SYFT 0.5: A PLATFORM FOR UNIVERSALLY DEPLOYABLE STRUCTURED TRANSPARENCY

Adam James Hall
Edinburgh Napier University / OpenMined
adam@openmined.org

Madhava Jay
OpenMined
madhava@openmined.org

Tudor Cebere
OpenMined
tudor@openmined.org

Bogdan Cebere
OpenMined
bogdan.cebere@gmail.com

Koen Lennart van der Veen
Memri
koen.vanderveen@memri.cloud

George Muraru
Politehnica University of Bucharest / OpenMined
george.muraru@openmined.org

Tongye Xu
Philisense / OpenMined
xutongye@philisense.com

Patrick Cason
OpenMined
patrick@openmined.org

William Abramson
Edinburgh Napier University / OpenMined
will.abramson@napier.ac.uk

Ayoub Benaissa
École Supérieure en Informatique,
Sidi Bel Abbès / OpenMined
a.benaissa@esi-sba.dz

Chinmay Shah
OpenMined
chinmayshah3899@gmail.com

Alan Aboudib
OpenMined
agabudeeb@gmail.com

Théo Ryffel
INRIA Paris ENS PSL University Arkhn OpenMined
theo.ryffel@ens.fr

Kritika Prakash
IIIT Hyderabad / OpenMined
kritika.prakash@research.iiit.ac.in

Tom Titcombe
Tessella / OpenMined
tom.titcombe@tessella.com

Varun Kumar Khare
OpenMined
varunkhare1234@gmail.com

Maddie Shang
OpenMined
shang.madeleine@gmail.com

Ionesio Junior
OpenMined

Animesh Gupta
OpenMined

Jason Paulmier
OpenMined

Nahua Kang
OpenMined

Andrew Trask
The University of Oxford / OpenMined
andrew@openmined.org

ABSTRACT

We present Syft, a general-purpose framework which combines a core group of privacy-enhancing technologies that facilitate a universal set of structured transparency systems. This framework is demonstrated through the design and implementation of a novel privacy-preserving inference information flow where we

pass homomorphically encrypted activation signals through a split neural network for inference. We show that splitting the model further up the computation chain significantly reduces the computation time of inference and the payload size of activation signals at the cost of model secrecy. We evaluate our proposed flow with respect to it's provision of the core structural transparency principles.

1 INTRODUCTION

The centralisation of resources in machine learning information flows forces a Pareto efficiency trade-off between data utility and privacy. When participating in these flows, data owners cannot know that their data has not been sold, retained for far longer than intended or otherwise used for purposes outside the understood context-relative informational norms Nissenbaum (2004). A key risk factor here is the data copy problem. Data once copied, even by well meaning actors, cannot ever be guaranteed to have been destroyed, as has become painfully clear by the take down of the DukeMTMC dataset (Ristani et al. (2016)). Controversy surrounding the unintended usage of the public dataset was raised when faces of ordinary people were used without permission to serve questionable ends. This resulted in the data being revoked, breaking the social context of the data subject's participation in that flow (Peng (2020)). Copies of the data and derivative datasets were still freely available on the internet and academic papers are still being published. This is such a pervasive issue that websites exist for tracking the leakage and misuse of peoples facial data for inappropriate, extra-contextual use cases (Harvey (2021)).

The increasing prevalence of organised criminal groups online remains a risk, forcing centralised data processors to respect informational norms and securely store the user's data. Adequate maintenance of confidentiality, integrity, and availability of data represents an increasing liability for processors. Between 2015 and 2020, cybersecurity spending worldwide almost doubled from $75.6 billion to $124 (Lee (2021)). The recent General Data Protection Regulation (GDPR) legislation (EU) requires explicit consent from data subjects to allow the processing of their data for any purpose. GDPR and similar laws represent a formidable bureaucratic overhead to researchers who process data concerning EU citizens. This issue is further compounded by a lack of trust between data subjects and data processors. Under these circumstances, research on private data is blocked due to privacy ramifications- without ever contributing to an information flow's contextually driven purpose or values. If these obstacles are removed, research goes ahead with potentially disastrous social and political consequences for the data subjects and their community.

In this work, we contribute an open-source, universally deployable framework capable of facilitating information flows that cater to the core tenets of Structured Tranansparency (ST); giving actors the confidence to participate with sensitive information. Our tool provides a broader gamut of privacy-enhancing technologies (PETs) than existing frameworks (Table 1) with the ability to accommodate ST in an increasingly nuanced set of information flows. We define a novel, structurally transparent information flow that utilises the split learning technique to reduce computation overhead in homomorphically encrypted inference.

2 FRAMEWORK DESCRIPTION

Syft allows data scientists to perform compute data they don't own and cannot see. This is achieved through the separation of concerns between where data resides and where code is executed. Syft allows for remote computation on any object or library that has been mapped to the Abstract Syntax Tree (AST). From a user perspective, remote variables are operated upon locally as network pointers. Operations executed on these network pointers are encapsulated as a remote procedure call performed on the remote machine. Agents can be referenced to resolve the verifiable attributes of credentials or data and models in the store without exposing a persistent identity- facilitating zero-knowledge access management as outlined in Abramson et al. (2020).

After a WebRTC connection is established between peers, privacy is guaranteed through the remote execution model in which you compute operations on data without revealing it or the underlying permission scheme. Even if data is shared, it can still be encrypted using our secure multi-party computation solution, SyMPC, or homomorphic encryption through TenSEAL. Intermediary computation and results are held in a store controlled by the data owner. The data owner is able to

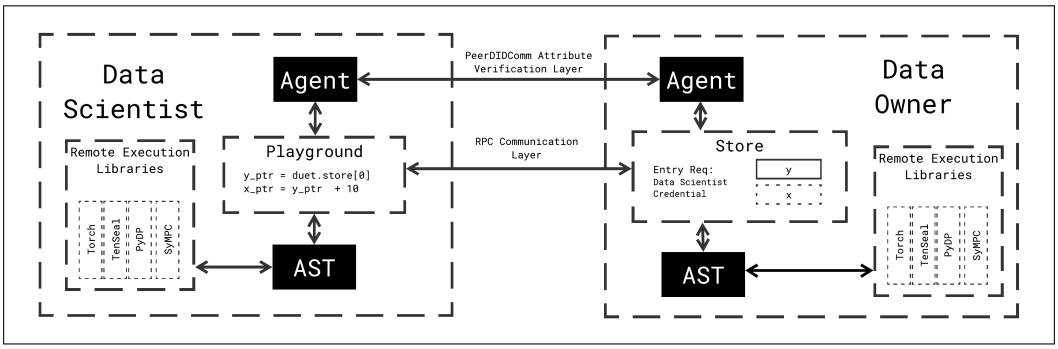

Figure 1: Duet Architecture Diagram

approve or deny requests to access the data in the store by either manual approval (by hand) or automatic approval via a policy.

As an API design, Syft aims to provide a near transparent layer around supported frameworks and libraries so that data scientists don't need to learn new tools to develop their algorithms in a privacy-friendly fashion. Users should expect their existing code to run with minimal change and to find writing new code within Syft intuitive. Syft significantly lowers the entry-level of access to cryptography, differential-privacy and distributed computing for researchers and practitioners.

3 Structured Transparency

ST is an extension of the contextual integrity framework (Nissenbaum (2009)). Contextual integrity views privacy in terms of information flowing between actors in roles governed by context-relative informational norms. Nissembaum's central thesis is that people feel their privacy is violated when these informational norms are broken, for example when unanticipated information is learnt by unexpected actors (Nissenbaum (2009)). The framework leaves room for positive changes to these informational norms, suggesting that if entrenched contextual integrity is breached by new systems then they should be evaluated against whether these changes contribute to the purpose and values of the context. Contextual integrity emphasises the importance of privacy, while challenging older definitions of privacy as either control or access to information Gavison (1980); Westin (1968). Instead, privacy is presented as the appropriate flow of information relative to a context, hence including control and access within a broader social framing (Nissenbaum (2004; 2009)).

ST extends contextual integrity's definition of an information flow by introducing a more nuanced set of properties of the flow. These properties take into account new possibilities for structuring the context and relative informational norms around an information flow. PETs implemented in Syft and presented in this paper, explicitly seek to facilitate these information flows. Information flows from a sender to a recipient and the information pertains to a data subject, who could possibly not be the sender. The context within which this information flow is embedded prescribes the roles, activities, norms and purposes for which information flows take meaning from and are justified against. ST proposes additional definitions such as for input and output privacy, input and output verification and flow governance (Trask et al. (2020)).

We seek to structure the transparency of an information flow by combining novel PETs for computation Cramer et al. (2015), encryption (Gentry (2009); Cheon et al. (2017)) and authentication (Diffie & Hellman (1976); Camenisch & Lysyanskaya (2001)). Such techniques can give confidence to the sender, recipients and subjects of an information flow that the information is integrity assured, verifiably from a sender and only visible to the set of actors and roles defined in the flow structure. Specifically, concerning machine learning use cases, the information contributed to a flow shouldn't be revealed or leaked at any step, only derivatives of this information aggregated to extract features without compromising the underlying information.

4 ENCRYPTED SPLIT-INFERENCE WITH STRUCTURAL TRANSPARENCY GUARANTEES

Our work is demonstrated through Duet, a 2-party scenario using the Syft framework. One party with ownership over data provides limited access to a model owned by another actor in order to conduct inference. We define an information flow wherein a data subject receives the result of a prediction performed on their private data using another entity's private model. These private information assets owned by the Data Owner (DO) and Data Scientist (DS) respectively must interact to produce a prediction to be consumed by the DO. In a societal context, this may be governed under the same norms as a user consuming inference as a service. We use this context to evaluate the ST guarantees of the encrypted inference flow in Figure 4. In the actual experiment we use MNIST (Yann LeCun, Corinna Cortes, Christopher J.C. Burges (2019)) images as data and a deep convolutional neural network as our model.

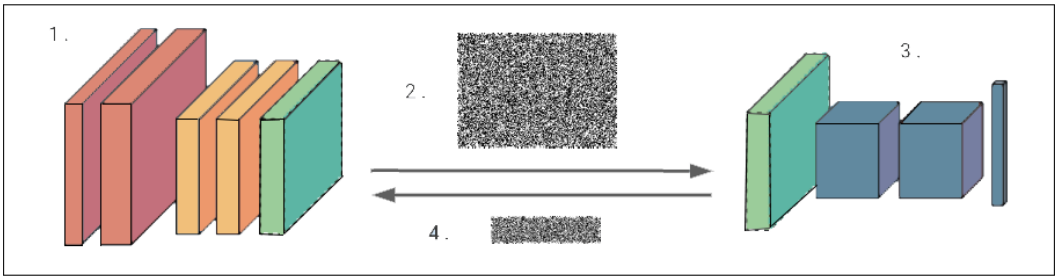

Figure 2: Private Inference Flow showing enccrypted activation signals being sent away and processed by a remote server. 1. DO segment, 2. Encrypted Activation Signal 3. DS segment 4. Encrypted Output

Governance - Before this information flow may begin, we establish the context between actors and the norms that flow from this context. This is performed in step 1 of Figure 4 where DO and DS verify attributes about one another. In Appendix A.1, Figure 3, the DS proves they are a DS using the verifiable credential they received from an authority. Similarly, the DO could declare their role as a DO suitable to participate. The social context of this flow is defined through the presentation of their relevant attributes under CL Signatures (Camenisch & Lysyanskaya (2001)) through their Agent.

Input Verification - This section refers to steps two and three in Figure 4, where DO and DS load their model and data. The parameters are composed of a model and some data; both of which are private. In this case, the DO is also the consumer of the inference so we are not concerned with their inputs in our threat model. We do however, want to make sure that the DS has the model we're interested in. This is possible through storing the model object in an Agent like a verifiable credential. Similarly, this can be done with datasets to define verifiable properties like completeness, provenance and scheme conformity.

Input Privacy - This section refers to steps four to eight in Figure 4 and is described visually in Figure 2 which describes the private inference flow. In order to maintain the privacy of the DO's data and the DS's model, the DO could encrypt their data and send it to the DS for private inference using CKKS Homomorphic Encrpytion (Cheon et al. (2020; 2017)) Takabi et al. (2019). However, high dimensional data incurs a significant computation overhead during encrypted computation and increased computation depth necessitates a larger polynomial modulus- increasing ciphertext size and computation requirements. Alternatively the DS may opt to only share a portion of their model with DO and execute a split neural network (SplitNN) training flow Vepakomma et al. (2018a;b). Data remains with the DO at inference time and only activation signal is shared. However, statistical information still persists in activation signals, despite being representing less information than the original data. The information contained in activation signals can be used by a malicious DS to exploit information leakage Vepakomma et al. (2019). The Shredder technique (Mireshghallah et al. (2020)) may also be used to apply noise to activation signals before transit, however, this does not provide as strong input privacy as encryption.

A hybrid approach is proposed which offers a trade-off between computational complexity and model privacy. We allow the DO to compute a portion of inference in plaintext and a the latter portion as ciphertext with the DS. The more model layers that are shared to the DO, the less computation depth is needed and the modulus coefficient is minimised. This results in transmitted ciphertext size and computation times which are dramatically reduced from 269.86KB to 139.6KB and 4.17s to 97ms respectively (Appendix Tables 2 and 3). However, sharing too many layers exposes the model to theft.

Output Privacy - is achieved here with respect to the output recipient in step 9 of Figure 4 where the DO decrypts the inference result locally. However, the real issue here is not that the output of the flow may reveal the DO's inputs, the DO is the consumer of the output. It's that over enough inferences, the DO may be able to infer membership of elements in the training set or perform model inversion attacks on the model (Shokri et al. (2017); Chen et al. (2020); Zhang et al. (2020)). To protect the privacy of the information in the model, we train using the Opacus library and the PrivacyEngine utility it provides for differentially private stochastic gradient descent (Facebook Differential Privacy). This allows us to fit our model to the general curve of the dataset rather than over-fitting. This obstructs an attackers ability to leverage the data in that data model (Abadi et al. (2016)). This DP training step is considered a step zero and is not included in this inference flow. However, it is implemented in our experiment source code.

Output Verification - In this flow output verification may relate to the removal of any bias in the model or data. The DO is the only stakeholder in the output and we can trust his data, however we should be concerned with veracity of the DS model. At the moment, we rely on the credibility of the authority that attested DS credential in step 1 of Figure 4 where DO and DS exchange credentials. This isn't fine-grained trust. However, with Aries infrastructure we can define and deploy schemes which may track the varacity and performance of models for presentation to inference consumers fit to the requirements of any regulation for model governance use cases.

5 Conclusions

Contextual integrity and ST of information flows provide the theoretical framework for privacy at scale. While privacy has traditionally been viewed as a permission system problem alone, Syft delivers privacy at all levels; by leveraging multiple cryptographic protocols and distributed algorithms, it enables new ways to structure information flows. The work being demonstrated is one such novel flow, containing ST guarantees in a 2-party setting, or Duet. Duet provides a research-friendly API for a Data Owner to privately expose their data, while a Data Scientist can access or manipulate the data on the owner's side through a zero-knowledge access control mechanism. This framework is designed to lower the barrier between research and privacy-preserving mechanisms, so that scientific progress can be made on data that is currently inaccessible or tightly controlled. In future work, Syft will integrate further with PyGrid (Ionesio Junior, Patrick Cason (2021)), to provide a user interface for organisational policy controls, cloud orchestration, dashboards and dataset management; to allow research institutions to utilise Syft within their existing research data policy frameworks.

References

Regulation (eu) 2016/679 of the european parliament and of the council. URL https://www.legislation.gov.uk/eur/2016/679/article/5.

Martin Abadi, Andy Chu, Ian Goodfellow, H Brendan McMahan, Ilya Mironov, Kunal Talwar, and Li Zhang. Deep learning with differential privacy. In *Proceedings of the 2016 ACM SIGSAC Conference on Computer and Communications Security*, pp. 308–318, 2016.

Alan Aboudib. Syfertext, 2021. URL https://github.com/OpenMined/Syfertext.

Will Abramson, Adam James Hall, Pavlos Papadopoulos, Nikolaos Pitropakis, and William J. Buchanan. A distributed trust framework for privacy-preserving machine learning. *Lecture Notes in Computer Science*, pp. 205–220, 2020. ISSN 1611-3349. doi: 10.1007/978-3-030-58986-8_14. URL http://dx.doi.org/10.1007/978-3-030-58986-8_14.

Nick Angelou, Ayoub Benaissa, Bogdan Cebere, William Clark, Adam James Hall, Michael A. Hoeh, Daniel Liu, Pavlos Papadopoulos, Robin Roehm, Robert Sandmann, Phillipp Schoppmann,

and Tom Titcombe. Asymmetric private set intersection with applications to contact tracing and private vertical federated machine learning, 2020.

Donald Beaver. Efficient multiparty protocols using circuit randomization. volume 576, pp. 420–432, 08 1991. ISBN 978-3-540-55188-1. doi: 10.1007/3-540-46766-1_34.

Daniel J. Beutel, Taner Topal, Akhil Mathur, Xinchi Qiu, Titouan Parcollet, and Nicholas D. Lane. Flower: A friendly federated learning research framework, 2020.

Sebastian Caldas, Peter Wu, Tian Li, Jakub Konecný, H. Brendan McMahan, Virginia Smith, and Ameet Talwalkar. LEAF: A benchmark for federated settings. *CoRR*, abs/1812.01097, 2018. URL http://arxiv.org/abs/1812.01097.

Jan Camenisch and Anna Lysyanskaya. An efficient system for non-transferable anonymous credentials with optional anonymity revocation. In *International conference on the theory and applications of cryptographic techniques*, pp. 93–118. Springer, 2001.

Si Chen, Ruoxi Jia, and Guo-Jun Qi. Improved techniques for model inversion attacks, 2020.

Jung Hee Cheon, Andrey Kim, Miran Kim, and Yongsoo Song. Homomorphic encryption for arithmetic of approximate numbers. In *International Conference on the Theory and Application of Cryptology and Information Security*, pp. 409–437. Springer, 2017.

Jung Hee Cheon, Seungwan Hong, and Duhyeong Kim. Remark on the security of ckks scheme in practice. *IACR Cryptol. ePrint Arch*, 2020:1581, 2020.

Ronald Cramer, Ivan Bjerre Damgård, et al. *Secure multiparty computation*. Cambridge University Press, 2015.

Georgios Damaskinos, Rachid Guerraoui, Anne-Marie Kermarrec, Vlad Nitu, Rhicheek Patra, and Francois Taiani. Fleet. *Proceedings of the 21st International Middleware Conference*, Dec 2020. doi: 10.1145/3423211.3425685. URL http://dx.doi.org/10.1145/3423211.3425685.

Whitfield Diffie and Martin Hellman. New directions in cryptography. *IEEE transactions on Information Theory*, 22(6):644–654, 1976.

Facebook Differential Privacy. Opacus differential privacy. URL https://github.com/pytorch/opacus.

Ruth Gavison. Privacy and the limits of law. *The Yale law journal*, 89(3):421–471, 1980.

Craig Gentry. Fully homomorphic encryption using ideal lattices. In *Proceedings of the forty-first annual ACM symposium on Theory of computing*, pp. 169–178, 2009.

Jules. Harvey, Adam. LaPlace. Exposing.ai, 2021. URL https://exposing.ai.

Chaoyang He, Songze Li, Jinhyun So, Xiao Zeng, Mi Zhang, Hongyi Wang, Xiaoyang Wang, Praneeth Vepakomma, Abhishek Singh, Hang Qiu, Xinghua Zhu, Jianzong Wang, Li Shen, Peilin Zhao, Yan Kang, Yang Liu, Ramesh Raskar, Qiang Yang, Murali Annavaram, and Salman Avestimehr. Fedml: A research library and benchmark for federated machine learning, 2020.

Cao Hui, Shubo Liu, Renfang Zhao, and Xingxing Xiong. Ifed: A novel federated learning framework for local differential privacy in power internet of things. *International Journal of Distributed Sensor Networks*, 16:155014772091969, 05 2020. doi: 10.1177/1550147720919698.

Alex Ingerman and Krzys Ostrowski. Introducing tensorflow federated., 2019. URL blog.tensorflow.org.

Ionesio Junior, Patrick Cason. PyGrid, 2021. URL https://github.com/OpenMined/PyGrid.

Brian Knott, Shobha Venkataraman, Shubho Sengupta Awni Hannun, Mark Ibrahim, and Laurens van der Maaten. Crypten: Secure multi-party computation meets machine learning, 2021. URL https://crypten.ai/.

In Lee. Cybersecurity: Risk management framework and investment cost analysis. *Business Horizons*, 2021.

Heiko Ludwig, Nathalie Baracaldo, Gegi Thomas, Yi Zhou, Ali Anwar, Shashank Rajamoni, Yuya Ong, Jayaram Radhakrishnan, Ashish Verma, Mathieu Sinn, Mark Purcell, Ambrish Rawat, Tran Minh, Naoise Holohan, Supriyo Chakraborty, Shalisha Whitherspoon, Dean Steuer, Laura Wynter, Hifaz Hassan, Sean Laguna, Mikhail Yurochkin, Mayank Agarwal, Ebube Chuba, and Annie Abay. Ibm federated learning: an enterprise framework white paper v0.1, 2020.

Max Wong, H Moster, and Lin, Bryce. Eggroll, 2018. URL https://github.com/WeBankFinTech/eggroll.

Microsoft SEAL. Microsoft SEAL (release 3.6). https://github.com/Microsoft/SEAL, November 2020. Microsoft Research, Redmond, WA.

Fatemehsadat Mireshghallah, Mohammadkazem Taram, Prakash Ramrakhyani, Ali Jalali, Dean Tullsen, and Hadi Esmaeilzadeh. Shredder: Learning noise distributions to protect inference privacy. In *Proceedings of the Twenty-Fifth International Conference on Architectural Support for Programming Languages and Operating Systems*, pp. 3–18, 2020.

Helen Nissenbaum. Privacy as contextual integrity. *Wash. L. Rev.*, 79:119, 2004.

Helen Nissenbaum. *Privacy in context: Technology, policy, and the integrity of social life*. Stanford University Press, 2009.

Nvidia. Clara, 2019. URL https://blogs.nvidia.com/blog/2019/12/01/clara-federated-learning/.

Kenny Peng. Facial recognition datasets are being widely used despite being taken down due to ethical concerns. here's how., 2020. URL https://freedom-to-tinker.com/2020/10/21/facial-recognition-datasets-are-being-widely-used-despite-being-taken-down-due-to-

Qinghe Jing, Dong Daxiang, and contributors. Paddlefl, 2019. URL https://github.com/PaddlePaddle/PaddleFL.

Ergys Ristani, Francesco Solera, Roger S. Zou, R. Cucchiara, and Carlo Tomasi. Performance measures and a data set for multi-target, multi-camera tracking. In *ECCV Workshops*, 2016.

Sherpa. We Research and Build Artificial Intelligence Technology and Services, 2019. URL https://github.com/sherpaai/Sherpa.ai-Federated-Learning-Framework.

Reza Shokri, Marco Stronati, Congzheng Song, and Vitaly Shmatikov. Membership inference attacks against machine learning models. In *2017 IEEE Symposium on Security and Privacy (SP)*, pp. 3–18. IEEE, 2017.

Daniel Takabi, Robert Podschwadt, Jeff Druce, Curt Wu, and Kevin Procopio. Privacy preserving neural network inference on encrypted data with gpus. *CoRR*, abs/1911.11377, 2019. URL http://arxiv.org/abs/1911.11377.

TenSEAL. TenSEAL (release 0.3.0). https://github.com/OpenMined/TenSEAL, February 2021.

Andrew Trask and Kritika Prakash. Towards general-purpose infrastructure for protecting scientific data under study, 2020.

Andrew Trask, Emma Bluemke, Ben Garfinkel, Claudia Ghezzou Cuervas-Mons, and Allan Dafoe. Beyond privacy trade-offs with structured transparency, 2020.

Praneeth Vepakomma, Otkrist Gupta, Tristan Swedish, and Ramesh Raskar. Split learning for health: Distributed deep learning without sharing raw patient data. *arXiv preprint arXiv:1812.00564*, 2018a.

Praneeth Vepakomma, Tristan Swedish, Ramesh Raskar, Otkrist Gupta, and Abhimanyu Dubey. No peek: A survey of private distributed deep learning, 2018b.

Praneeth Vepakomma, Otkrist Gupta, Abhimanyu Dubey, and Ramesh Raskar. Reducing leakage in distributed deep learning for sensitive health data. *arXiv preprint arXiv:1812.00564*, 2019.

H. Wang, Z. Kaplan, D. Niu, and B. Li. Optimizing federated learning on non-iid data with reinforcement learning. In *IEEE INFOCOM 2020 - IEEE Conference on Computer Communications*, pp. 1698–1707, 2020. doi: 10.1109/INFOCOM41043.2020.9155494.

Wenbin Wei. FATE Documentation, 2018. URL `https://github.com/FederatedAI/FATE`.

Alan F Westin. Privacy and freedom. *Washington and Lee Law Review*, 25(1):166, 1968.

Will Abramson, Adam Hall, Lohan Spies, Tom Titcombe. PyDentity, 2021. URL `https://github.com/OpenMined/PyDentity`.

Yann LeCun, Corinna Cortes, Christopher J.C. Burges. THE MNIST DATABASE of handwritten digits, 2019. URL `http://yann.lecun.com/exdb/mnist/`.

Yuheng Zhang, Ruoxi Jia, Hengzhi Pei, Wenxiao Wang, Bo Li, and Dawn Song. The secret revealer: Generative model-inversion attacks against deep neural networks. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2020.

6 Acknowledgments

The authors would like to thank the OpenMined community, without which PySyft would not be possible. In particular, the authors of this paper would like to give acknowledge to contribution of; Karl Higley, Anshuman Singh, Anubhav Raj Singh, Ariann Farias, Arturo Marquez Flores, Avinash Swaminathan, Ben Fielding, Chirag Gomber, Chitresh Goel, Harkirat Singh, Hideaki Takahashi, Irina Bejan, Jason Paulmier, Jirka Borovec, Joel Lee, Lee Yi Jie Joel, Nahua Kang, Nicolas Remerscheid, Param Mirani, Plamen Hristov, Raghav Prabhakar, Rima Al Shikh, Vaibhav Vardhan, Wansoo Kim and Zarreen Naowal Reza.

A Appendix

A.1 Aries Agent Verification flow

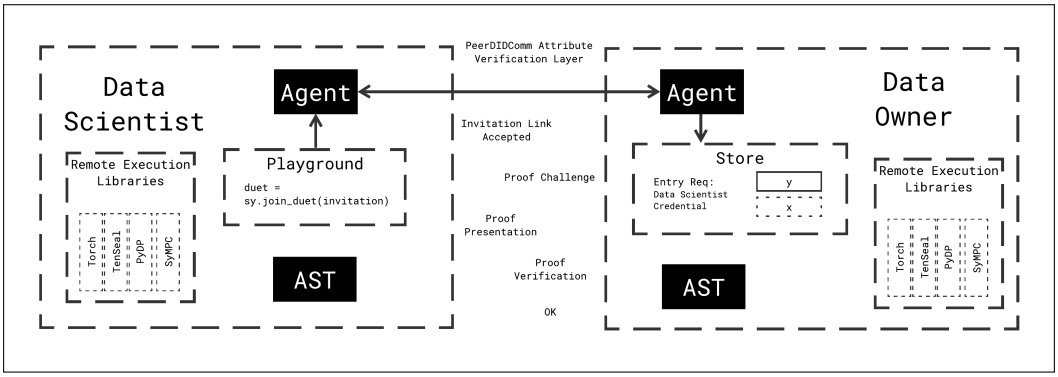

Figure 3: Aries Agent Verification flow

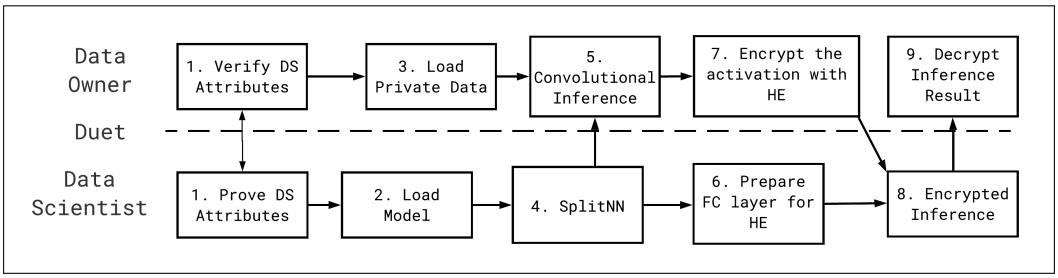

Figure 4: Private Inference Flow

A.2 Information flow

A.3 Architecture Concepts

Data Owner / Data Scientist - The Data Owner (DO) is the Duet session creator, the party that has implicit control over the store and permissions. The Data Scientist (DS), is the general terminology for the peer node which connects to a DO, and has tightly controlled limitations on access to objects in the store. There are multi-party configurations of peers which provide more complex scenarios which we won't detail here.

Node & Client - In Syft the smallest unit of secure communication and computation is a Node. These Nodes are capable of running a variety of services. By registering new message types and code that executes on receipt of those messages; services augment the capability of a Node with additional functionality. To ensure validity, all messages are signed by their sender and include a verification key. Signing is done with a PyNaCl a Python libsodium wrapper using 256-bit Ed25519. When a message arrives it is verified and then the delivery address is checked. Nodes are designed to be aware of other Nodes, and are able to forward on messages they receive to other known addresses. Once a message arrives at its destination, it is delivered to the appropriate service for execution.

Nodes have the concept of a root verification key and all services can opt to allow execution by messages signed by root clients only. In the event of an error, exceptions can be transmitted back to the sender, defaulting to UnknownPrivateException and allowing for careful exposure of real exceptions when safe and relevant. Nodes can optionally contain a key-value store, which can be backed with memory or disk.

To provide a public API interface for users to communicate with a Node, we have a Client which provides a handle to easily invoke remote functions and view metadata about a remote Node. While there are two Nodes in the Figure 1 Duet Architecture, there is asymmetry with the DO side having a Store and both Clients having a handle to the DO Node. In this sense all transfer of data is handled by requests to the DO node and approvals by the DO root client to those requests. Additionally the DO's Client never explicitly sends data or execution requests to the DS Node. This configuration provides a streamlined work flow in scenarios where one side has private data they wish to host.

While this is one possible topology, due to the flexible design, other architectures of two or more parties are possible over any size or shape of network. By simply adding additional services to a Node, entirely new networked functionality can be constructed. This mechanism is how PyGrid is able to augment Syft to create Cloud and on-prem worker distribution and data storage.

The Store - Duet shares data through the serialization of objects as Protobuf messages. These messages are wrapped in a Storable Object (SO) interface that handles metadata and interaction with a Store. The Store may be located either in-memory or on disk. In Duet, the Store is located within the Python process of the DO. However, because all the store interaction is done through serialized communication, nothing prevents the Store from existing on a completely separate process or network node. The Store hosts all the intermediate data created in the process of remote execution. Networked Pointers require the concept of a Distributed Garbage Collection (DGC) mechanism, which is responsible for tracking and freeing objects from the Store when certain conditions are met.

Abstract Syntax Tree - The Abstract Syntax Tree (AST) is responsible for resolving the correct remote procedure call paths. The AST is a tree that maps any Python module or object to its desired remote behaviour, handling chained pointer generation and message creation on pointer method invocation. This one to one mirror with the existing Python module system makes it trivial to support nearly any third-party Python library with Syft.

A node in the AST can represent anything that a normal Python package can provide. Each node contains references to the actual module, class or callable object and, at the leaves of this tree, all attributes have a return type which determines the synthesised pointer class that is generated when they are invoked remotely. When the caller invokes a remote node via either a handle to the remote systems AST tree or an existing network pointer, a new network pointer is immediately created of the expected return type. A message is then dispatched to the remote system where the AST is used to locate the original reference and execute it; placing the real result into the Store. Under this model only functionality within Python which is explicitly allowlisted and loaded during startup or later via an explicit loading command, can be executed remotely.

Communication Protocol - Under the Duet architecture, Syft's network code only supports peer-to-peer connections between two nodes. To connect two peers, by default Duet initialises an outgoing connection to a STUN server operated by OpenMined. STUN is a technology commonly used by video conferencing software which is designed to allow applications to traverse NAT and firewalls. It works by establishing an outgoing connection first to allow any subsequent external traffic on the same port to be routed to the application which made the outgoing request. The STUN server brokers both connections by allowing each side to establish an outgoing connection first. Once paired, the individual peers establish a connection to each other using WebRTC. This connection uses the DataChannel API from WebRTC over UDP, using DTLS as an integrity mechanism. After connection no further traffic is sent to the STUN server. Additionally the source code and instructions to run a copy of this service is available, and connection to a self run STUN service only requires adding a single URL parameter to the Duet initialisation function.

Pointers - Pointers are dynamic objects created by the AST to provide proxy control of remote objects. Pointers are usually generated from the known return type of a computation. Pointers map the functionality of a remote object to a local object by wrapping and handling the communication layer. Pointers are also responsible for garbage collecting objects which are no longer reachable. The Data Scientist can create remote objects on the Data Owner's side through messages that are sent to the DO. The implementation of the garbage collection (GC) process relies heavily on the underlying Python GC implementation. When the local pointer, that resides on the DS side, goes out of scope, a special Protobuf message is sent to the store. The sole purpose of this message is to remove the actual object that the DS pointer was referring to. This mechanism assumes that an object created by a Data Scientist would not be referred by another user, but it could easily be extended to use a reference counting mechanism where multiple nodes can hold a pointer to the same object.

A.4 LIBRARIES

Syft becomes truly powerful when remote computation is extended with the functionality provided by other high performance computing libraries or privacy enhancing frameworks. These tools can be aggregated to support the core components of structurally transparent information flows.

PyTorch - is our initial support of high performance computing on tensors, and the majority of the PyTorch API can be used in Syft. An important note is that our current architecture is not dependent on PyTorch, and our roadmap includes support for other tensor libraries like Numpy, Tensorflow, JAX and scikit-learn.

Opacus - enables PyTorch models to be trained with respect to privacy with minimal changes in the codebase and infrastructure. It can be used to train PyTorch models (Facebook Differential Privacy).

TenSEAL - a library that enables machine learning frameworks to work on encrypted data, using homomorphic encryption schemes implemented in Microsoft SEAL.

SyMPC - SyMPC is a relatively young secure multi party computation library developed in-house. It can not be used as a standalone piece of software since it highly relies on the communication primitives that can be found in Syft. Because of this, you would need to install SyMPC alongside

Syft if you want to use any of the implemented functionalities. For the moment, since SyMPC it is still at the beginning of the journey, it has some basic arithmetic operations between secretly shared values and secret/public values. For computing the correct result for simple multiplication and matrix multiplication there are employed some triples (presented in Beaver (1991). These primitives are generated by a Trusted Third Party and in our case we presume it is the node that orchestrates the entire computation. A part of the design decisions and implementation details are taken from the Facebook Research Project - CrypTen (Knott et al. (2021)). SyMPC aims is to offer the possibility to train/run inference on opaque data using a range of different protocols depending on each Data Scientist use case. As an implicit goal, the library should provide a simple way to implement new protocols, regardless of the underlying ML framework that you use.

AriesExchanger - Aries agents facilitate the construction, storage, presentation and verification of credentials which may be used anywhere in the Hyperledger ecosystem. At it's core, the AriesExchanger allows actors in an information flow (Sender or recipient) to verify attributes about the other based on attestations made by trusted authorities, determined by the context. Aries agents facilitate the zero-knowledge presentation of group membership as described by Camenisch & Lysyanskaya (2001). The Aries agent interface is accessed through the AriesExchanger class. From a Syft perspective, AriesExchanger allows Data Owners to only initiate connections with Data Scientists who have credentials to work with their data as described in (Abramson et al. (2020)). Similarly, Data Scientists can verify whether remote datasets held by Data Owners comply with certain scheme requirements as attested to by the appropriate authority. Governance systems are defined and then implemented through the definition of credential schemes on a distributed ledger. This infrastructure allows for governance systems to be constructed and enforced without the need for a central governing authority. All trust verification is performed peer-to-peer using the Peer DID Communications (PeerDIDComm) protocol (Will Abramson, Adam Hall, Lohan Spies, Tom Titcombe (2021)).

PyDP - The application of statistical is a Python wrapper for Google's Differential Privacy project. The library provides a set of ϵ-differentially private algorithms. These can be used to produce aggregate statistics over numeric data sets containing private or sensitive information. With PyDP you can control the privacy guarantee and accuracy of your model written in Python. PyDP is being extended to provide DP training of conventional data science algorithms such as bayesian networks and decision trees with scikit-learn.

Syfertext - is a library which provides secure plaintext pre-processing and secure pipeline deployment for natural language processing in Syft(Aboudib (2021)).

openmined_psi - a Private Set Intersection protocol based on ECDH, Bloom Filters, and Golomb Compressed Sets as described in Angelou et al. (2020).

A.5 TABLES

Name	Date	Org	Base	PSI	VFL	HFL	DP	HE	SMPC	ZkAC	OPC
PySyft	Jul17	OpnMnd	TH,+	3rd	Y	Y	3rd	3rd	Y	3rd	Y
TFF	Sep18	Google	Any	N	N	Y	3rd	N	3rd	N	N
FATE	Sep18	WeBank	TH,+	N	N	Y	3rd	N	Y	N	N
LEAF	Dec18	CMU	TF	N	N	Y	3rd	N	N	N	N
eggroll	Jul19	WeBank	TF,+	N	N	Y	3rd	N	N	N	N
PaddleFL	Sep19	Baidu	PD	Y	Y	Y	Y	N	Y	N	N
FLSim	Nov19	iQua	TH	N	N	Y	N	N	N	N	N
DT-FL	Nov19	BIL Lab	TH	N	N	Y	N	N	N	Y	N
Clara	Dec19	NVIDIA	TF	N	N	Y	Y	N	N	N	N
DP&FL	Feb20	SherpaAI	TF,SKL	N	N	Y	Y	N	N	N	N
IBMFL	Jun20	IBM	KS+	N	N	Y	Y	N	N	N	N
FLeet	Jun20	EPFL	TF?	N	?	?	Y	N	N	N	N
IFed	Jun20	WuhanU	CS	N	?	?	Y	N	N	N	N
FedML	Jul20	FedML	TH	Y	Y	N	N	N	N	N	N
Flower	Jul20	Cmbrdge	Any	N	N	N	Y	N	N	N	N

Table 1: Federated Learning Systems Feature Support Comparison. We extend the work of Trask & Prakash (2020) to compare PET feature support in existing federated learning architectures. Org: Organisation associated with tool. Base: core machine learning technology. PSI: does the framework provide an api for PSI computation. VFL: does the framework provide an api for some form of Vertically Federated Machine Learning. HFL: does the framework provide an api for some form of Horizontally Federated Machine Learning. DP: does the framework provide an api for some form of Differential Privacy. HE: does the framework provide an api for some form of Homomorphic Encryption. SMPC: does the framework provide an api for some form of Secure Multiparty Computation. ZkAC: does the framework provide an api for some form of Zero-knowledge Access Control. OPC: Does the framework provide an object level RPC. Refs: TFF - Ingerman & Ostrowski (2019)), FATE - Wei (2018), LEAF - Caldas et al. (2018)), eggroll - Max Wong, H Moster, and Lin, Bryce (2018), PaddleFL - Qinghe Jing, Dong Daxiang, and contributors (2019), FLSim - Wang et al. (2020), DT-FL - Abramson et al. (2020). Clara - Nvidia (2019), DP%FL - Sherpa (2019), IBMFL - Ludwig et al. (2020)), DT-FL - Abramson et al. (2020), FLeet - Damaskinos et al. (2020), IFed - Hui et al. (2020), FedML - He et al. (2020), Flower - Beutel et al. (2020)

Forward Step	Processor	Modulus (bits)	File Size	Time Taken
Input Data	DO	plaintext	3.47KB	
Conv1	DO	plaintext	10.97 KB	
Conv2	DO	plaintext	2.4 KB	27ms
Encrypt Signal	DO	140	269.86 KB	11 ms
FC1	DS	140	205.06 KB	
Sq. Activation	DS	140	139.6 KB	
FC2	DS	140	68.81 KB	4.17s

Table 2: Experiment 1, where the DO only receives the convolutional layers. The split layer at FC2 is represented by the bar. As more layers need to be processed, a higher modulus is used. Tests were performed using 4 cores Intel(R) Core(TM) i7-6600U CPU @ 2.60GHz. Forward Step describes the layer of processing the data has just emerged from. The Processor is the actor currently performing the computation. File Size is the size of the signal as it emerged from this layer of processing. Time taken gives the time that has passed since the last time taken. HE parameters; polynomial degree is 8192, coefficient modulus is 140-bits, there is a security level of 128-bits and the scale is 2^{26}

.

Forward Step	Processor	Modulus (bits)	File Size	Time Taken
Input Data	DO	plaintext	3.47KB	
Conv1	DO	plaintext	10.97 KB	
Conv2	DO	plaintext	2.4 KB	
FC1	DO	plaintext	529 B	
Sq. Activation	DO	plaintext	529 B	1ms
Encrypt Signal	DO	88	139.62 KB	4.6 ms
FC2	DS	88	68.76 KB	97ms

Table 3: Experiment 2, where the FC1 layer is also sent to the DO. The split layer is represented by the bar. As less layers need to be processed, a lower modulus is used. Tests were performed using 4 cores Intel(R) Core(TM) i7-6600U CPU @ 2.60GHz. Forward Step describes the layer of processing the data has just emerged from. The Processor is the actor currently performing the computation. File Size is the size of the signal as it emerged from this layer of processing. Time taken gives the time that has passed since the last time taken. CKKS parameters; polynomial degree is 8192, coefficient modulus is 88-bits, there is a security level of 128-bits and the scale is 2^{26}

.