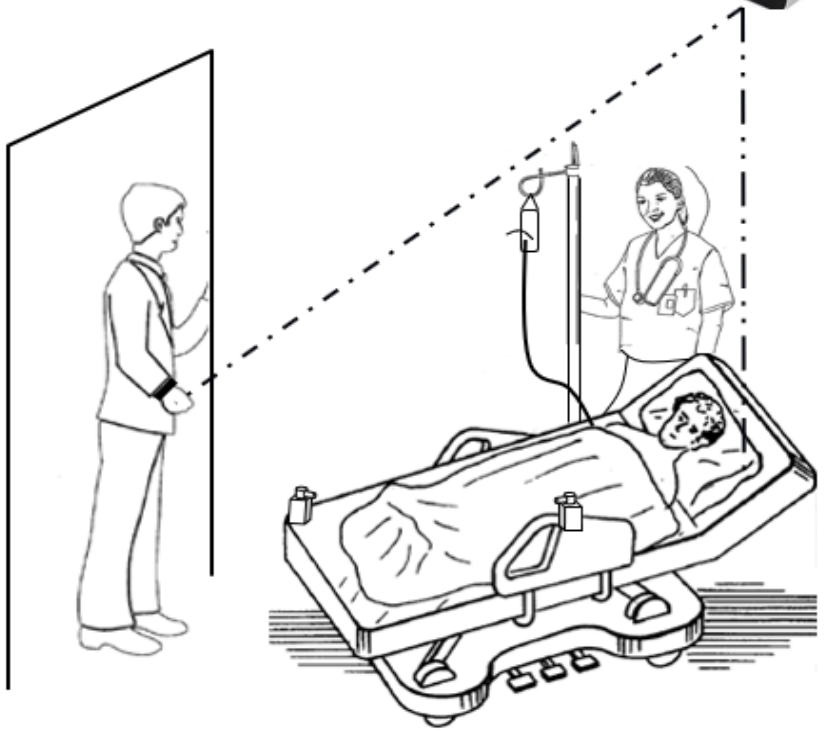
63/452,834 PROVISIONAL

Depth only
Ethical/Privacy
Protected
Information

Hygiene

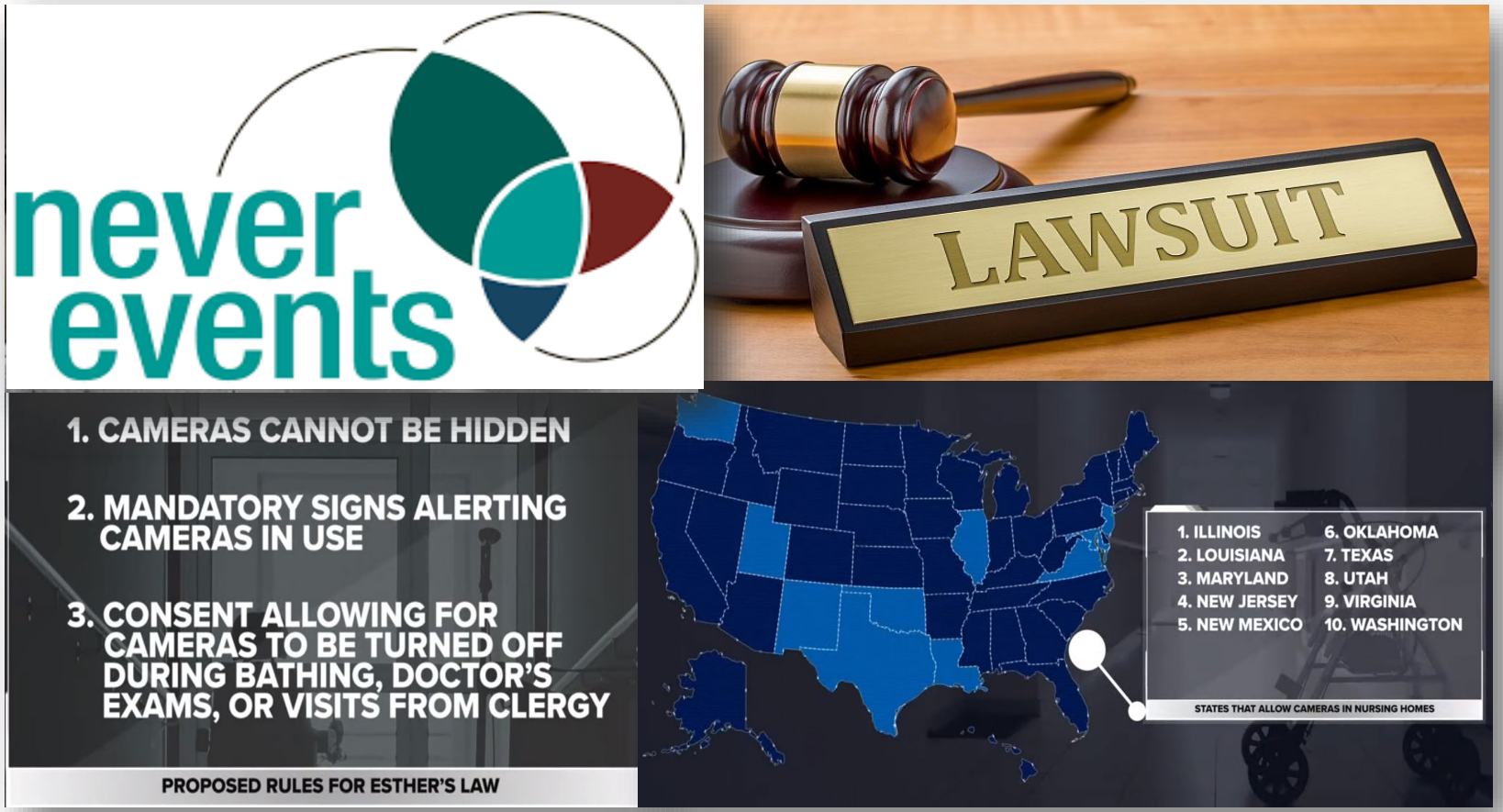
Abuse

Hand Hygiene | Hand contact Detection | Surface Hygiene | HCWs Productivity



Monitor Interact Educate Prevent

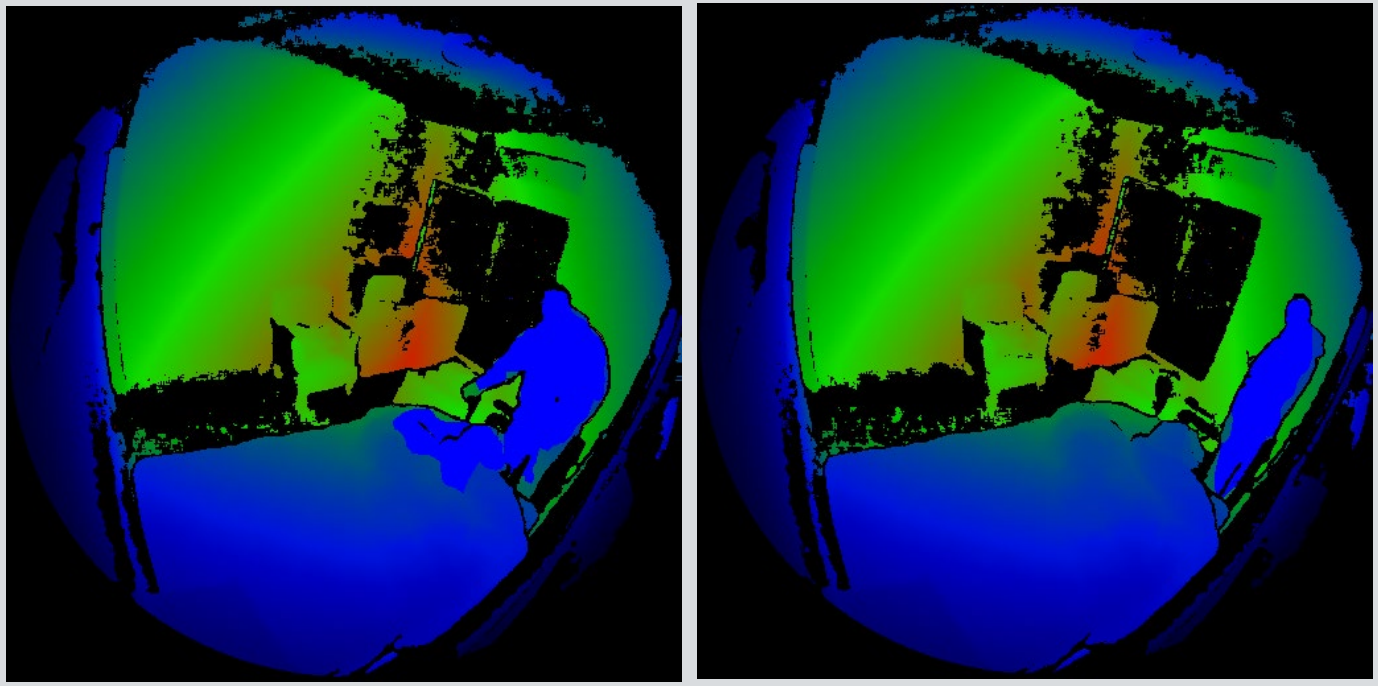
The Market – US



Monitor Interact Educate Prevent

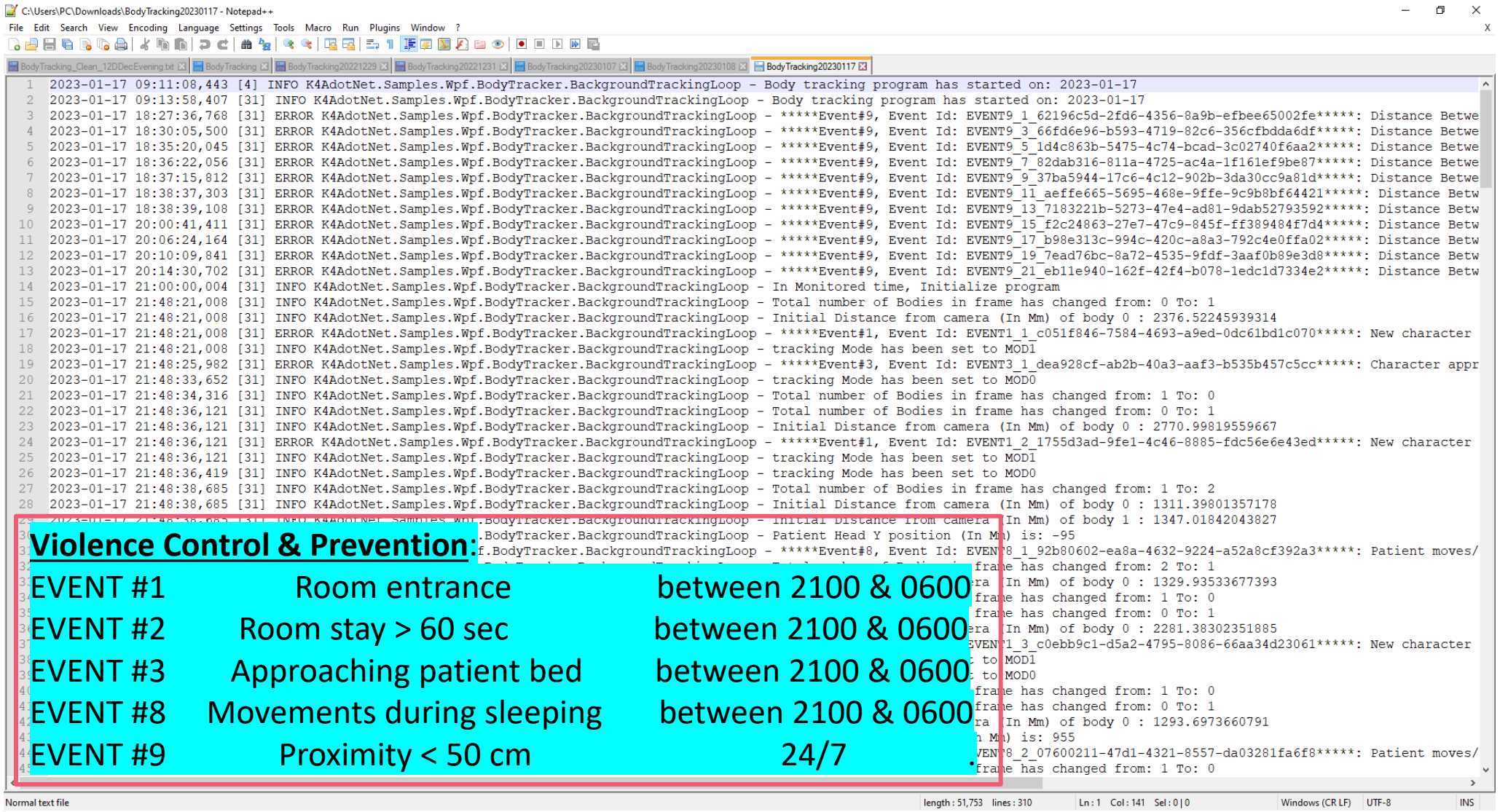
Current Status - Recording

EVENT #3



Monitor Interact Educate Prevent

Current Status - Analytics



Violence Control & Prevention:

EVENT #1	Room entrance	between 2100 & 0600
EVENT #2	Room stay > 60 sec	between 2100 & 0600
EVENT #3	Approaching patient bed	between 2100 & 0600
EVENT #8	Movements during sleeping	between 2100 & 0600
EVENT #9	Proximity < 50 cm	24/7

Monitor Interact Educate Prevent

Traction - AntisepTech

Clinical pilot – Kfar Ofarim, ALUT Israel (Geriatric/Autistic)

On going discussions – Israel Minister of Health Hospitals Division

Centers for Disease Control and Prevention

"We would like to invite you to continue in the collaboration if you are interested. This would include participating in teleconferences discussing progress, barriers, problem solving etc. We hope you might be interested in continuing the collaboration in this way-we think it might be mutually beneficial".

Prof John Jernigan
Deputy director
Division of Healthcare Quality Promotion
CDC's National Center for Infectious Disease

Columbia University

"Your system is innovative and unique and if the beta testing is promising, it may be very welcome and useful in a variety of healthcare settings. You might also consider alternative health care sites such as nursing homes, ambulatory care clinics, dialysis units, cardiac catheterization units, surgical suites, etc. Please accept my best wishes as you embark on this important venture."

Prof. Elaine Larson
Associate Dean for Nursing Research
School of Nursing, Columbia University
Leader in the field of Hand Hygiene

Monitor Interact Educate Prevent


1. Abuse
2. Bedsores
3. Falls
4. Infections
1
Geriatric Rehabilitation
2
ICU
MINISTRY OF HEALTH
מדינת ישראל, משרד הבריאות
מר"ג - מרכז רפואי גריאטרי נתניה
STATE OF ISRAEL MINISTRY OF HEALTH
DOROT - NETANYA GERIATRIC MEDICAL CENTER
דורות
מרכז רפואי
לשיקום וגריאטריה

AMaze

Future Traffic Management
Beyond Autonomic Vehicles

Hannah Yair, Shlomi Dolev, Ehud Gudes

CSCML 2023

The Problem

- Time
- Money
- Pollution
- Accidents

The Opportunities

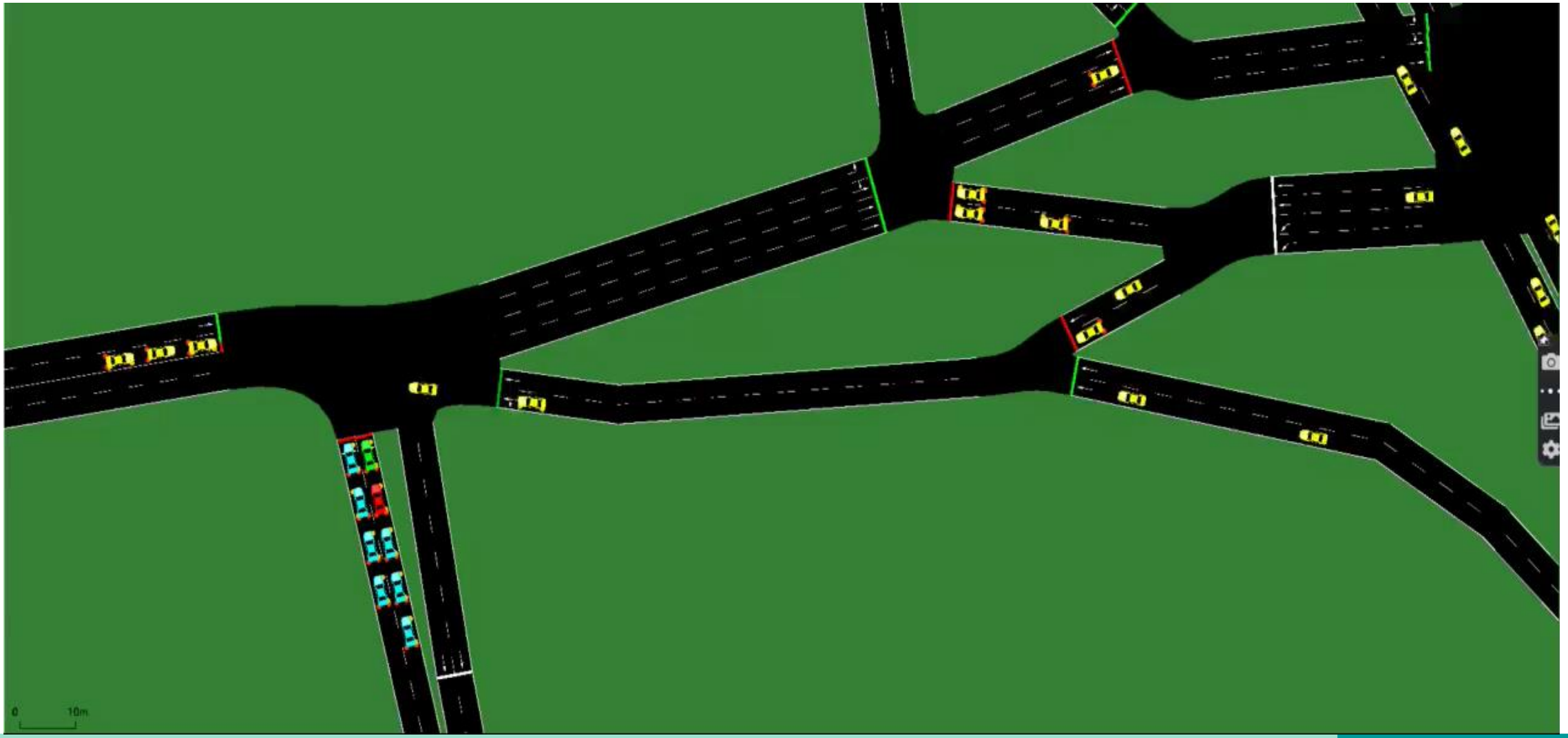
The Technology

- A new algorithm that is based on graph methods for identifying real-time platoons
- A new algorithm that synchronizes vehicles into junctions by scheduling a timing, based on dynamic programming and scheduling the vehicles accurately
- Solved the scheduling timing for the extended problem (not only for one junction)
- Cyber security solution for the sever identification problem in V2V communication
- One is approved one is ending patents

- *The algorithms are implemented by the Sumo simulator [1] and can demonstrate any kind of map on Google maps*

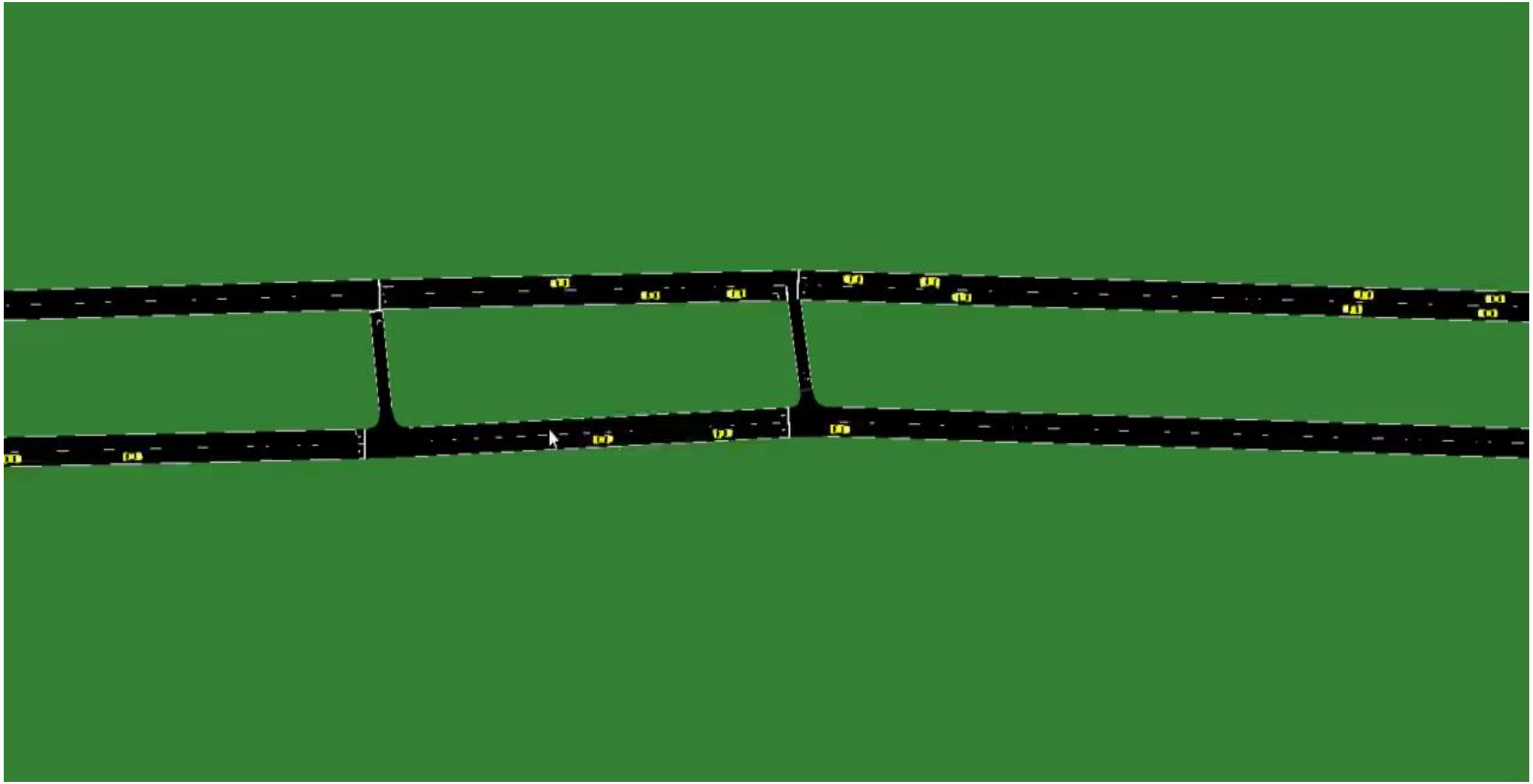
Train Replacement

The Solution



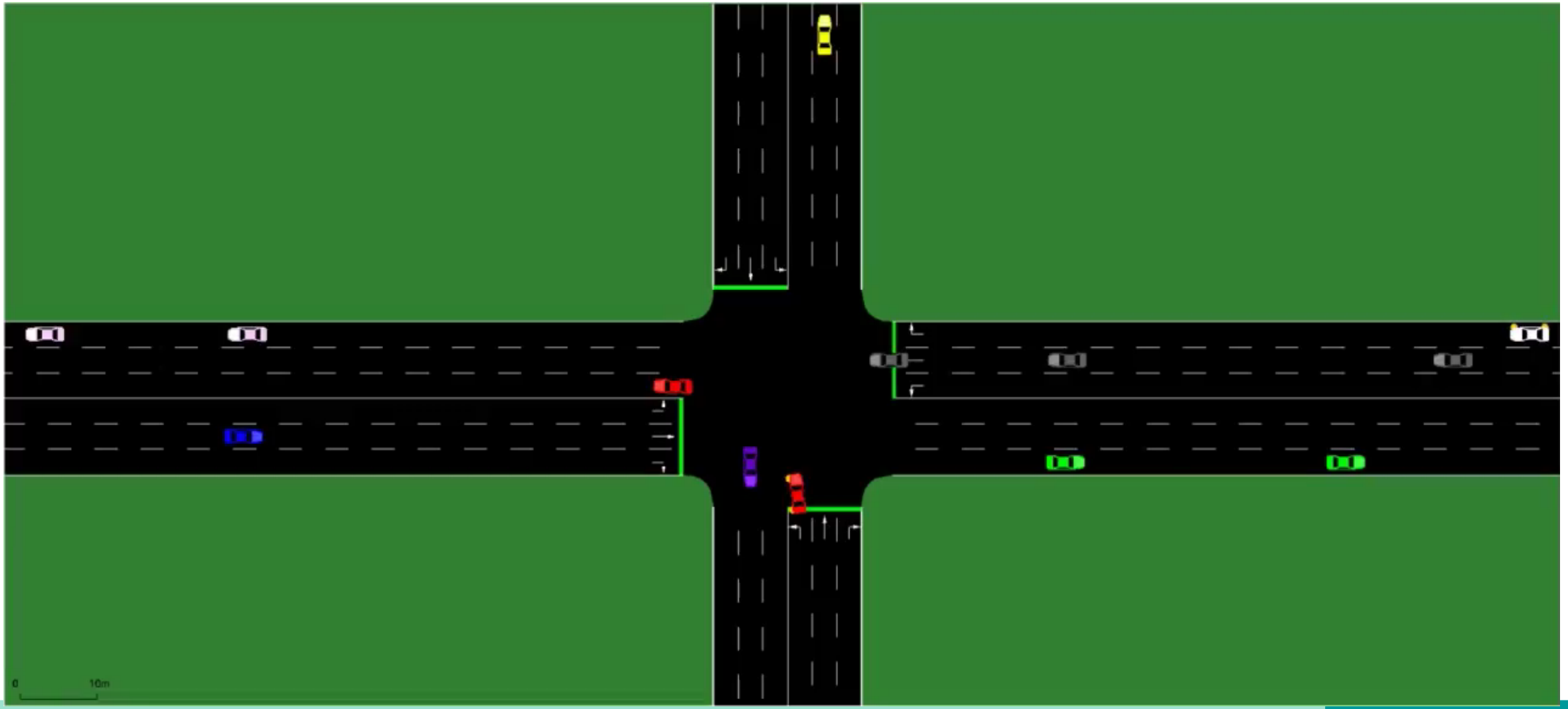
Traffic Control

The Solution



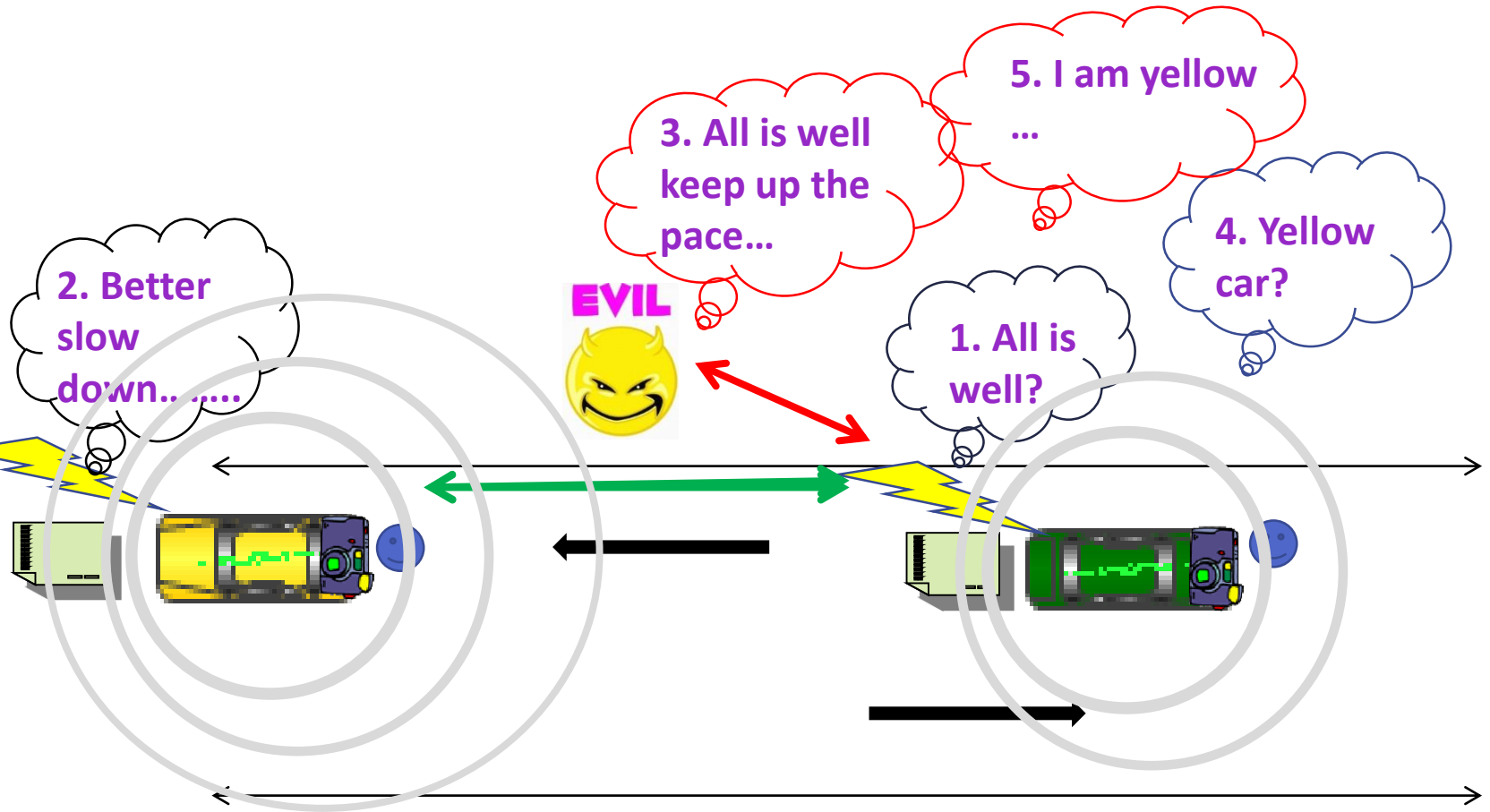
Junction Management

The Solution



Extended Problem

Insecure Communication



Proposed Approach

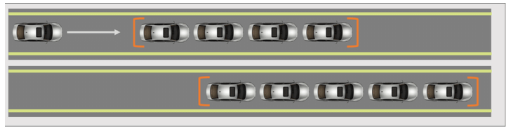
- Certificate Authority (CA)
  - Certificate pre-processing
  - Certified signed attributes
  - Certified signed public key

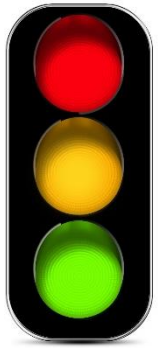
- Out-of-band channel of communication
  - Vehicle authentication with **verifiable attributes**
  - Certified **coupling** in **public key** and **attributes**

Go to Market

- The Product:

  Traffic Management Algorithms





- The Target Audience:

  Transportation authorities and urban planners,

  Traffic engineers and designers,

  Policy-makers and government officials.

  - The result can be used in airplane scope and robotics (drones).

Go to Market

- The Value:

  Efficient resource utilization,

  Improved traffic flow,

  Enhanced safety,

  Reduced travel time,

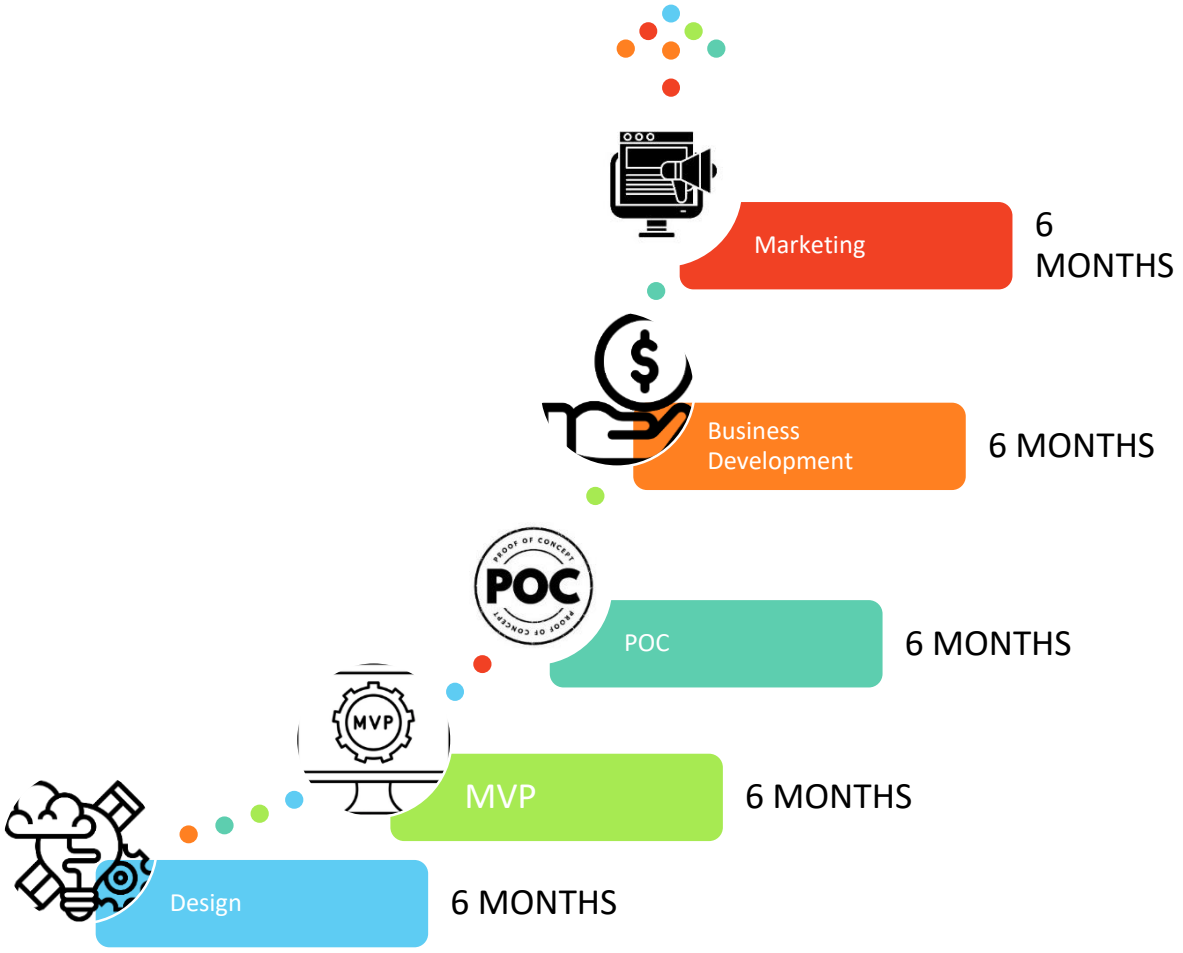
  Environmental benefits.

- We try to collaborate with IAI and ISTRC.
  - The research was awarded an honors scholarship by ISTRC.

THE ISRAELI SMART TRANSPORTATION RESEARCH CENTER

IAI

Timeline


Marketing
6 MONTHS
Business Development
6 MONTHS
POC
6 MONTHS
MVP
6 MONTHS
Design
6 MONTHS

The Market's Competitors

NOTRAFFIC

- Focuses on a traffic management platform that utilizes connected vehicles' data and intelligent edge sensors to optimize traffic flows and reduce accidents in real-time.
- Transforms traffic signals into smart infrastructure capable of understanding the complete traffic picture and responding to road users. (by AI modles)

AMaze

- Aims on synchronize vehicles at junctions by scheduling timing using dynamic programming, enabling accurate vehicle scheduling to optimize traffic flows and reduce accidents in real-time.
- Analysies the data from the GPS's and gives instructions to the vehicle at junctions. (By graphs and DP algorithms)

B2B

נתיבי איילון
תחבורה מתקדמת לישראל

U.S. Department of Transportation

נתיבי ישראל
החברה הלאומית לתשתיות תחבורה בע"מ

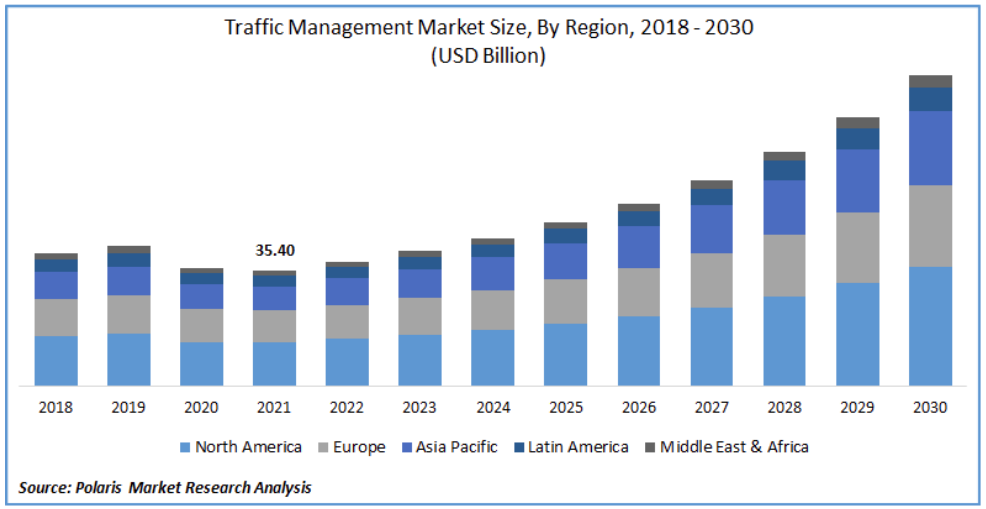
רשות החדשנות
Israel Innovation Authority

eit Urban Mobility MASTER SCHOOL

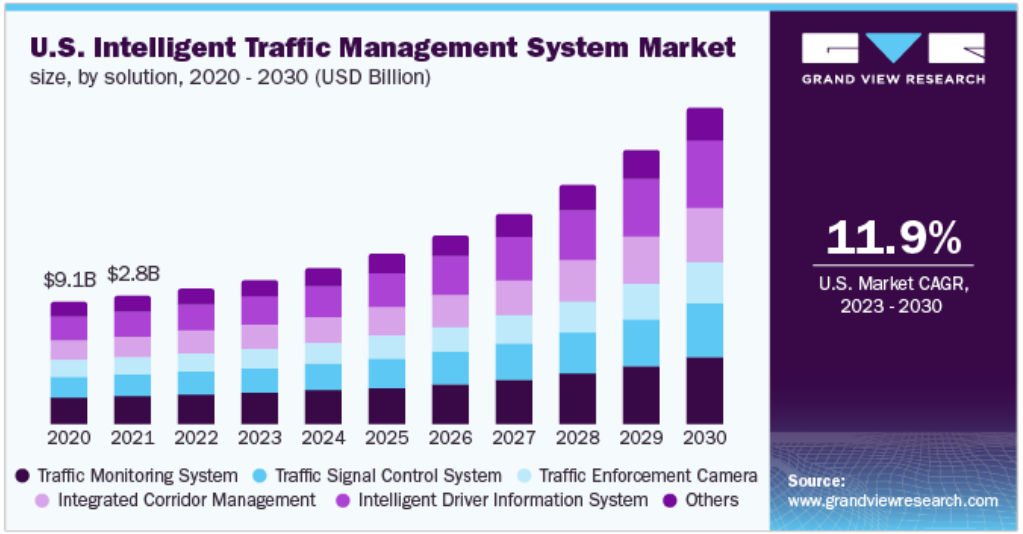
AMaze

The Market

[2]



[3]

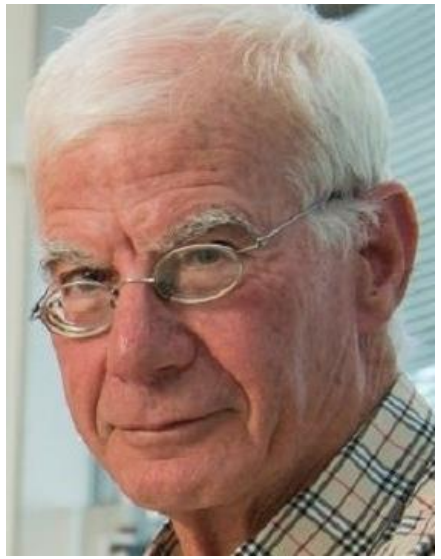


The Team

Ms. Hannah Yair

PhD Candidate

Prof. Shlomi Dolev



Prof. Ehud Gudes

AMaze

Thank you!

References

[1] Lopez, P.A., Behrisch, M., Bieker-Walz, L., Erdmann, J., Fl¨otter¨od, Y.P., Hilbrich, R., L¨ucken, L., Rummel, J., Wagner, P., Wießner, E.: Microscopic traffic simulation using sumo. In: The 21st IEEE International Conference on Intelligent Transporta-tion Systems. IEEE (2018)

[2] https://www.polarismarketresearch.com/industry-analysis/traffic-management-market

[3] https://www.grandviewresearch.com/industry-analysis/intelligent-traffic-management-system-market