PRIVACY-PRESERVING PHISHING WEB PAGE CLASSIFICATION VIA FULLY HOMOMORPHIC ENCRYPTION

Edward J. Chou[^, *], *Arun Gururajan*[±], *Kim Laine*[†], *Nitin Kumar Goel*[±], *Anna Bertiger*[±], *Jack W. Stokes*[†]

[^] Stanford University, Palo Alto, CA 94305 USA
[±] Microsoft Corp., One Microsoft Way, Redmond, WA 98052 USA
[†] Microsoft Research, One Microsoft Way, Redmond, WA 98052 USA

ABSTRACT

This work introduces a fast and lightweight homomorphic-encryption pipeline that enables privacy-preserving machine learning for phishing web page recognition. The primary goals are to use visual features to train an accurate model and to implement an inference pipeline with practical runtime and communication costs. To do so, we deploy a variety of techniques that cover deep learning and optical character recognition to extract salient visual features, and optimize the inner mechanisms of state-of-the-art homomorphic encryption schemes to reduce the encryption-related costs. Our presented system is able to achieve over 90% on the visual classification task, while using less than 250 KB of communication bandwidth and around 0.7 seconds of computation time. We hope our work not only demonstrates a private visual phishing detection pipeline, but also outlines techniques to practically utilize homomorphic encryption in a variety of machine learning tasks.

Index Terms— Phishing, Classification, Homomorphic Encryption

1. INTRODUCTION

Phishing attacks attempt to fool users into revealing sensitive data and personal information, and an especially prevalent technique is to create a fake web page that visually imitates an authentic login page and harvests credentials from the unsuspecting user [1]. Leveraging the visual attributes of web pages using deep learning-based computer vision techniques could provide valuable signals to detect and screen phishing attacks, greatly enhancing existing tools which currently use only URL information and other simple metadata [2]. However, there are serious privacy concerns involved in displaying the visual screenshot of a user's internet activity to third-party services, which may include snapshots of banking information, email correspondences, corporate trade secrets, etc. This provides a strong motivation to create a privacy-preserving approach that allows for cloud computation while keeping the end user's data secure. A technique highly relevant to this use case is fully homomorphic encryption (HE), a specialized form of cryptography designed for private computation [3].

HE was first developed by Gentry et al., [3] and encompasses techniques, which preserve homomorphisms when performing operations on encrypted data. At a high level, this means that encrypted data can be sent over the internet to a third party, who can perform arbitrary computational steps without ever having access to the raw user data, thus protecting the privacy of the user. The most efficient fully HE schemes support additions and multiplications, which enable the computation of dot products and hence, naturally lend themselves to neural-network models [4]. The caveat, however, is that HE is known for its high computation and communication costs, and specialized schemes need to be developed for practical deployment of HE in real-world applications.

In this work, we demonstrate a privacy-preserving system that leverages visual features of a web page to identify the brand that was phished by a doppelganger page. The proposed approach introduces a new method of producing a compact yet salient feature vector for training a machine learning model, and in addition, uses a variety of novel techniques to speed up homomorphic encryption run-times and communication costs. For this paper, Microsoft's open source SEAL library [5] is used to perform HE. This work is one of the first demonstrations of a practical use-case of homomorphic encryption in a highly relevant, real-world scenario. The following sections will detail the methods used for both the machine learning and homomorphic encryption aspects of the aforementioned pipeline, and discuss the experimental results that indicate the efficacy of the proposed approach.

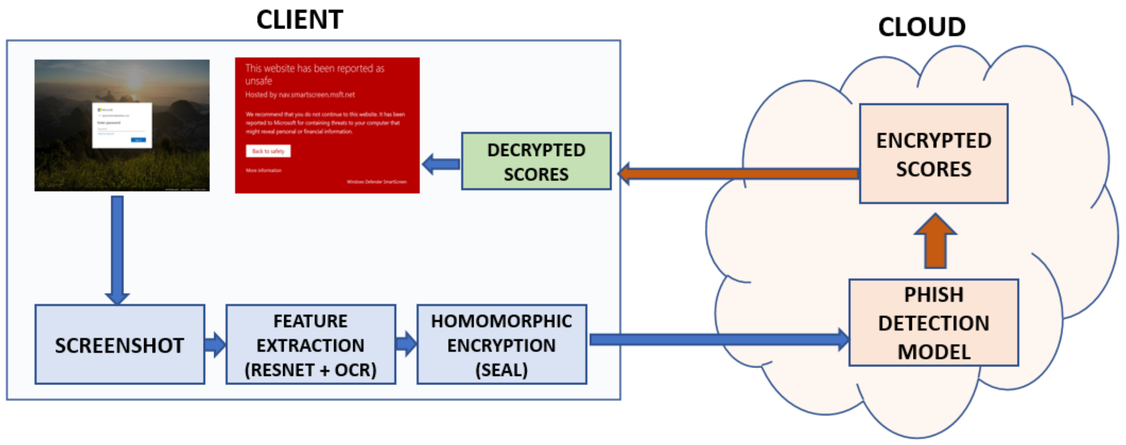
2. METHODS

Task Setting. Before presenting our methods, we first cover the task setting under consideration. In our envisioned use case, a client's browser efficiently queries a cloud phishing-detection service without revealing its plain input data or output results. The client machine (end-user's local machine) collects screenshots of web pages and/or emails the user is accessing, and collects visual features of the page. Following this, the client encodes the computed features and transmits the encrypted data to the cloud. In this setup, we implicitly assume the client has the computing power to perform preprocessing on the image as well as the encryption/decryption tasks.

In the cloud, the proprietary model performs inference on the encrypted features, batches the encrypted probabilities together and sends this back to the client, where the secret key is used to decrypt the probabilities and a decision engine is used to identify the type of phishing web page (e.g., Facebook, Outlook). If the system detects that it is a phishing page, a warning message is shown to the end user. The end-to-end workflow for this process is clearly illustrated in Figure 1.

It should be noted that in this work, we are considering visual signals for brand recognition and not other usable metadata such as URLs and IP addresses. Essentially, our machine learning task focuses on analyzing visual similarity for brand recognition rather than direct phishing classification, to avoid misclassifying malicious web sites that are visually indistinguishable from a legitimate web page.

[*]The first author performed the work while at Microsoft Research.


CLIENT
CLOUD
DECRYPTED SCORES
ENCRYPTED SCORES
PHISH DETECTION MODEL
SCREENSHOT
FEATURE EXTRACTION (RESNET + OCR)
HOMOMORPHIC ENCRYPTION (SEAL)
This website has been reported as unsafe
Hosted by nav.smartscreen.msft.net
Back to safety
Windows Defender SmartScreen

..

Fig. 1. End-to-end Pipeline of the Proposed Phish Brand Recognition system

Methods: Machine Learning. As mentioned earlier, the first step is to extract relevant features from the web page screenshot that comprise the visual information in the image. The following steps are used for featurization:

1. The first set of features are obtained by evaluating the image which is a model ResNet-52 model [6] where the final output layer has been removed. This model includes a series of convolutional layers that have been pretrained on the ImageNet dataset [7]. The features are essentially the outputs of the penultimate layer of the network.
2. In addition, a second set of features are computed by evaluating the image with an optical character recognition (OCR) engine to extract text in the web page. This text is then encoded using a TF-IDF featurizer [8].

The reasons for this choice of featurizers are as follows. The convolutional layers in the ResNet are intended to capture visual elements (look and feel) of the page. However, using a model that is pretrained on the ImageNet data entails downsizing the web page screenshot, originally of dimensions 1000 x 750, to a much smaller size of 224 x 224. The effect of this downsampling results in severe degradation of textual information in the image. This loss is counteracted by the second set of features that extract OCR text from the original image.

Subsequently, these features are concatenated, and then encrypted (as described in the following section) prior to transmission to the cloud. We implement a logistic regression classifier using HE in the cloud that performs inference on the encrypted features. It should be noted that the linear model is trained on an unencrypted feature set - only the inference happens on encrypted feature vectors. The choice of a homomorphically encrypted linear model is due to the following reasons:

- Since the web page's image is first processed by a ResNet-52 model on the client, we found that including additional encrypted dense fully connected layers did not improve the overall detection performance and increases the noise budget as well as the inference time on the encrypted features.
- The entire phishing detection model is intended to be operationalized for real-time phishing protection, and in such a setting, latency is of consequence as it impacts the end-user experience. Thus, a linear model, by its formulation, enables rapid inference on the generated features.

Methods: Fully Homomorphic Encryption. There are numerous methods to enable privacy-preserving oblivious inference, of which we elect to use HE as our technical basis for our techniques. Secure Multiparty Computation (SMC) is a separate encrypted computation paradigm that involves two parties performing joint computation without ever leaking information [9]. SMC's primary drawback is related to its prohibitively high communication cost, a severe limitation in our cloud-based task setting. Secure hardware enclaves [10] require the use of specialized hardware, which are not widely available on our cloud servers. In addition, this approach is also not based on cryptographic primitives and are not as theoretically secure.

That said, HE is still known for its high computational costs [11], and has thus seen limited adoption in practical use cases. In this section, we cover techniques that can be used to reduce both the runtime and the communication cost of the HE-based privacy-preserving pipeline. These techniques work specifically on linear models under the assumption of single examples rather than batched, where users send over individual inputs and expect a prompt output rather than an intermediary sending over a collection of inputs and the server performing a batched computation.

Currently, Microsoft's SEAL library implements two popular HE schemes, namely BFV [12] and CKKS [13]. In the following section, we conduct experiments with both schemes and compare their performance.

We start with the most naïve approach we can take to implement homomorphic encryption for linear models. If we have an input vector of dimension x and y number of possible output classes, we know our weight vector will have dimensionality of $(x$,x,$y)$ and the bias to have dimension y. The computation involves performing a dot product on the input vector and the weights and adding the output to the bias vector. We can encode each individual value within each of these vectors as a ciphertext, and perform each individual operation using the widely used HE scheme BFV. Because BFV only supports integer encoding and does not provide rescaling, we also perform a transformation to transform our vector's floating point values to integers using a fixed-point conversion, and adjust our parameters to account for the scale growth with each subsequent operation layer.

The first technique we focus on is batching, a powerful encoding technique that can greatly reduce the amortized computation time of encryption. The plaintext encodings used by HE schemes encode the data in "slots" equal to the number of moduli, a parameter also used to adjust the security level of the encryption. A convenient

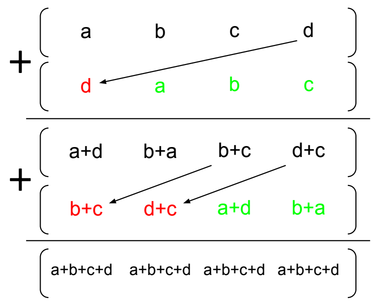


Fig. 2. Visualization of rotation/summation technique

property of these slots is that they can each hold a value, allowing multiple values to be packed into one ciphertext and be computed on in parallel without an increase in computational time or memory. This technique has been used in the past to allow computation on a batch of data, but we use this technique to parallelize the dot product and addition operations in our linear model. We can pack our entire input into one vector and our weight and bias features into y vectors, performing a dot product with the input plaintext and each of our weight ciphertexts in a parallelized fashion.

Before we do so, we need to leverage another technique, summation through rotation, to complete our dot product operation. Without batching, we simply multiplied our arrays element-wise, then added each of the products to perform the summation. With batching, we are able to parallelize the multiplication step, but performing summation on all the elements packed inside the ciphertext is not as straightforward. The server cannot decrypt the single ciphertext, which means it cannot access the values to perform the summation, and needs to find a way to add up all the values given only the ciphertext. A useful technique in this case is ciphertext rotation, where by creating special rotation keys known as Galois keys [14], we are able to rotate a ciphertext's slots in place. This allows us to create a copy of our ciphertext and rotate it element by element, adding up each of the values to obtain the summed value in our first slot. This approach can be improved further by rotating in increments of powers of 2 (e.g. rotate 1, 2,..$2^{(n/2)}$) while adding our ciphertexts. As we can see in Figure 2, by our last rotation/summation, each slot in our ciphertext will contain the summed value of our ciphertext.

The approach above solves the summation problem, but there is a noticeable inefficiency in the dot product output. There are now multiple ciphertexts with slots containing the same value, and we will need a ciphertext for each bias value to add to our dot product outputs. Furthermore, assuming the model has n outputs, the client will receive n ciphertexts as output after sending over just one ciphertext. Collapsing these output ciphertexts into one ciphertext will both reduce the output size and also perform the bias addition in parallel. As seen in Figure 3, by constructing bitmasks to strategically zero out slots of each ciphertext, and then by adding these ciphertexts together, we can get a final ciphertext where each slot holds a value corresponding to its output position.

Our next optimizations involve the use of a different encryption scheme called CKKS, which has several distinct advantages over BFV. CKKS supports built-in floating point encoding, although it does so by sacrificing the number of available slots by half. The main advantage of CKKS is that it enables "rescaling", a technique which allows us to reduce the parameters of CKKS with some precision loss of our floating point values. The parameter reduction speeds up computation times and also reduces the size of the final ciphertext.

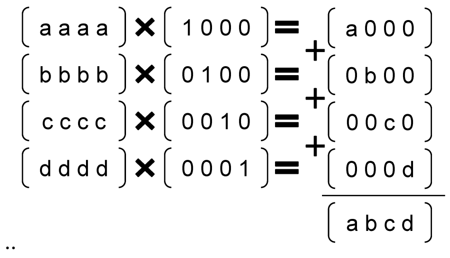


Fig. 3. Visualization of masked summation

Furthermore, by carefully selecting our initial parameters (polynomial moduli), we can increase the number of empty segments in our ciphertext consisting of null values, which allows us to reduce the ciphertext even more by using a compression library.

Method	Encryption Time	Computation Time
Plain	N/A	0.0014 sec
BFV (no batching)	13.1326 sec	211.229 sec
BFV (with batching)	0.313412 sec	1.18373 sec
CKKS	0.183926 sec	1.14705 sec
CKKS + rescale	0.008562 sec	0.689715 sec

Table 1. Homomorphic Encryption Results - Timing

3. EXPERIMENTS

Dataset. Our dataset consists of web site screenshots collected from Microsoft's in-house URL detonation tool, accompanied by manually-generated labels indicating the brand that was being phished. This dataset comprises 18,854 malicious screenshots, as well as an additional 1000 screenshots of legitimate web pages. The classes represented within our dataset consist of 20 commonly targeted companies, including social media web sites like Facebook and LinkedIn, banking web sites like Bank of America and American Express, and enterprise pages like Outlook and Adobe.

Results: Machine Learning. In Section 2, we describe how to construct a feature vector by using a deep CNN as a feature extractor and an OCR algorithm. To convert the text from OCR to a mathematical vector, we use TF-IDF which weighs rare keywords like brand names heavily. Specifically, we use an Imagenet-pretrained ResNet-52 model with the final fully connected layer removed and the open sourced Tesseract software [15], each of which create a 2048 x 1 vector, which we subsequently concatenate to create a 4096 x 1 feature vector. By using open-source algorithms like ResNet-52 and Tesseract, the user is able to construct features on their local computer without the server having to send over proprietary algorithms or models to the user, which ensures training data privacy.

As we can see from accuracy results presented in Table 3, solely using ResNet features or OCR features can only produce a model with mediocre accuracy, while combining the features together allows our model to attain 90.57% accuracy. This is a good indicator that the two feature vector sources encode different aspects of the image, with the OCR-based features capturing the textual data and the CNN-based features capturing the visual feel of the web page.

Results: Homomorphic Encryption. The homomorphic encryption timing results are summarized in Tables 1. Our final trained model consists of 40,960 multiplicative operations and 4,106 additive operations and takes 0.0014 seconds to run in a normal setting.

Method	Input Size	Output Size	Bit Precision
Plain	32.768KB	0.168KB	N/A
BFV (no batching)	1.61GB	8.26MB	4th digit
BFV (with batching)	393.29KB	8.26MB	4th digit
CKKS	393.29KB	8.26MB	No Loss (7th digit)
CKKS + masking	393.29KB	393.29KB	No Loss (7th digit)
CKKS + masking/rescale	393.29KB	131.145KB	6th digit
CKKS + masking/rescale + compression	329.628KB	130.096KB	6th digit
CKKS + masking/rescale + new params for compression	246.825KB	88.94KB	4th digit

Table 2. Homomorphic Encryption Results - Memory and Precision

Features	Accuracy
Resnet Features	61.1%
OCR	71.25%
Resnet Features + OCR	90.57%

Table 3. Phishing Classifier Accuracy

We can easily observe that our first naïve approach requires a prohibitively large amount of encryption and computation time, taking over 3 minutes to perform both steps. With batching, we are able to achieve a massive speedup, reducing our computational time to a little over a second. Switching to the CKKS scheme results in a slight amount of further speedup. Utilizing the rescale operations in CKKS cuts our final encryption time to milliseconds and the computation time to around 0.7 seconds, which is below the reasonable amount of time the user would need to enter their personal information like their passwords, preventing data theft.

We see similar benefits in Table 2 in memory consumption by using the proposed optimization schemes. As a baseline, our plain inputs and outputs are around 30KB and 0.2 KBs, respectively. With our naïve approach, we have a prohibitively high input size of 1.61 GB and output size of 8.26MB. With batching, we are able to cut down our input size significantly to around 400KB; however, we still have the same number of ciphertext outputs. We need to use the masking trick to cut down our output to the same size as our input. We can see that by switching to CKKS and using rescaling, we can further cut down our output size by almost a third. Finally, we can see that though non-optimal parameter selection does not result in much memory reduction through compression, careful selection of our parameters allows us to cut down our input and output sizes further by around 30% to 250 KB and 90KB, respectively, which, in turn, are much more practical for wireless transfers.

We also analyze the effects of our optimizations on the precision of our computations in Table 2. The integer encoding we perform to use BFV results in a loss of precision to about the 4^{th} digit, which is high enough to perform analysis like softmax to obtain reliable results. Switching to CKKS immediately results in much higher precision due to the support of floating point encoding. By performing rescaling, we drop our precision to around the 6^{th} digit, and parameter reselection drops our precision to about the 4^{th} digit once more.

Summary. Our experiments clearly show the benefits of our model selection and homomorphic encryption optimizations. Our final pipeline results in model accuracies of over 90%, computational times under a second, and communication costs significantly below a MB. This is achieved without any loss of client or server privacy through our use of open-sourced models and software.

4. RELATED WORK

Many techniques have been developed using both hardware and software to protect the privacy of computational data. Several of these methods such as HE are designed to enable *oblivious computation*, a paradigm in which the party computing on a data item is unaware of the contents or the output of the computation [16]. SMC is another cryptographic technique, which allows multiple parties to jointly compute on data while keeping their own subsets private [9]. Secure hardware enclaves are a hardware-based approach towards oblivious computation, which utilizes practices related to physical security and side channel mitigation rather than cryptographic primitives to preserve privacy [10].

A different group of algorithms related to differential privacy are designed to address a different notion of privacy that aims to limit privacy leakage when data is accessed [17] [18] [19] [20] [21]. These techniques often involve transforming the dataset to normalize the statistics of the general population in some way, whether through removing outliers or adding random noise to a dataset [22].

Phishing defenses have long used the metadata of web sites such as URL information and page-content to detect malicious sites [2]. One work detects attacks that substitute parts of the URL with visually similar characters (e.g., 0 for O) meant to fool users [23]. Other works [24] have used visual aspects of the web page (color distribution, logos etc.) for phishing detection.

5. CONCLUSION

Phishing attacks continue to be a major cybersecurity problem as attacks become more elaborate and targeted. The visual appearance of a web page can hold a great deal of information for phishing detection and defense, but revealing screenshots to third party services represents a severe privacy risk. By carefully selecting how we extract signals from visual features and utilizing a multitude of homomorphic encryption techniques, we can create a practical pipeline that allows cloud services to provide visual phishing classification without any privacy loss.

Further improvements can be made to improve both our visual signals for machine learning and homomorphic encryption costs. Our feature vectors can be enhanced to capture other aspects of our web page, which could involve using object detection to segment logos from web sites [25] [26]. Encrypted computation time can also be accelerated through specialized hardware such as FPGAs [27]. We hope that this work demonstrates a practical privacy-safe pipeline for not only phishing detection, but also other machine learning tasks used in privacy-sensitive environments.

Acknowledgements. The authors would like to acknowledge Christian Seifert for participating in discussions and providing feedback.

6. REFERENCES

[1] Sanchari Das, Andrew Kim, Zachary Tingle, and Christena Nippert-Eng, "All about phishing: Exploring user research through a systematic literature review," 2019.

[2] Biju Issac, R. Chiong, and Seibu Mary Jacob, "Analysis of phishing attacks and countermeasures," *CoRR*, vol. abs/1410.4672, 2014.

[3] Craig Gentry, "Fully homomorphic encryption using ideal lattices," in *Proceedings of the Forty-first Annual ACM Symposium on Theory of Computing*, New York, NY, USA, 2009, STOC '09, pp. 169–178, ACM.

[4] Ran Gilad-Bachrach, Nathan Dowlin, Kim Laine, Kristin Lauter, Michael Naehrig, and John Wernsing, "Cryptonets: Applying neural networks to encrypted data with high throughput and accuracy," 2016.

[5] "Microsoft SEAL (release 3.3)," `https://github.com/Microsoft/SEAL`, 2019, Microsoft Research, Redmond, WA.

[6] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun, "Deep residual learning for image recognition," 2016.

[7] Olga Russakovsky, Jia Deng, Hao Su, Jonathan Krause, Sanjeev Satheesh, Sean Ma, Zhiheng Huang, Andrej Karpathy, Aditya Khosla, Michael Bernstein, et al., "Imagenet large scale visual recognition challenge," *International Journal of Computer Vision*, 2015.

[8] Joeran Beel, Bela Gipp, Stefan Langer, and Corinna Breitinger, "Research-paper recommender systems : a literature survey," *International Journal on Digital Libraries*, vol. 17, no. 4, pp. 305–338, 2016.

[9] Vladimir Kolesnikov and Thomas Schneider, "Improved garbled circuit: Free xor gates and applications," in *International Colloquium on Automata, Languages, and Programming*. Springer, 2008.

[10] Florian Tramèr and Dan Boneh, "Slalom: Fast, verifiable and private execution of neural networks in trusted hardware," 2018.

[11] Ehsan Hesamifard, Hassan Takabi, and Mehdi Ghasemi, "Cryptodl: Deep neural networks over encrypted data," *arXiv*, 2017.

[12] Shai Halevi, Yuriy Polyakov, and Victor Shoup, "An improved rns variant of the bfv homomorphic encryption scheme," in *Topics in Cryptology – CT-RSA 2019*, Mitsuru Matsui, Ed., Cham, 2019, pp. 83–105, Springer International Publishing.

[13] Jung Cheon, Andrey Kim, Miran Kim, and Yongsoo Song, "Homomorphic encryption for arithmetic of approximate numbers," 11 2017, pp. 409–437.

[14] David A. McGrew and John Viega, "The security and performance of the galois/counter mode (gcm) of operation," in *Progress in Cryptology - INDOCRYPT 2004*, Anne Canteaut and Kapaleeswaran Viswanathan, Eds., Berlin, Heidelberg, 2005, pp. 343–355, Springer Berlin Heidelberg.

[15] Google, "tesseract-ocr," 2008.

[16] Olga Ohrimenko, Felix Schuster, Cédric Fournet, Aastha Mehta, Sebastian Nowozin, Kapil Vaswani, and Manuel Costa, "Oblivious multi-party machine learning on trusted processors.," in *USENIX Security Symposium*, 2016.

[17] Martín Abadi, Úlfar Erlingsson, Ian Goodfellow, H. Brendan McMahan, Nicolas Papernot, Ilya Mironov, Kunal Talwar, and Li Zhang, "On the protection of private information in machine learning systems: Two recent approaches," in *IEEE 30th Computer Security Foundations Symposium (CSF)*, 2017, pp. 1–6.

[18] Reza Shokri, Marco Stronati, and Vitaly Shmatikov, "Membership inference attacks against machine learning models," *CoRR*, vol. abs/1610.05820, 2016.

[19] Matt Fredrikson, Somesh Jha, and Thomas Ristenpart, "Model inversion attacks that exploit confidence information and basic countermeasures," in *Proceedings of the 22Nd ACM SIGSAC Conference on Computer and Communications Security*, New York, NY, USA, 2015, CCS '15, pp. 1322–1333, ACM.

[20] Martin Abadi, Andy Chu, Ian Goodfellow, H. Brendan McMahan, Ilya Mironov, Kunal Talwar, and Li Zhang, "Deep learning with differential privacy," pp. 308–318, 2016.

[21] Nicolas Papernot, Shuang Song, Ilya Mironov, Ananth Raghunathan, Kunal Talwar, and Úlfar Erlingsson, "Scalable private learning with pate," *arXiv preprint arXiv:1802.08908*, 2018.

[22] Cynthia Dwork, "Differential privacy: A survey of results," in *International Conference on Theory and Applications of Models of Computation*. Springer, 2008.

[23] Doyen Sahoo, Chenghao Liu, and Steven C. H. Hoi, "Malicious URL detection using machine learning: A survey," *CoRR*, vol. abs/1701.07179, 2017.

[24] Igino Corona, Battista Biggio, Matteo Contini, Luca Piras, Roberto Corda, Mauro Mereu, Guido Mureddu, Davide Ariu, and Fabio Roli, "Deltaphish: Detecting phishing webpages in compromised websites," *CoRR*, vol. abs/1707.00317, 2017.

[25] Joseph Redmon, Santosh Kumar Divvala, Ross B. Girshick, and Ali Farhadi, "You only look once: Unified, real-time object detection," *CoRR*, vol. abs/1506.02640, 2015.

[26] Shaoqing Ren, Kaiming He, Ross B. Girshick, and Jian Sun, "Faster R-CNN: towards real-time object detection with region proposal networks," *CoRR*, vol. abs/1506.01497, 2015.

[27] Erdinç Öztürk, Yarkin Doröz, Berk Sunar, and Erkay Savas, "Accelerating somewhat homomorphic evaluation using fpgas.," *Cryptology ePrint Archive*, 2015.