VDiSC: An Open Source Framework for Distributed Smart City Vision and Biometric Surveillance Networks

Joel Brogan
broganjr@ornl.gov

Nell Barber
barbercl@@ornl.gov

David Cornett
cornettdciii@ornl.gov

David Bolme
bolmeds@ornl.gov

Abstract

Recent global growth in the interest of smart cities has led to trillions of dollars of investment toward research and development. These connected cities have the potential to create a symbiosis of technology and society and revolutionize the cost of living, safety, ecological sustainability, and quality of life of societies on a world-wide scale. Some key components of the smart city construct are connected smart grids, self-driving cars, federated learning systems, smart utilities, large-scale public transit, and proactive surveillance systems. While exciting in prospect, these technologies and their subsequent integration cannot be attempted without addressing the potential societal impacts of such a high degree of automation and data sharing. Additionally, the feasibility of coordinating so many disparate tasks will require a fast, extensible, unifying framework. To that end, we propose the Distributed Smart City framework for Vision, or VDiSC. VDiSC serves as a unified biometric API harness that allows for seamless evaluation, deployment, and simple pipeline creation for heterogeneous biometric software. VDiSC additionally provides a fully declarative capability for defining and coordinating custom machine learning and sensor pipelines, allowing the distribution of processes across otherwise incompatible hardware and networks. VDiSC ultimately provides a way to quickly configure, hot-swap, and expand large coordinated or federated systems online without interruptions for maintenance. Because much of the data collected in a smart city contains Personally Identifying Information (PII), VDiSC also provides built-in tools and layers to ensure secure and encrypted streaming, storage, and access of PII data across distributed systems.

1. Introduction

It is estimated that by 2023, research and development toward smart city applications will reach a market share of over $700 billion dollars [14], with hundreds of billions more going towards Internet of Things (IoT) research. While innovation in this area pushes forward at unprecedented speed, less attention is being directed toward the safe and secure capture and transmission of biometric information the likes of which is vital to an effective smart city implementation. Designed as a successor to the groundwork laid by Face Recognition Oak Ridge (FaRO) in 2019 [5], VDiSC is a highly scalable inferencing framework for streaming, processing, and visualization of biometric data across distributed systems. Robust smart city and insight-driven surveillance use cases rely on heterogeneous data sources and software that may be incompatible or prove unwieldy when used in tandem. VDiSC addresses this problem by providing a unified API harness to accelerate the creation and deployment of custom biometric pipelines, the acquisition of high-quality data, and the capture of real-time insights from streaming video. VDiSC is also redesigned to include privacy protection and network security components that are critical to its intended applications. Though VDiSC is primarily configured for tasks related to image-based biometric detection and identification using modalities like face, whole body, and gait, it is flexible enough that data sources capturing different modalities for varied purposes can be integrated with relative ease.

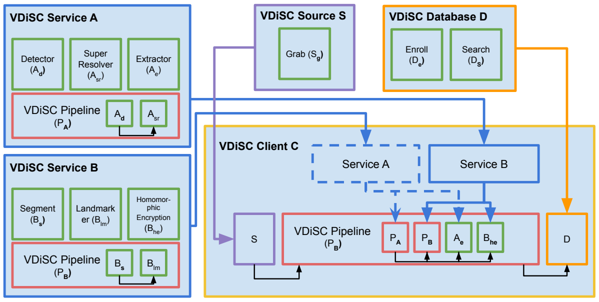

Figure 1. An abstract example of how a declarative VDiSC computer vision pipelined infrastructure works.

This paper is organized as follows: In Section 2, we discuss privacy and potential means by which to protect it with regard to face imagery. In Section 3, we give an overview of related work. In Section 4, we provide a technical overview of the VDiSC architecture. In Section 5, we provide information about accessing VDiSC. In Section 6, we give a conclusion and discuss future work.

2. Privacy

Privacy is a key component of smart city research [15]; however, at this point in time, no studies, surveys, or software have been published pertaining to this specific topic [14].

While privacy remains an abstract concept in an era characterized by the confluence of social media, omnipresent digital connectivity, and inescapable data capture, failure to adequately protect biometric information can yield very tangible consequences. Advances in techniques used to reconstruct faces from face feature vectors have necessitated additional safeguards for not just raw face images but also the unique descriptors extracted from them. Reconstructed face images can be submitted to cheap consumer services that can identify other images with matching subjects, many of which contain identifying metadata or are otherwise linked to identifying information online [24]. This undercuts assurances from many face recognition vendors over the years that, once extracted, these feature vectors do not require encryption because they are already unusable in the wrong hands.

A popular solution to face feature vector vulnerability is homomorphic encryption (HE), and the application of fully homomorphic encryption (FHE) to biometric matching has been a popular area of research in recent years. HE enables the encryption of face feature vectors such that matching is done in homomorphic space, and plain face feature vectors need never be exposed to networks or to server environments. In spite of its advantages, however, FHE does not support real-time biometric identification and, as it stands, is not a fit for VDiSC. While authors in [18] show that simple matching can be performed in real time using FHE, there is little utility in these results when considering the need to encrypt all new faces captured by a particular sensor and search a gallery of arbitrary length for matching individuals. Instead, VDiSC leverages partially homomorphic encryption (PHE) to serve in its place. The principal distinction between FHE and the PHE utilized by VDiSC is that the former supports the evaluation of arbitrary functions by way of its support for both addition and multiplication in homomorphic space [2] while the latter does not. The PHE implementation used by VDiSC is homomorphic over addition but supports only multiplication between encrypted and unencrypted operands to yield encrypted results. This means that cosine similarity is not supported in homomorphic space and matching must be performed by other means. VDiSC's PHE implementation is described in greater technical detail in Section 4.7.

3. Related Work

The FaRO [3, 5] framework, which inspired the foundation of VDiSC, arose from the need to create a convenient pipeline for biometric evaluations that would mitigate the often cumbersome integration tasks found in open-source and academic algorithms. The goal was an efficient framework in which components of a given pipeline (e.g., sensor source, template extractor, detector, matcher) could be swapped in and out with ease to provide the optimal system for a particular use case. The initial development team was also concerned with scalability in the shift from cloud-based servers to edge-deployed systems. To provide this optimized computational architecture, gRPC was tightly implemented to manage the streaming interfaces between server systems and client systems.

The VDiSK framework and its FaRO predecessor have been utilized in the development of a specialized biometrics system for the identification of drivers and passengers in moving vehicles [7, 21]. Given FaRO's successful performance in this implementation, it was further extended to incorporate more benchmarking and state-of-the-art algorithms to benefit the biometrics community and was released openly [3, 5].

In parallel to the release of VDiSC's predecessor FaRO, which was released as a gRPC client–server biometrics evaluation framework, other similar frameworks were also being developed and released. The Nvidia Triton Inference Server [1], which began development in November 2018, provided similar gRPC client–server functionality, but was developed with TensorRT specifically in mind. Additionally, VDiSC now provides complete declarative infrastructure for chaining microservices together into workflows—a feature Triton does not have.

Apart from Triton, other software suites aim to provide similar unified API function calls to hosted machine learning models. Software such as Amazon's Sagemaker [13] provides cloud-specific hosting tools callable by a unified API, while libraries such as OpenVINO [10] make it possible to load models on many different hardware architectures. VDiSC provides generic worker interfaces that harness APIs such as Sagemaker and OpenVINO, which allows for greater diversity of hostable models. In the next section, we will discuss the unique and novel aspects of VDiSC's architecture and implementation, and what sets it apart from other software in the model workflow space.

4. VDiSC Architecture Overview

VDiSC was designed specifically as a unifying API harness for biometric algorithms, usually for the purposes of evaluations and experiments. The underpinning design decisions for VDiSC revolve around the ability to quickly develop and deploy real-time, secure, and safe computer vi-

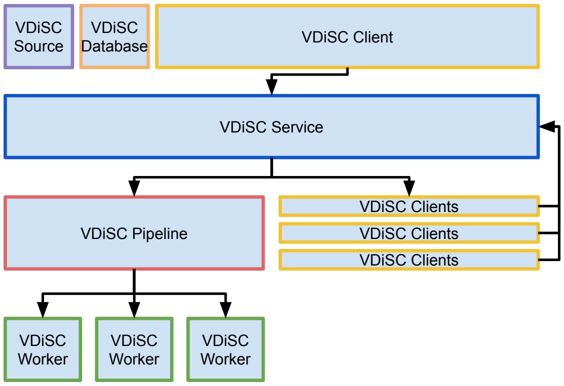

Figure 2. The hierarchy of VDiSC entities. Generic gRPC messages called VDiSCRecord and VDiSCReply, which contain outputs from workers and pipelines, can be passed among all entities at any level. VDiSC Clients connect to VDiSC Services, which hold persistent instantiations of VDiSC Pipelines. VDiSC Pipelines contain chained-together graphs of VDiSC Workers, which act as microservices. Each VDiSC Service also contains persistent connections to other VDiSC Clients in its area network, which in turn connect to their own services. This provides the basis for the recursive VDiSC chaining that makes it so powerful.

sion and biometric workflows in unconstrained settings, using a declarative paradigm that allows the user to easily connect a myriad of vision algorithms and microservices.

VDiSC 's design revolves around a gRPC server–client architecture that utilizes message passing and procedure calls to stream real-time video from client to server and asynchronously receive return results from server to client. VDiSC is built on three main hierarchical concepts: workers, pipelines, and services. The hierarchy itself can be seen in Figure 2. All of these entities perform machine learning tasks on inputs by accepting a generic gRPC message, called a VDiSCRecord, and returning a generic gRPC message, called a VDiSCReply. These record and reply messages act as a unifying language within VDiSC and can be passed among between clients, servers, workers, and pipelines interchangeably. In this way, workers, pipelines, and services can pass messages either locally within themselves or between services and pipelines being hosted elsewhere. As can be seen in Figure 2, this implicitly creates a recursive hierarchical networking structure that allows workers hosted within remote services to propagate through network channel chains, similar to a mesh network. VDiSC client calls can be made to connect these workers and pipelines in a variety of ways using a declarative interface. These powerful concepts provide the foundation for VDiSCs remarkable online configurability when being used to deploy and fine-tune distributed and heterogeneous machine learning systems in smart city infrastructures.

4.1. VDiSC Workers

Each individual worker performs a single task, known as a microservice, such as detection, segmenting, or feature extraction. The worker contains the initialization constructors to load the particular libraries, models, and resources required to perform that task. Each worker also implements a set of initialization options. The VDiSCWorker also implements a method to report information about itself, including what type of microservice it provides, and what resources it has available to it. Each worker takes VDiSCRecords as input and returns VDiSCReplies as output. On an abstract level, all workers are able to interface with each other through records and replies but will throw exceptions if a given VDiSCReply does not contain the required input for a subsequent worker.

4.2. VDiSC Pipelines

Each VDiSC Pipeline consists of a directed acyclic graph workflow of DiSC Workers, chained together via their inputs and outputs. The VDiSC Pipeline leverages the Multiprocessing library to allow for asynchronous calls to various workers in either an unordered manner or first-in-first-out order queues. In this way, local resources can be utilized to their fullest extent to perform parallel jobs that do not require order, or that require only minimal ordered dependencies between workers within the workflow.

The VDiSC Pipeline itself subclasses the VDiSC Worker and can utilize nested pipelines within its own workflow. This nested pipeline functionality helps provide further flexibility when creating more complex dependency graphs in distributed wide area networks. An example of this nested pipeline infrastructure is shown in Figure 1.

4.3. VDiSC Services

A VDiSC Service implements the entire gRPC server required to make remote calls across a network channel. This server contains four main elements:

- The code and infrastructure that allows online declaring of VDiSC Pipelines constituted of chained-together VDiSC Workers.
- A set of local VDiSC Workers that are loadable and run-able within the VDiSC Service's local environment.
- A set of VDiSC Clients that all connect with other visible VDiSC Services available on the network.
- One or more VDiSC Databases that are endpoints to collect output from various VDiSC Workers, in scenarios when databases must be created for enrollments or searches of various entities.

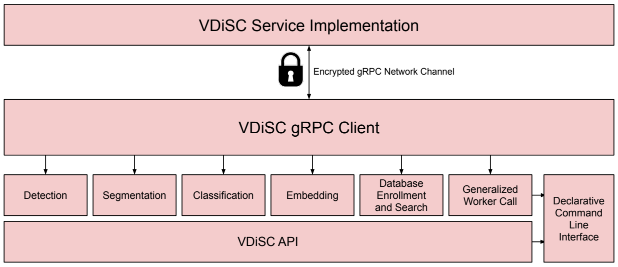
VDiSC Service Implementation
Encrypted gRPC Network Channel
VDiSC gRPC Client
Detection
Segmentation
Classification
Embedding
Database Enrollment and Search
Generalized Worker Call
Declarative Command Line Interface
VDiSC API

Figure 3. A visualization showing the specific and generic computer vision calls implemented in the VDiSC Client that allow for communication to VDiSC Services, and their nested workers and pipelines. The client provides a unified API and CLI to access workers and pipelines within a service. If a computer vision–type call that does not fall within the outlined client framework is required, such as a special type of super-resolution or other processing, it can be implemented ad-hoc using the "generalized worker call."

These main elements allow users to connect to a VDiSC Service via a VDiSC Client, then create pipelines of VDiSC Workers (either local or remote) via a declarative interface. These chains can be created from either fully local workers or a mixture of local and remote workers.

4.4. VDiSC Clients

A VDiSC Client acts as the interface that connects with any given VDiSC Service. Each client connects to a single VDiSC Service via a dedicated gRPC channel. Clients provide an API to call workers, pipelines within a VDiSC Service, and a command–line interface (CLI) that allows end users to interact with the services and workers available to a given client.

Clients have a set of specific, specialized remote procedure calls that can be requested from a VDiSC Service, along with a generic call that can be made for procedures that do not fall within the general detect–extract–enroll–search architecture of computer vision. The specific remote procedure calls with dedicated implementations within the client are enumerated in Figure 3.

Each client–service pair connects via an encrypted gRPC network channel. Section 4.7 provides more detail on this channel encryption.

4.5. VDiSC Sources and Databases

VDiSC Sources implement a simple finite or indefinite iterator for streaming media. Each VDiSC Source implements a grab functionality, which "grabs" the next frame from a given camera or video file. The built-in generic VDiSC Source can read most video files and codecs, along with the ability to read Real-Time Streaming Protocol, M3U8, and gstreamer syncs [23]. Other sources that utilize proprietary software development kits, such as Pylon or Vimba, can be easily implemented and integrated into the VDiSC Ecosystem by subclassing the VDiSC Source.

VDiSC Clients connect directly to a VDiSC Source and stream their output to connected services. Because VDiSC Clients and Services need not live within the same environment, software or environment conflicts can easily be resolved by hosting a service outside of the environment on which the camera or source must run. For example, if a given machine vision camera worked only with software designed for Windows, the VDiSC Client could run within a Windows environment while streaming output to a service located on a DGX Linux environment suited for real-time processing.

VDiSC Databases are designed to store enrolled templates and embeddings extracted from various worker microservices—at a high level, implementing record enrollment, deletion, and search. The built-in VDiSC Database implementation also connects directly to the PHE layer to create a secure template storage solution that cannot leak private information. VDiSC databases are loaded as a persistent object within a VDiSC Service and are, therefore, accessible as a remote database propagated through nested VDiSC server–client connections.

4.5.1 Zero-Configuration Networking

VDiSC utilizes Zero-Configuration Networking (ZeroConf) [22], also known as Apple Bonjour, to make services discoverable within a local area network or wide area network. This feature creates a DNS-like service with which VDiSC Services are easily addressable over the network, even in the presence of changing IP addresses. Services hosting both workers and pipelines broadcast their presence with a unique name. This allows other services within the network to automatically connect and discover other workers and pipelines available to it through service-to-client connections. Using ZeroConf, VDiSC clients can discover and utilize all workers and pipelines active and visible on a given network without needing to know exact IP addresses.

4.6. Streaming and Analytics in Real Time

Because VDiSC is designed as a real-time streaming client–server framework, decisions were made with the end user in mind. In Figure 4, we show a simple graphical user interface (GUI), built directly into the VDiSC library, that can display real-time streaming results from given workers and pipelines. This GUI can be deployed either on the server or client end, or both. While the GUI is relatively bare-bones, it is designed to be easily extensible and cross-platform capable. This feature provides users of VDiSC either a out-of-the-box visualization tool for their real-time deployed systems or a straightforward guide on how to implement their own custom interface.

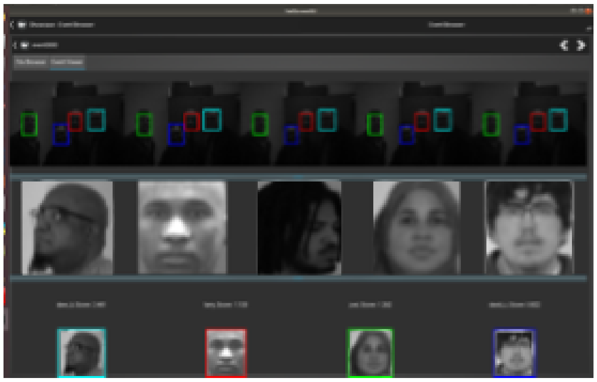

Figure 4. A simple cross-platform GUI built into VDiSC for easy visualization of biometric data.

4.7. Privacy and Network Security

4.7.1 Partially Homomorphic Encryption Layer

VDiSC implements a built-in service that provides PHE using the Paillier Cryptosystem [19]. More specifically, VDiSC utilizes a modified version of the Python implementation of Paillier PHE provided here [8]. One of the bottlenecks of the Paillier Cryptosystem is the speed and efficiency with which the modular multiplicative inverse (MMI) can be run [12]. Traditionally, this calculation is performed utilizing the extended Euclidian algorithm (EEA) [17]. However, naive implementations of the EEA can be prohibitively slow.

In [8], authors sped up this calculation for single numbers by utilizing the GMPY2 library [16], a C-based Python interface that utilizes the GNU MPFR library for multiple-precision arithmetic [9]. For VDiSC, we further vectorized this MMI using Numpy [20]. In Figure 5, we show the effectiveness of this vectorized algorithm in the use of performing vector dot products for L1 distance calculations on face templates ranging from 1 to 1,024 dimensions. As can be seen in Figure 5, the VDiSC vectorized multiplicative inverse performs almost an order of magnitude faster than the original implementation in [16].

4.7.2 Encrypted gRPC Channels

As a second layer of security, VDiSC provides one-line flags to ensure that the gRPC channel connecting a VDiSC Client and Service is encrypted using either RSA or ED25519. This allows data to be passed securely either on local gRPC channels or when being transmitted over wide area networks.

5. Where to Access VDiSC

VDiSC will be freely available and hosted on Github. The repository contains documentation as well as Jupyter Notebooks as a get-started guide for implementing VDiSC Workers and utilizing the VDiSC API and CLI.

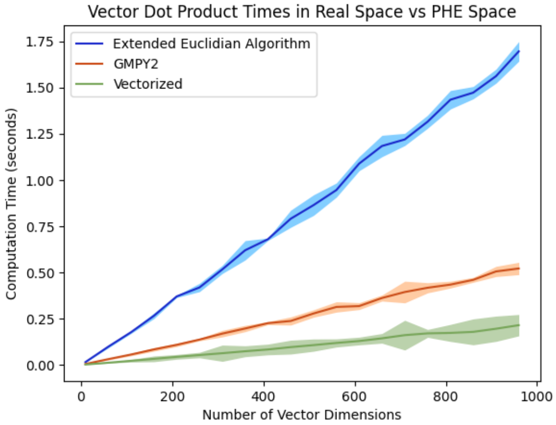

Figure 5. PHE encryption times for face vector embeddings of different dimensions. The original EEA, while the most straightforward, performs the slowest. We have markedly optimized the GNU implementation using GMPY2 (orange) to perform two to three times faster (blue) within the VDiSC framework by utilizing vectorized operations to parallelize the EEA.

Currently, VDiSC or its variants are used on multiple projects ”in the wild.” A through-windshield imaging system [7] utilizes the framework to perform its recognition. Similarly, the Deep-HDR fusion algorithm pipeline [21] utilizes the framework to route imagery from multiple camera sources to channels within the given network.

VDiSC is also being utilized in driver safety projects [11] to determine what methods of data privacy work best to protect data of drivers in driver-facing camera systems. VDiSC has also been effectively utilized for low-resource real-time computer vision on edge devices [6, 4].

6. Conclusions and Future Work

VDiSC is a ground-up reinvented declarative computer vision framework that builds upon the achievements and successes of VDiSC by increasing flexibility and efficiency and responding to the growing need for truly secure, streaming-based inferencing frameworks for use in smart city and real-time surveillance implementations.

A major avenue for future work is the continued development of the Oak Ridge Identity Testbed (ORID), a framework composed of sensors, biometric detection and identification algorithms, and computing platforms. Powered by VDiSC, ORID streams video data over the internal network from cameras deployed across Oak Ridge National Laboratory campus to one or more GPU servers for processing and routes results back out to create impactful visualizations for security operations centers and other demonstrations. It provides flexibility for rapid deployment of sensors and algorithms for testing and evaluation and the infrastructure required to build novel datasets with minimal lead time. Future work will also include integration of multimodal sensors and algorithms with VDiSC and ORID, further investigation into privacy-protecting technologies, and automated collection and annotation of novel datasets for algorithm development.

7. Acknowledgements

This research was supported in part by an appointment to the Oak Ridge National Laboratory sponsored by the U.S. Department of Energy and administered by the Oak Ridge Institute for Science and Education, the U.S. Department of Energy, and Office of Science, Office of Workforce Development for Teachers and Scientists (WDTS) under the Science Undergraduate Laboratory Internships Program(SULI), and the Texas A&M University-Kingsville, Office of University Programs, Science and Technology Directorate, Department of Homeland Security Grant Award # 2012-ST -062-000054.

Notice: This manuscript has been authored by UT-Battelle, LLC, under contract DE-AC05-00OR22725 with the US Department of Energy (DOE). The US government retains and the publisher, by accepting the article for publication, acknowledges that the US government retains a nonexclusive, paid-up, irrevocable, worldwide license to publish or reproduce the published form of this manuscript, or allow others to do so, for US government purposes. DOE will provide public access to these results of federally sponsored research in accordance with the DOE Public Access Plan (http://energy.gov/downloads/doe-public-access-plan).

Research sponsored by the Laboratory Directed Research and Development Program of Oak Ridge National Laboratory, managed by UT-Battelle, LLC, for the U. S. Department of Energy.

References

[1] Triton inference server, Aug. 2022.

[2] Abbas Acar, Hidayet Aksu, A. Selcuk Uluagac, and Mauro Conti. A survey on homomorphic encryption schemes. *ACM Computing Surveys*, 51(4):1–35, 2019.

[3] David S. Bolme, David C. Cornett III, and Nisha Srinivas. FaRO: FAce Recognition from Oak ridge. https://github.com/ORNL/faro, 2019.

[4] David S. Bolme, Hector J. Santos-Villalobos, and David C. Cornett III. Rifle-like camera for long distance face recognition, Mar. 25 2021. US Patent App. 17/024,855.

[5] David S. Bolme, Nisha Srinivas, Joel Brogan, and David Cornett. Face recognition oak ridge (faro): A framework for distributed and scalable biometrics applications. In *2020 IEEE International Joint Conference on Biometrics (IJCB)*, pages 1–8. IEEE, 2020.

[6] David S. Bolme, Hector J. Santos Villalobos, and Aravind K. Mikkilineni. Handheld multi-sensor biometric imaging device and processing pipeline, May 24 2022. US Patent 11,341,224.

[7] David Cornett, Alec Yen, Grace Nayola, Diane Montez, Christi R. Johnson, Seth T. Baird, Hector Santos-Villalobos, and David S. Bolme. Through the windshield driver recognition. *Electronic Imaging*, 2019(13):140–141, 2019.

[8] CSIRO's Data61. Python paillier library. `https://github.com/data61/python-paillier`, 2013.

[9] Laurent Fousse, Guillaume Hanrot, Vincent Lefèvre, Patrick Pélissier, and Paul Zimmermann. Mpfr: A multiple-precision binary floating-point library with correct rounding. *ACM Transactions on Mathematical Software (TOMS)*, 33(2):13–es, 2007.

[10] Yury Gorbachev, Mikhail Fedorov, Iliya Slavutin, Artyom Tugarev, Marat Fatekhov, and Yaroslav Tarkan. Openvino deep learning workbench: Comprehensive analysis and tuning of neural networks inference. In *Proceedings of the IEEE/CVF International Conference on Computer Vision Workshops*, pages 0–0, 2019.

[11] Kimberley D. Orsten Hooge, Asal Baragchizadeh, Thomas P. Karnowski, David S. Bolme, Regina Ferrell, Parisa R. Jesudasen, Carlos D. Castillo, and Alice J. OToole. Evaluating automated face identity-masking methods with human perception and a deep convolutional neural network. *ACM Transactions on Applied Perception (TAP)*, 18(1):1–20, 2020.

[12] Zhengbing Hu, IA Dychka, Onai Mykola, and Bartkoviak Andrii. The analysis and investigation of multiplicative inverse searching methods in the ring of integers modulo m. *International Journal of Intelligent Systems and Applications*, 8(11):9, 2016.

[13] Doug Hudgeon and Richard Nichol. Machine learning for business: Using amazon sagemaker and jupyter, 2020.

[14] Latif U. Khan, Ibrar Yaqoob, Nguyen H. Tran, S.M. Ahsan Kazmi, Tri Nguyen Dang, and Choong Seon Hong. Edge-computing-enabled smart cities: A comprehensive survey. *IEEE Internet of Things Journal*, 7(10):10200–10232, 2020.

[15] Christopher Grant Kirwan. Defining the middle ground: A comprehensive approach to the planning, design and implementation of smart city operating systems. In *International Conference on Cross-Cultural Design*, pages 316–327. Springer, 2015.

[16] Alex Martelli. Aleaxit/gmpy: General multi-precision arithmetic for python 2.6+/3+ (gmp, mpir, mpfr, mpc).

[17] JAM Naranjo, JA López-Ramos, and LG Casado. Applications of the extended euclidean algorithm to privacy and secure communications. In *Proceedings of 10th International Conference on Computational and Mathematical Methods in Science and Engineering*, pages 702–713, 2010.

[18] Vishnu Naresh Boddeti. Secure face matching using fully homomorphic encryption. In *2018 IEEE 9th International Conference on Biometrics Theory, Applications and Systems (BTAS)*, pages 1–10, 2018.

[19] Michael OKeeffe. The paillier cryptosystem. *Mathematics Department April*, 18:1–16, 2008.

[20] Contributors to Wikimedia projects. Algorithm implementation/mathematics/extended euclidean algorithm, Feb 2021.

[21] Max Ruby, David S. Bolme, Joel Brogan, David Cornett III, Baldemar Delgado, Gavin Jager, Christi Johnson, Jose Martinez-Mendoza, Hector Santos-Villalobos, and Nisha Srinivas. The mertens unrolled network (mu-net): A high dynamic range fusion neural network for through the windshield driver recognition. In *Autonomous Systems: Sensors, Processing, and Security for Vehicles and Infrastructure 2020*, volume 11415, pages 72–83. SPIE, 2020.

[22] Ion Stoica, Daniel Adkins, Shelley Zhuang, Scott Shenker, and Sonesh Surana. Internet indirection infrastructure. In *Proceedings of the 2002 Conference on Applications, Technologies, Architectures, and Protocols for Computer Communications*, pages 73–86, 2002.

[23] Wim Taymans, Steve Baker, Andy Wingo, Rondald S. Bultje, and Stefan Kost. Gstreamer application development manual (1.2.3). *Publicado en la Web*, 72, 2013.

[24] Emily Wenger, Francesca Falzon, Josephine Passananti, Haitao Zheng, and Ben Y. Zhao. Assessing privacy risks from feature vector reconstruction attacks, 2022.