

Hard to See, Harder to Block: Exploring the Challenges in Email Tracking Pixels Detection

FEDERICO CERNERA¹, VINCENZO IMPERATI¹, MASSIMO LA MORGIA¹, ALESSANDRO MEI¹, ALBERTO MARIA MONGARDINI¹, EUGENIO NERIO NEMMI¹, LUCIANO ORTENZI², FRANCESCO SASSI¹

¹Computer Science Department, Sapienza University of Rome, Italy (e-mail: surname@di.uniroma1.it)

²Department of Agriculture and Forest Science, Tuscia University, Italy (e-mail: luciano.ortenzi@unitus.it)

ABSTRACT Email tracking pixels are a well-known mechanism for monitoring user interactions, yet reliably detecting and blocking them remains challenging in practice. These pixels can be implemented in many subtle ways, and our analysis shows that detection outcomes vary significantly depending on the heuristics applied. Using a dataset of more than 44,000 real-world emails, we observe that the proportion of emails flagged as containing tracking pixels varies heavily depending on the method used. In evaluating 22 popular commercial tracking services, we find that small changes, such as modifying image dimensions, altering URL paths, or obfuscating identifiers, are often sufficient to evade detection. We also compare Proton Mail's private blocklist with leading public blocklists and find that each identifies URLs missed by the other, indicating their complementary strengths. Our results show that, despite years of attention, tracking pixels remain difficult to detect consistently because their implementations can be easily adapted to bypass existing techniques, and no single approach provides complete coverage.

I. INTRODUCTION

In modern digital ecosystems, user data has become a critical asset driving online services, targeted advertising [1], and personalized user experiences [2], [3]. One popular mechanism enabling such data collection is the tracking pixel. Embedded within digital content such as emails, tracking pixels function as invisible tracking tools. When an email client renders the image, it sends a request to a server controlled by the sender or a third-party service, logging when and where an email was opened, on what type of device, and often the IP address, which can be used to approximate the recipient's location. Although tracking pixels support legitimate purposes such as analytics, campaign measurement, and content optimization, the hidden nature of this data collection raises significant privacy concerns, as it facilitates the construction of detailed behavioral profiles without user awareness or explicit consent [4].

The privacy implications of email tracking mechanisms have led to research efforts focused on their detection. Existing academic detection methods often rely on a set of heuristics that attempt to identify tracking pixels by analyzing attributes such as image dimensions (*e.g.*, 1x1 pixels), and URL structure [4]–[6]. However, these heuristics rely

on assumptions about how trackers are implemented that do not always hold in practice. Indeed, simple countermeasures, such as obfuscating URL identifiers or embedding tracking functionalities within larger, benign-looking images, are sufficient to evade these detection systems. In this paper, we show that, despite increasing attention from academia, industry, and regulators, accurately detecting tracking pixels remains a challenging task. First, we provide a comprehensive overview of state-of-the-art detection methods. Then, we build a novel dataset of more than 44,000 emails collected over four years and apply the detection methods to evaluate their limitations. Our findings reveal a significant variance in the detection outcomes. Depending on the chosen heuristic, the estimated proportion of emails containing tracking pixels fluctuates from 10% to 85%. Next, we examine the implementation strategies used by popular tracking services and evaluate how effectively existing detection models identify the tracking pixels they deploy. Our results confirm that the evasion strategies purposefully designed by tracking services bypass existing detection techniques.

Then, we study the ability of popular email clients to detect and block tracking pixels. Our analysis shows that most mainstream web and desktop clients offer no meaningful protec-

tion against tracking pixels unless image loading is disabled entirely, a measure that severely degrades user experience. Although Apple’s ecosystem provides limited blocking, only privacy-focused clients like Proton Mail demonstrate robust protection, primarily due to curated lists of known tracking services. However, Proton’s solution is not easily compatible with non-Proton email accounts, and its effectiveness depends on a proprietary, undisclosed blocklist. This motivates our final research direction: an evaluation of publicly available blocklists. Although promising, we find that these public lists also offer an incomplete solution to the problem.

Our contributions are as follows:

- **Empirical evaluation of email tracking pixel detection:** This paper provides novel insights into the challenges of email tracking pixel detection through the analysis of a large real-world email dataset. We replicate and analyze state-of-the-art detection methods on a novel dataset of more than 44,000 real-world emails collected over four years. Our analysis reveals key limitations and inconsistencies of existing approaches, offering important insights for researchers and practitioners.
- **Analysis of Commercial Tracking Pixel Providers:** We analyze how popular third-party services implement tracking pixels and use the emails they generate as ground-truth to evaluate detection techniques reported in the scientific literature. Our findings show that even minimal obfuscation is sufficient to evade detection. Additionally, we assess the protection capabilities of popular email clients and find that most fail to block tracking pixels unless image loading is disabled. Only Proton Mail offers reliable protection, primarily through curated blocklists.
- **Evaluation of blocklist-based approaches:** We assess the extent to which Proton Mail’s tracking pixel detection can be approximated using publicly available blocklists. The results indicate that while blocklists can identify many tracking pixels detected by Proton Mail, they also flag a significant number of pixels missed by Proton. This finding further corroborates that despite ongoing efforts from academic, commercial, and open-source platforms, a complete and reliable defense against email tracking remains an open research problem.

II. BACKGROUND

A. TRACKING PIXELS

In this section, we provide a high-level overview of how tracking pixels typically operate through the lens of a standard email tracking system.

The sender hosts an image on a server they control, with the intent of using it for tracking. To ensure that the image remains unobtrusive to the recipient, it is typically a very small transparent image, so that it is effectively invisible when the email is shown to the user. However, this is not strictly required for tracking to work.

To track email openings, the sender embeds the server-hosted image into the email’s HTML body using an `img` tag, whose `src` attribute points to the tracking image. The image is fetched via an HTTPS request when the email is opened. Then, the sender transmits the email to the target user through its Mail Transfer Agent (MTA, *e.g.*, Postfix, Exim, and Sendmail). Optionally, the sender may incorporate a unique identifier into the URL to associate image requests with specific recipients. This can be achieved in multiple ways, such as placing the identifier directly in the URL path or including it as a query parameter. The identifier can also be implemented as a random string, a user ID, the recipient’s email address, or a hashed version of the email address. Alternatively, the identifier may refer to a marketing campaign instead of a specific user. In this case, a group of users receives emails embedding the same campaign ID, allowing the sender to measure the effectiveness of the campaign without tracking individual recipients.

When the recipient receives and opens the email, the email client automatically fetches the embedded image, triggering an HTTP request to the specified URL. This request occurs without any explicit action by the user. The server hosting the image logs the request, enabling the sender to infer that the email was opened. If no identifier is included in the URL, the server can still log metadata such as the time of access, the IP address (which may be used to infer geolocation), and the user-agent string (which reveals device and browser details). This information can be used to fingerprint users with high accuracy [7], [8]. However, if an identifier is present, the request can be directly linked to a specific recipient. Indeed, email tracking functionality may be implemented not only by the sender but also by third-party services. These services handle image hosting, track recipient interactions, and provide analytics to the sender without requiring them to operate their own tracking infrastructure.

B. ANTI-TRACKING TOOLS

The most common anti-tracking solutions are browser extensions that analyze the content of incoming emails [9]. These extensions typically operate by parsing the email content rendered in the browser, extracting all `` tags, and inspecting their `src` attributes. Although the only guaranteed method of eliminating email tracking is to block all externally hosted resources, this approach substantially degrades usability by removing legitimate images and layout elements from the message [10]. As a result, browser extensions adopt a more selective strategy, relying on pattern matching to block content that is likely to be a tracking pixel. To implement the pattern matching, the extensions rely on lists of domains, URL patterns, or query parameters that are linked to tracking activities [9]. Notable examples include EasyPrivacy [11], EasyList [12], the Ugly Email Trackers list [13] used by the Ugly Email extension [14], the tracking filter list maintained by uBlock Origin [15], [16], and the StevenBlack hosts file [17], which aggregates multiple blocklists and has received significant community attention. However, the effec-

tiveness of this rule-based approach is tightly coupled to the completeness and accuracy of the underlying lists. Trackers that employ obfuscation techniques or serve tracking content from their own domain (*i.e.*, first-party domain) instead of relying on third-party services can often bypass these filters [18].

III. RELATED WORK

Users are tracked on the web, mostly for marketing and targeted advertising [19], [20]. Krishnamurthy and Wills were among the first to study the phenomenon [21], showing a continual increase in third-party tracking over the years [22], a trend confirmed by later studies [23], [24]. One common technique for tracking is the use of invisible tracking pixels [25], deployed on both websites and email. Ruohonen et al. showed that the use of this technique on the web is widespread [6], while Bekos et al. demonstrated how Facebook leverages pixels and click IDs for cross-site tracking even before account registration [26]. More recently, Trampert et al. identified novel privacy risks resulting from modern CSS features and dynamically injected content [27].

Tracking pixels. In this work, we focus on tracking pixels used in emails, a topic previously explored in prior studies, particularly regarding the associated privacy concerns. Englehardt et al. [28] and Xu et al. [4] highlight the serious privacy risks posed by email tracking, showing that up to 30% of emails leak the recipient's email address to one or more third parties. Englehardt et al. [28] find that 85% of emails embed third-party content, and 29% contain the recipient's email address. Similarly, Xu et al. [4] find that up to 25% of emails contain at least one tracking pixel, with the majority being invisible images and the remainder consisting of URLs embedding the recipient's email address either in plaintext or in a hashed form. Fabian et al. [29] analyze more than 600 newsletter emails for several companies, finding that 97.8% of all emails received contained at least one email tracking method. Haupt et al. [10] propose a machine learning model for detecting tracking pixels with a precision of 92%. They report that nearly 50% of newsletters and close to 100% of emails in consumer-oriented industries contain at least one tracking image. Similarly, Fabian et al. [30] present a software tool that detects and sanitizes tracking images in emails. Maass et al. [31] introduce PrivacyMail, a public email privacy benchmarking system to collect tracking information from emails. Ruohonen et al. [6] propose a supervised learning approach and show that 1x1 invisible pixels are still commonly employed for third-party tracking.

Although most previous works focus on detection techniques, Hu et al. [5] conduct a large-scale measurement study. They collect over 2 million emails from disposable email services (*i.e.*, temporary email accounts often used to avoid spam or protect privacy) and analyze tracking behaviors based on common indicators such as 1×1 pixel dimensions, invisibility, and the presence of recipient identifiers (in plaintext or hashed form) within URLs. Their findings reveal that approximately 24% of emails contain at least one tracking

pixel, and 31.5% of URLs are associated with tracking, with invisible 1×1 images being the most common technique. Additionally, they find that most tracking originates from the email senders themselves. Chand et al. [32] show how an attacker can use email tracking techniques to evade anti-phishing systems used by major email providers.

Although prior work has employed detection techniques primarily as tools to support specific analyses of tracking practices, the detection methods themselves have rarely been the subject of systematic scrutiny. Existing studies often assume the reliability of these methods without questioning their accuracy, robustness, or consistency across different contexts. This creates a research gap: Despite the central role of detection in studying online tracking, we still lack a clear understanding of how well current approaches perform in practice, what kinds of tracking pixels they miss, and under which conditions their effectiveness declines. To address this gap, our work shifts the focus from applying detection to critically evaluating it. We assess state-of-the-art detection techniques on a large real-world dataset, revealing their limitations and inconsistencies, and we identify the key challenges to overcome to improve detection in future research.

Anti-Tracking Methodologies. A few studies evaluate the effectiveness of anti-tracking tools. Bielova et al. [18] show that the two most widely used anti-tracking lists, EasyList&EasyPrivacy and Disconnect, fail to block 25.2% and 30.3% of trackers, respectively. Merzdovnik et al. [9] find that while certain browser extensions can block most stateful third-party trackers, they still exhibit significant blind spots. Englehardt et al. [28] show that 70% of emails include at least one resource flagged as a tracker by popular tracking protection lists. Finally, they analyze various email clients and find that only a few either proxy or block images by default.

IV. EXPLORING TRACKING PIXEL DETECTION

As detailed in Section III, scientific studies about tracking pixels differ significantly in their definitions and methodologies, highlighting the lack of a universally accepted definition of what constitutes a tracking pixel. In this section, we first perform a review of which methods for tracking pixel detection are reproducible. Then, we build a novel dataset of emails and evaluate how the different methodologies influence the detection results.

A. TECHNIQUES FOR TRACKING PIXEL DETECTION

Among the works discussed in Section III, we select for this study only those that employ reproducible detection techniques, highlighted in yellow in Table 1. Approaches relying on manual analysis are excluded, as they are not replicable and are susceptible to human bias. In the following, we provide a detailed description of the methodologies used in the reproducible works.

To detect tracking pixels, Hu et al. begin by discarding all `<i mg>` tags whose URLs do not contain any parameters, as a preprocessing step. Then, they classify the remaining `<i mg>` tags as tracking pixels if they satisfy at least one of

Work	Query parameters	(Hashed) email in URL	Tag img dimension	Image dimension	Tag Transparency	Image Transparency	Third-Party	Manual analysis
Hu et al. [5]	•	•	•	•	•	•		
Haupt et al. [10]								•
Xu et al. [4]		•	•					
Ruohonen et al. [6]			•				•	
Fabian et al. [29]								•
Fabian et al. [30]								•
Englehardt et al. [28]		•					•	
Maass et al. [31]								•

TABLE 1: Ground truth table for identifying tracking pixels in prior work. Highlighted entries correspond to studies that use reproducible techniques.

two criteria. The first criterion is based on the image URL: if it contains the recipient’s email address, either in plaintext or as a hashed value, the image is considered a tracking pixel. The second criterion relies on image size: an image is flagged as a tracking pixel if its width and height attributes are less than or equal to one pixel, if it is defined as transparent, or if, once the image is downloaded, its actual dimensions are equal to or smaller than 1×1 pixels. Similarly to Hu et al., Xu et al. [4] classify `<i mg>` tags as tracking pixels if they meet at least one of two criteria. The first criterion is based on image size: an image is considered a tracking pixel if its width and height attributes indicate a "small" size, specifically 0×0 , 1×1 , or 1×3 pixels. The second criterion considers the URL of the associated image. An `<i mg>` is also considered a tracking pixel if the URL contains the recipient’s email address, either in plaintext or hashed using MD5.

Instead, Ruohonen et al. [6] follow an approach based on the domain of the image referenced by the `<i mg>` tag. Specifically, they first identify potential tracking pixels by selecting tags whose URLs are *cross-domain*, meaning that the second-level domain of the image differs from that of the email sender. For these third-party images, they then download the content and measure the actual image dimensions, rather than relying solely on the `width` and `height` attributes declared in the tag. An image from a third-party domain is classified as a tracking pixel if its actual size is exactly 1×1 pixels.

Englehardt et al. [28] also focus on third-party tracking pixels. They consider a URL domain as third-party if it differs from the domain of the service they registered to and the sender’s domain. To detect tracking pixels, they flag any `<i mg>` tag whose URL is third-party and contains the recipient’s email address in plaintext or as a hashed/encoded variant (e.g., MD5, SHA-1, or SHA-256). Tab. 1 shows a summary of previous work on tracking pixel detection along the heuristics they used.

B. DATASET

To collect a dataset of emails that may contain tracking pixels, we follow the approach of [10]. In particular, we create a new email account and subscribe to several popular services

using that email address. To select popular websites, we use Alexa’s ranking system [33], a service that ranked¹ websites by popularity based on estimated daily unique visitors during the preceding three months. From Alexa’s top site rankings as of June 1, 2020, we select the 20 highest-ranked websites from each of the 16 main categories: Adult, Arts, Business, Computers, Games, Health, Home, Kids and Teens, News, Recreation, Reference, Regional, Science, Shopping, Society, and Sports, resulting in a total of 320 services. We subscribed to each service and never interacted with the received emails. In December 2024, we collected all received emails, accounting for more than four years of continuous monitoring. In total, we collected 44,710 emails originating from 620 different domains. The number of domains differs from the number of services (320) as a single service may send emails from multiple domains.

C. METHODOLOGY

In this section, we describe the methodology used to replicate the detection techniques proposed in prior work. Figure 1 illustrates the steps of our data processing pipeline. First, we export the received emails into a dataset in the mbox format, a standard format that stores multiple email messages in a single text file. Next, we extract the HTML content of each email using BeautifulSoup, a Python library for parsing HTML and XML (step 1 in Figure 1). We then extract all `<i mg>` tags from each message by parsing the HTML content, resulting in a total of 1,143,016 image tags (step 2). Using BeautifulSoup, we also extract the `width` and `height` attributes of each `<i mg>` tag (step 3). In addition, we parse the `style` attribute, as it may contain inline CSS specifying size-related properties (step 4). Specifically, we extract the `width` and `height` values defined within this field, along with the `display` and `visibility` properties. We focus on `visibility: hidden`, which conceals the image while preserving layout space, and `display: none`, which removes the image from the rendered content. Next, we analyze the URL associated with each image tag.

¹Note that Alexa was dismissed in 2022

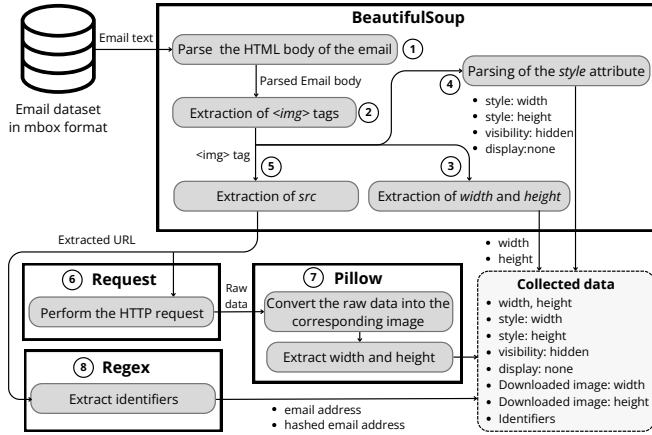


FIGURE 1: The image displays the steps of our data processing pipeline.

We extract the URL contained in the `src` attribute (step 5) and use the Python Requests library to attempt to download each image via the provided link (step 6). We successfully retrieved 85.7% of them, while the remaining downloads failed primarily due to HTTP 404 (Not Found) errors. For each successfully downloaded image, we use the Pillow library, an image-processing library for handling and manipulating image files, to extract its actual dimensions (step 7). Finally, we analyze image URLs for evidence of recipient identifiers, such as plaintext or hashed email addresses, using the hashing schemes described in the original work (step 8). All hashing algorithms used in this study are summarized in Table 8 in the Appendix.

We use the collected information to replicate the techniques described in Sec. IV-A. In particular, to replicate Hu et al. [5], we analyze both the dimensions and visibility specified in image tags and the actual size of the downloaded images. Then, we search for the recipient’s email address in plaintext or hashed form. To replicate Xu et al. [4], we analyze the dimensions specified in image tags and examine whether their respective URLs contain the recipient’s email address in plaintext or hashed form. Unlike Hu et al., who analyze only URL parameters, Xu et al. search for identifiers across the entire URL. As for Ruohonen et al. [6], we extract the size of each downloaded image file and check whether the second-level domain of each image URL differs from the email sender’s domain registration. Finally, to replicate Englehardt et al. [28], we retain only third-party `` tags, defined as tags where the domain in the `src` URL differs from the sender’s domain. We then inspect each URL for the presence of the recipient’s email address, either in plaintext or hashed form, using the hashing algorithms specified in their study.

D. RESULTS

Table 2 presents the number and percentage of image tags identified as tracking pixels by each detection method, and the number and percentage of emails containing at least one such pixel. The results highlight substantial variation across

Work	# of tracking pixels	# of email
Hu et al. [5]	186, 413 (16.3%)	28, 977 (64.8%)
Xu et al. [4]	210, 213 (18.4%)	38, 710 (86.6%)
Ruohonen et al. [6]	101, 560 (8.9%)	15, 699 (35.1%)
Englehardt et al. [28]	54, 670 (4.8%)	4, 580 (10.2%)

TABLE 2: Number of tracking pixels and their percentage over all image tags, and number and percentage of emails containing at least one tracking pixel.

detection approaches. In particular, the method proposed by Xu et al. identifies 210,213 tracking pixels, nearly four times more than the technique of Englehardt et al., which detects only 54,670 tracking pixels. The disparity becomes even more evident when considering the number of emails flagged. Using Englehardt et al.’s method, only 10.2% of the 44,710 emails in our dataset contain at least one tracking pixel. In contrast, Hu et al.’s heuristic flags 86.6% of emails as containing tracking pixels. In the following paragraphs, we compare the heuristic proposed by the state of the art to highlight the challenges in detecting email tracking.

1) Third party-based heuristics

We begin by analyzing the techniques proposed by Englehardt et al. and Ruohonen et al., which focus exclusively on detecting third-party trackers. To identify tracking pixels, these approaches first detect third-party images and then perform additional checks, such as image size or the presence of unique identifiers, only within this subset. The authors consider an image tag as third-party if its domain differs from both the domain of the service to which the user subscribed and the domain of the sender. While this definition is conservative, it is not immune to false positives. In particular, our manual inspection reveals the proposed heuristic misclassifies organizationally affiliated domains. For example, Disney communicates using multiple domains, including `disney@mail.disney.it`, `starwars@mail.it.starwars.com`, and `support@espn.com`. These domains are classified as third-party tracking by the proposed heuristic, even if they are all entities controlled by the same company. We perform a manual inspection on all the URLs detected by Englehardt et al. and Ruohonen et al. as third-party, by grouping related domains based on common corporate ownership (i.e., the same parent company). To determine whether domains share a common corporate owner, we relied on a combination of tools, including WHOIS records, Netify² (a network intelligence service that provides domain ownership data), and targeted web searches. This analysis reveals that their detection technique incorrectly classifies 8,572 image tags as third-party. In addition, detection methods that examine only the domain serving the image cannot identify cases where a third-party tracker is masked behind a first-party URL. In such cases, the URL initially points to a domain controlled by the email

²<https://www.netify.ai/resources/domains>

sender, but the request is redirected to a third-party tracking service, effectively bypassing domain-based filtering. To assess this, we perform HTTP requests to each image URL and inspect the full redirect chain. In 5.76% of cases, the request ultimately resolves to a third-party domain, even though the original image tag points to a first-party domain. These cases are missed by Englehardt et al. and Ruohonen et al. which rely solely on static analysis of the URL. Finally, approaches that focus exclusively on third-party domains also overlook tracking performed directly by first-party services.

While third-party-based heuristics are often presented as a conservative baseline, our analysis shows they are error-prone and may be overly restrictive. Identifying organizational relationships between domains is non-trivial and requires a significant manual effort. Moreover, third-party tracking can be obfuscated behind first-party domains, which represent the majority of our dataset. One such example is URLs we find in the form of *sli.domain* (See an example of *sli.webmd* in Fig. IV-D2). In these cases, the URL initially appears to point to the original registration domain but ultimately redirects to a third-party tracking service.

2) Image size based heuristics

Hu et al. and Xu et al. identify significantly more tracking pixels than Englehardt et al. and Ruohonen et al. This discrepancy occurs primarily because the authors do not restrict their analysis to third-party domains in these works. Instead, their heuristics rely on characteristics of the image tags themselves, and the presence of user identifiers within the URL. However, the two works adopt different definitions for identifying image size and locating user identifiers. As a result, 8,001 tracking pixels are detected using the definition proposed by Hu et al., but not by Xu et al. Conversely, 31,801 tracking pixels are identified using Xu et al.'s definition but not by Hu et al. This discrepancy occurs because both Hu et al. and Xu et al. consider that tracking pixels should be associated with "small" images that are likely to be invisible to the user. However, Xu et al. consider only the dimensions specified in the HTML attributes of the image tag, while Hu et al. take into account the minimum size between what is specified in the tag and the actual size of the image downloaded.

A first consideration is that the approach of Hu et al. is suitable for post-hoc analysis but requires downloading the image, which is not possible in a real-use scenario, as it would trigger the tracking of the user. However, the approach of Xu et al. still has a limitation, since the analysis of our dataset shows that only 55% of image tags explicitly include size attributes in the HTML. By comparing the tracking pixels identified by the approach of Hu et al. but missed by Xu et al., we observe that some images that are slightly larger than 1x1 but still function as tracking pixels. For instance, we identify 3,682 image tags with dimensions of 2x6 pixels.

```

```

These tags are associated with 1x1 transparent GIFs, making them effectively invisible to the user despite their HTML-specified size. Since Hu et al. evaluate the size of the actual image (in addition to the HTML tag), they successfully detect these cases, whereas Xu et al. cannot.

Conversely, among the tags flagged by Hu et al., we find examples that are unlikely to serve any tracking purpose. Specifically, some 1x1 pixel images are visually enlarged using HTML attributes and appear to be used solely for layout or stylistic purposes. Consider the following example:

```

```

In this case, only the height attribute is specified, so the image retains its original width of 1 pixel, resulting in a vertical line of 15 pixels. This tag is detected by Hu et al. as a tracking pixel because the associated image file is a 1x1 pixel. Xu et al., in contrast, only inspect the dimensions specified in the HTML tag, and, therefore, do not flag this tag. A manual inspection of the email reveals that the image is rendered as a visible gray horizontal line in the header, likely used for layout or stylistic purposes rather than tracking.

Beyond these observations, the analysis of our dataset reveals additional challenges associated with size-based heuristics. In particular, the assumption that tracking pixels are necessarily small or transparent images is not always valid. Tracking can also be implemented using larger, visible images. We identify several cases where large, single-color images are used as trackers. For example, Figure 2 shows an email containing a purple rectangle, highlighted by a rectangular dashed box. The HTML code for the image is:

```

```

We can hypothesize that this tag is used for tracking, as the hash of the user's email is embedded in the URL path. Although the image is large (520x30 pixels) and technically visible, it blends seamlessly into the email layout (it's colored with the same purple color as the above ads), thus remaining unnoticed by the user. This example highlights that the common assumption that tracking pixels must be small to be unnoticed is simplistic; invisibility can be achieved through other techniques.

3) Identifier based heuristics

Finally, we compare the detection results of Hu et al. and Xu et al., focusing specifically on how each approach detects user identifiers embedded in image URLs. Both heuristics attempt to match the user's email address, in either plaintext or hashed form, against components of the URL. Xu et al. adopt a broader strategy, scanning the entire URL string, including both the path and the query parameters. In contrast, Hu et al. restrict their analysis to query parameters only (e.g., ?id=user@example.com), which may cause their heuristic

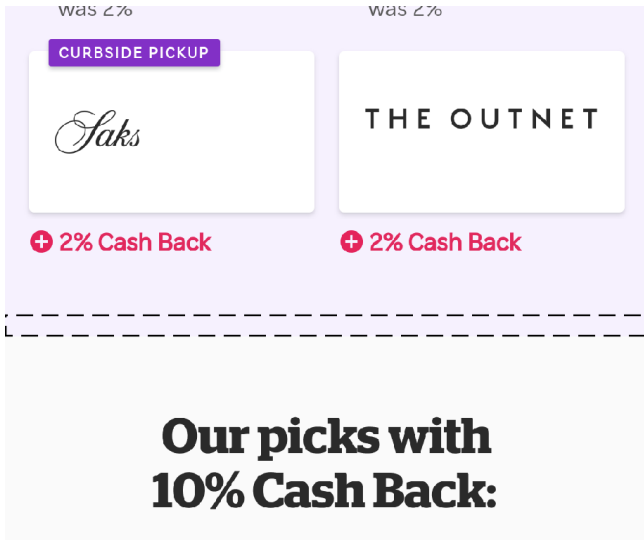


FIGURE 2: The image displays a segment of an email containing a large purple rectangle (520×30 pixels), highlighted with a black border, which functions as a tracking pixel.

to miss identifiers embedded elsewhere in the URL. For instance, we observe cases where the identifier is embedded in the path of the URL rather than in its parameters. For example:

```

```

In this case, the user identifier `1fe45e0403d91d092b29bf0e12a00507` (a hash of the email address) appears in the path segment of the URL. Since Hu et al. examine only the parameters, their heuristic would not flag this image tag, underscoring a limitation in their detection approach. Moreover, limiting the search to only the plain or hashed version of the recipient’s email is a significant limitation. This method would fail to identify other forms of custom identification, such as the simple use of an incremental unique number. This significantly limits the comprehensiveness of identifier-based detection techniques. As we show in the following section (Section V), the use of recipient email addresses, whether in plaintext or hashed form, is now relatively rare among tracking systems, further reducing the coverage of such approaches.

E. LONGITUDINAL ANALYSIS

As a further analysis, we perform a longitudinal study of tracking techniques over the four-year collection period. Fig. 3 shows the proportion of tracking pixels detected by each detection methodology, aggregated monthly. The figure shows that the methodologies proposed by Hu et al. and Xu et al. (blue and orange areas) consistently detect more pixels than those of Ruohonen et al. and Englehardt et al. (green and red areas). A closer look at the two most effective

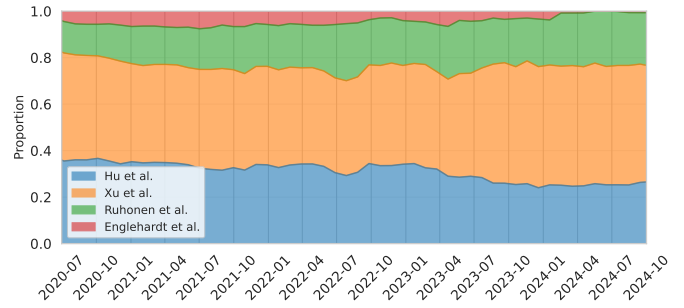


FIGURE 3: Proportion of tracking pixels detected over time by the four detection techniques.

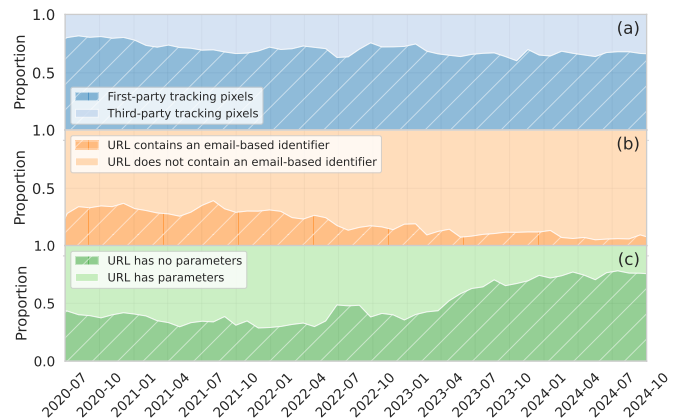


FIGURE 4: Proportion of tracking pixels identified by the four techniques that are first-party (a), contain the recipient’s email address in plaintext or hashed form (b), and include parameters in the URL (c).

methodologies highlights that the performance of Hu et al. decreases over time. Indeed, there is an important distinction between the two methodologies: the key difference is that Hu et al. search for identifiers exclusively within the parameters, whereas Xu et al. also examine the path of the URL to locate identifiers. Fig. 3(c) shows that the proportion of tracking pixels without URL parameters rises from 40% in the first two years to nearly 90% by the end. This trend explains the growing effectiveness of Xu et al.’s methodology, since identifiers are increasingly embedded in the path instead of being carried as URL parameters. By contrast, the methodologies of Ruohonen et al. and Englehardt et al. detect substantially fewer pixels. We hypothesize that this discrepancy arises because these methodologies exclusively detect third-party tracking pixels. To assess whether this factor explains the discrepancy, we plot the proportion of first-party and third-party tracking pixels over the entire collection period (Fig. 4(a)). The plot shows that first-party tracking remains more prevalent throughout the entire period, which explains why restricting detection to third-party tracking pixels results in fewer pixels being identified overall. An additional insight from Fig. 4(a) is the increase in third-party tracking pixels, which grows from 18% at the beginning of the data collection to over

30% by the end of the period. However, despite the overall increase in third-party tracking pixels, the methodologies that specifically target them (green and red areas in Fig. 3) do not show a significant increase in the number of detected tracking pixels over time. In particular, the methodology proposed by Ruohonen et al. exhibits only a modest increase, while the technique of Englehardt et al. detects almost no pixels during the last six months of the collection period.

To better understand this phenomenon, we compare the two methodologies and observe that the approach of Englehardt et al. requires the associated URL to explicitly contain the email of the receiver in plaintext or hashed form, whereas the methodology of Ruohonen et al. does not impose this constraint. However, Fig. 4(b) shows that this type of identifier becomes much less prevalent toward the end of the collection period (35% in the first year versus 10% in the last year), explaining the decline in tracking pixels detected by the methodology of Englehardt et al.

Among these insights, we present a case study that illustrates a shift in the form of identification strings used in URL parameters. Specifically, we observed that the Hindustan Times, one of the largest Indian newspapers and the second-largest English-language daily [34], changed its tracking technique on October 29, 2021. Earlier emails embedded the Base64-encoded recipient email directly in the URL path of the tracking pixel, whereas later emails adopted a different form of identifier based on an alphanumeric string. In particular, this identifier is not captured by applying the hashing and encoding transformations of previous works, listed in Table 8. This example highlights how tracking implementations can evolve over time to adopt new techniques.

V. COMMERCIAL TRACKING PIXELS IMPLEMENTATIONS

The previous section shows that tracking pixels are difficult to detect using existing academic techniques and that many implementations can easily evade current heuristics. To shed light on how tracking pixels are actually deployed, we examine the real-world practices of third-party services that specialize in email tracking. We begin with a systematic analysis of popular tracking providers to uncover the technical details of their tracking mechanisms. Next, we evaluate whether the pixels embedded by these services are effectively detected by the heuristics proposed in previous work. Finally, we investigate whether email clients implement measures to block such tracking attempts.

A. COMMERCIAL TRACKING PLATFORMS

To analyze commercial tracking pixel implementations, we conduct a comprehensive web search to identify the most widely deployed tracking platforms. We employ targeted keyword searches using terms such as "email tracking services," "pixel tracking platforms," "email open tracking," and "commercial email analytics." In this way, we identify 29 popular

tracking services³. To perform our analysis, we attempted to register on 29 platforms and successfully registered on 22 services. Specifically, we excluded platforms that provide only aggregated campaign-level analytics without individual recipient tracking.

After registering on the commercial email tracking platforms, we investigate how these services operate. We find that these platforms allow users to send emails that include a dedicated tracking pixel. The image is hosted by the tracking provider, which logs access events when the email client retrieves it. Users of these platforms are then notified, either via email alerts or through a web dashboard, when a recipient opens the tracked email.

1) Tracking pixels implementation

We systematically analyze the implementation of tracking pixels by each of the 22 commercial tracking services we registered for. To this end, we create and send tracking emails using each platform, then inspect the HTML content to understand how the corresponding tracking pixels are embedded. Our analysis starts by focusing on the HTML attributes associated with each pixel tag. We observe that platforms manipulate attributes such as width, height, and style to control the visual appearance of the pixel. Most services aim to make the tracking pixel invisible: 9 services set both dimensions to 0×0 , and 10 services use 1×1 dimensions. Among the remaining platforms, one uses CSS to render the pixel fully transparent, another references a URL that returns a non-existent image (therefore not displayed), and only one platform uses a visible image that explicitly signals its tracking purpose.

We then analyze how services identify individual recipients through tracking URLs. The first approach implemented by 13 services relies on using URL parameters such as "u", "user", "id", "eId", "e", "rId", "r", "guid", or "userId". These parameters typically contain integer-based identifiers or strings that link back to specific recipients. The second strategy used by the other 9 services involves embedding the recipient's information directly in the URL path. One service includes the recipient's email in plaintext, while the other 8 services embed what appear to be hashed or encrypted identifiers.

B. DETECTION OF COMMERCIAL TRACKING SERVICES

In this section, we investigate the effectiveness of tracking pixel detection in real-world scenarios. First, we apply the detection techniques reported in Sec. IV-A to assess whether they can accurately identify the tracking pixels embedded by third-party services. Then, we evaluate whether browsers, web clients, and desktop email clients can detect and block

³Salesflare, HubSpot, Litmus, Campaigner, Validity, Constant Contact, Amazon Simple Email Service (SES), Iterable, Adobe Audience Manager (Demdex), Mailbutler, Mailsuite, Snovio, Streak, MailTracker, Woodpecker, Yesware, MySignature, SendGrid, Lotame, Right Inbox, GMass, MixMax, Mailtag.io, Respona, Autoklose, Cirrus Insight, Velloxy, Vocus.io, and Boomerang.

these tracking pixels. Finally, we evaluate the prevalence of commercial tracking pixels in our email dataset.

1) Can academic techniques detect commercial tracking services?

We evaluate the effectiveness of the detection methodologies described in Section IV against the 22 commercial tracking services identified in our study. The detection technique proposed by Hu et al. [5] successfully identified tracking activity in 20 out of the 22 services. These services were detected primarily because they embed images with dimensions smaller than or equal to 1 × 1 pixels, a pattern the technique is designed to identify. However, two services evaded detection. The first is GMass, a service that explicitly advertises countermeasures to evade tracking pixel blockers [35]. According to their documentation, many blockers rely on analyzing HTML tag attributes and URL parameters, as also shown in Sec IV. To bypass such detection, GMass uses encrypted image URLs with no exposed parameters and the corresponding `` tags do not include explicit sizing information. This technique is shown in the following example, collected from the GMass documentation:

```

```

These countermeasures are sufficient to evade the detection approach proposed by Hu et al. The second undetected service is Boomerang, which takes a markedly different approach. Rather than attempting to conceal its tracking behavior, Boomerang makes it explicit by embedding a large image at the end of the email with a message such as: “*The sender has requested a read receipt. If you do not wish to provide one, click here.*” While this method is clearly noticeable to human recipients, the detection algorithm by Hu et al. fails to flag it because the image is not a small pixel and the URL does not contain any identifying parameters (e.g., the recipient’s email or its hashed version).

The method proposed by Xu et al. [4] has slightly lower performance, identifying 19 out of 22 services. In addition to Boomerang and GMass, it also failed to detect Vocus.io. In this case, Hu et al.’s method was able to identify the tracking pixel by analyzing the dimension of the image, even though the `` tag does not specify size attributes, while Xu et al.’s method, which relies on HTML attributes, failed to detect it. The technique proposed by Ruohonen et al. [6] focuses on detecting third-party domains and then considers as tracking pixels all the images with an actual dimension of exactly 1 × 1. In our analysis, all 22 commercial services were successfully identified as third parties according to their heuristic for third-party detection. However, only 12 of them included image tags with a 1 × 1 size, and thus only those were flagged as tracking pixels. Englehardt et al. [28] also focus on third-party detection, but adopt a different strategy. In addition to

checking for third-party domains, they search for the presence of the recipient’s email address or its hashed version in the image URL. Using this technique, only one commercial service in our dataset was detected, as the remaining services rely on alternative forms of identifiers (e.g., encrypted tokens or session-based tracking) that do not expose the recipient’s email in the URL.

Tab. 3 summarizes the results. Overall, our findings show that modern commercial tracking services can evade academic detection methods through obfuscation, larger visible images, or encrypted URLs.

Work	# of blocked commercial services
Hu et al. [5]	20
Xu et al. [4]	19
Ruohonen et al. [6]	12
Englehardt et al. [28]	1

TABLE 3: Number of commercial services blocked by applying the academic detection methodologies.

2) Do email clients block commercial tracking services?

In this subsection, we seek to understand whether users are offered protection against commercial tracking pixels by their most commonly used email clients.

To this end, we evaluate whether the most widely used email clients, both web-based and desktop, actively block or mitigate tracking mechanisms. To identify the most widely used email clients, we consulted publicly available usage statistics [36]–[38], which indicate that the most popular web clients are: Gmail, Yahoo Mail, and Microsoft Outlook (Web). Instead, the most popular desktop clients reported are: Microsoft Outlook (Desktop), Mozilla Thunderbird, and Apple Mail. Moreover, we included in our evaluation the web and desktop versions of Proton Mail, one of the most widely adopted privacy-first email services, with over 100 million users [39]. For web clients, the browser used to open the email is also a crucial factor, as their security and privacy settings can directly affect whether tracking pixels are loaded. To evaluate the impact of browser choice, we repeated our experiments using the four most popular browsers: Google Chrome, Mozilla Firefox, Microsoft Edge, and Apple Safari. Additionally, we included Brave, which is well known for its privacy features.

For the evaluation, we created an email account on Gmail, Yahoo Mail, Outlook.com, and Proton Mail. We sent one email for each of the 22 commercial tracking pixel services to each of these accounts. We then opened these emails using all possible combinations of browsers and email clients (web and desktop), and recorded whether the tracking pixel was blocked. In Appendix A we report the experimental setup.

Web Clients. Analyzing the behavior of browsers with their default settings, we noticed that Gmail, Yahoo Mail, and Outlook Web do not block any tracking services. In contrast, Proton Mail Web blocks all trackers by default. To understand

the reason behind this effectiveness, we analyze Proton’s approach to tracker blocking. We find that Proton relies on a list-based filtering mechanism: URLs found in `` tags are compared against a set of known tracker domains, and if a match is found, the corresponding resource is not loaded. This blocking behavior remains consistent across all tested browsers. We also examined whether web clients offer user-facing controls to block tracking pixels. Gmail and Yahoo Mail allow users to choose, on a per-email basis, whether to load external images. Outlook Web provides the option to maintain a blocklist of senders whose images should not be automatically displayed. However, these mechanisms do not block tracking by default and rely on disabling all images in the email, which may degrade user experience.

Desktop Clients. First, we perform the experiment without modifying any of the default settings, as these are the ones initially applied and are arguably used by the majority of users. Mozilla Thunderbird prompts the user each time an email is received from a new sender, asking whether to load external images. Consequently, all commercial trackers are blocked unless the user explicitly consents. Microsoft Outlook Desktop does not block any tracking pixels, regardless of settings. Apple Mail blocks only three tracker services: Mailbutler, Mailtracker, and Yesware. Upon investigation, we find that Apple Mail provides Mail Privacy Protection (MPP) by default. MPP works by preloading remote content, including tracking pixels, on proxy servers operated by Apple, shortly after the email is delivered. Later, when the user opens the email, Apple Mail displays the content retrieved from those proxy servers instead of requesting it directly from the server controlled by the sender [40]. As a result, the tracking pixel is still fetched, but the request originates from Apple rather than from the user’s device, making it impossible for the sender to determine when, where, or even whether the email was actually opened. Like in the web client, Proton Mail Desktop consistently blocks all embedded trackers by default.

We further investigated additional tracking protection features. We find that Outlook offers an option to block images, but this feature is only available when using an Outlook account and is not supported for third-party email providers. Thunderbird does not provide additional functionalities that explicitly block email tracking beyond the image-blocking setting. Finally, Apple offers only the Mail Privacy Protection feature, which is enabled by default.

3) Prevalence of Commercial Tracking Services

As the final step of this section, we assess the real-world prevalence of the third-party tracking services we analyzed. To perform this analysis, first we identify the image tags within our dataset that reflect the known implementation techniques used by the commercial tracking services identified in Sec. V-A. Then, we confirm a match by verifying that these images’ URLs include the domain associated with that tracking service. This approach detects 10 out of the 22 tracking services actively employed in our dataset. Tab. 4 presents the prevalence for each detected service, including

the number of unique emails and senders utilizing these tracking platforms. The results reveal significant variation in adoption across different services. Litmus is the most prevalent service, appearing in 3,310 URLs across 3,259 unique emails (7.23% of our dataset) from 30 different senders (4.84%). Demdex (Adobe Audience Manager) and Validity also show substantial presence, appearing in 4.69% and 4.45% of unique emails, respectively. In particular, Validity emerges as the service employed by more senders in our dataset (5.16% of all senders). Interestingly, while HubSpot appears in a considerable number of URLs (1,690), these correspond to only 94 unique emails (0.21% of the dataset), suggesting that HubSpot embeds multiple tracking pixels per email or employs tracking URLs with many parameters that generate numerous unique combinations. This pattern contrasts with services like Campaigner, Amazon SES, and Constant Contact, where the number of URLs closely matches the number of unique emails, indicating a more straightforward one-pixel-per-email implementation.

TABLE 4: Prevalence of commercial tracking services in the email dataset. The table shows the number and percentage of tracking URLs, unique emails containing these services, and unique senders employing them.

Service	URLs (%)	# Emails (%)	# Senders (%)
Litmus	3,310 (0.29)	3,259 (7.23)	30 (4.84)
Demdex	2,308 (0.20)	2,096 (4.69)	13 (2.09)
Validity	2,237 (0.20)	1,988 (4.45)	32 (5.16)
HubSpot	1,690 (0.15)	94 (0.21)	6 (0.97)
Campaigner	771 (0.07)	771 (1.73)	6 (0.97)
Constant Contact	753 (0.07)	753 (1.68)	2 (0.32)
Amazon SES	708 (0.06)	708 (1.58)	3 (0.48)
SendGrid	569 (0.05)	124 (0.28)	15 (2.42)
Iterable	443 (0.04)	443 (0.99)	2 (0.32)
Lotame	72 (0.01)	16 (0.04)	1 (0.16)

Overall, this result shows that approximately 15% of the services in our dataset leverage these popular third-party tracking services. This relatively low adoption can be attributed to our data collection methodology, specifically our registration on high-profile websites and services. Large technology companies and established platforms could opt to develop their own in-house email tracking solutions rather than relying on third-party services, allowing them greater control over data collection and analytics infrastructure. Additionally, our detection approach may underestimate the true prevalence of certain tracking services. Indeed, it is possible, and sometimes recommended by the tracking services themselves, as in the case of Mailchimp [41], to configure popular tracking services to use custom domains that match the sender’s brand, effectively masking their involvement in the tracking process. In such cases, tracking pixels served by these platforms would appear to originate from the sender’s domain rather than the tracking service’s domain, causing them to be missed by our domain-matching methodology. This result suggests that the actual prevalence of commercial tracking services may be higher than our measurements indicate, particularly for other

popular tracking services that offer domain customization features.

VI. LIST-BASED TRACKING PIXEL DETECTION

The previous section shows that Proton Mail is the only email client capable of consistently blocking all tracking pixels embedded by third-party commercial services. Thus, it arguably represents the most effective option for users seeking to avoid such tracking. However, adopting Proton would require most users to migrate from their current email provider or set up a forwarding mechanism to Proton. This solution is not always practical, as it may introduce friction in daily usage, require configuration effort, or lead to compatibility issues with existing workflows and accounts.

Given these practical limitations, we explore alternative detection mechanisms that can provide comparable protection against tracking pixels but are public, transparent, and easily integrable into various email clients. In the following, we evaluate whether solutions based on publicly available blocklists (mentioned in Sec.II), such as those used by popular browser extensions, can achieve the effectiveness demonstrated by Proton, while offering greater compatibility and ease of adoption.

A. DETECTION OF PUBLIC TRACKING SERVICES

As a first experiment, we evaluate the effectiveness of public blocklists in detecting the commercial tracking services analyzed in the previous section. To compare Proton Mail’s behavior with publicly curated blocking lists, we collect the two most widely used lists identified in prior work [18]: EasyPrivacy [11] and EasyList [12]. We also include several other popular blocklists: the Ugly Email Trackers list [13] and the uBlock Origin tracking filter list [15] respectively used by the popular browser extensions Ugly Email [14] and uBlock Origin [16]. Moreover, we include the StevenBlack hosts list [17], which aggregates multiple blocklists and has collected nearly 30,000 stars on GitHub. These lists consist of regular expressions that match known tracking URLs or domains.

We evaluate how many commercial tracking services are detected by each blocklist. To detect flagged URLs, we implement a Python pipeline that downloads public lists and parses their rules. Since most lists already provide rules in the form of regular expressions or regex-like strings, we adapt them into executable Python regex objects, preserving Adblock-style filter syntax [42] and host-based entries. These rules are then applied to all URLs in our dataset. For each flagged URL we record the matched rule and its source list. Tab. 5 shows the results of the detection and the number of regexes used by each list. The table shows that there is a large discrepancy in performance among the analyzed lists. In particular, UglyEmail is the list that has the best performance, detecting 11 tracking services, while Easylist detects none. There is an overlap between the commercial services detected, as the total number of services detected is 16. The most commonly detected services are Yesware, Emltrk, and Awstrack, which

are matched by three different lists. Interestingly, the number of regular expressions in each list does not directly correlate with higher detection rates. The list with the most matches is UglyEmail, which uses only 58 regular expressions, achieving better performance than lists that use tens of thousands of regular expressions, like Easylist or StevenBlack. Several factors may contribute to this behavior. Indeed, most evaluated lists are used to block trackers or advertising on the web, but are not specifically designed to detect email tracking pixels. Instead, the only one explicitly tailored for email tracking detection is the UglyEmail list, which is used by the popular UglyEmail browser extension.

List	# of blocked services	# of regexes
Easylist	0	70,372
EasyPrivacy	8	54,754
UglyEmail	11	58
uBlock Origin	3	1,761
StevenBlack	7	227,998
Total	16	

TABLE 5: Number of commercial services blocked by using public blocklists and their number of regexes.

B. DETECTION IN THE WILD

The previous subsection shows that public blocklists perform well in blocking commercial tracking pixels but still fall short of Proton’s performance. However, this analysis is limited by the number of sample of commercial services considered. Moreover, since the commercial tracking services we analyze are well known, it is expected that Proton would detect and block them.

Thus, we further assess the effectiveness of these methods by analyzing the detection results on our email dataset. To estimate how many tracking pixels are detected by each public blocklist, we extract the URL of each of the 1,143,016 img tags in our dataset, obtaining 430,926 unique URLs. Then, we match each URL with the regular expressions contained in the lists. However, evaluating the performance of Proton on our dataset is more challenging, as the lists used by Proton are not publicly available. To overcome this problem, we sent our entire dataset of 430,926 unique URLs directly to a dedicated Proton Mail address. Given the size of the dataset, we automated the email creation and tag incorporation using a Python script. We grouped the URLs into batches of 700, ensuring no URL repeated across batches.

Table 6 reports the comparison between the URLs flagged by Proton Mail and those identified by public blocklists. For each source, we provide the total number of flagged URLs and emails, and the unique contribution, defined as the set of URLs or emails exclusively flagged by that source. Overall, Proton Mail blocked 92,377 unique URLs, accounting for 21.4% of all unique URLs in our dataset. The unique contribution analysis shows that Proton and the public blocklists detect complementary sets of URLs, with each identifying some that the other misses. Specifically, Proton uniquely

flagged 23,631 URLs and 1,413 emails, corresponding to 18.2% of all flagged URLs and 4.3% of all flagged emails. A manual inspection of the URLs revealed that a large proportion follow variants of the `sl.i.domain` pattern described in Sec. IV-D1. In particular, we find that 86% of the 23,631 URLs flagged only by Proton matched two variations of the `sl.i.domain`: `li.domain` and `s.sl.i.domain`. This finding is significant, as it suggests that incorporating regular expressions for these variations into public blocklists could substantially reduce the detection gap between Proton and blocklist-based approaches.

Examining the unique contributions of public blocklists reveals that they flag a substantial number of URLs not detected by Proton. This difference is even more pronounced when considering flagged emails: for example, EasyPrivacy alone uniquely detects 7,214 emails, representing 22.1% of all flagged emails in our dataset. To better understand these detections, we perform a manual analysis of tracking pixels identified by public blocklists but not by Proton. Many of these URLs originate from domains linked to marketing or analytics services (e.g., `responsys.net`, `braze.eu`, `appboy-images.com`, `iterable.com`). A significant proportion also use subdomains such as `click`, `links`, `email`, or `promo`, which are strongly indicative of infrastructures for click tracking or email campaign management. Additionally, several blocklists contain patterns targeting tracking pixels hosted as static images on content delivery networks, often reflected in URLs containing subdomains like `static.cdn` or `simplecdn`. Overall, this analysis suggests that, although public blocklists may be less restrictive than Proton's approach, the URLs they uniquely flag are still strongly associated with tracking activity.

The analysis also indicates that Proton's detection capabilities can be partially approximated by leveraging public blocklists, particularly when these lists are augmented with a small number of targeted rules tailored to detect patterns observed in URLs identified exclusively by Proton. However, the fact that public blocklists flag a substantial number of tracking domains missed by Proton further corroborates that there is no definitive solution for tracking pixel detection.

VII. DISCUSSION AND FINAL REMARKS

In this work, we perform a comprehensive analysis of tools that can detect and block email tracking pixels. By applying prior detection heuristics to a novel dataset of more than 44,000 real-world emails, we show that detection outcomes vary greatly depending on the methodology used, with the percentage of flagged emails ranging from 10.2% to 86.6%. We also evaluate 22 commercial tracking services and find that even minimal evasion techniques, such as encrypted URLs, non-standard dimensions, or transparent styling, are enough to bypass academic approaches. In addition, our experiments with popular email clients show that mainstream platforms offer little protection by default, and only privacy-focused clients such as Proton Mail appear to provide effective blocking. However, when we run Proton on our dataset

of real emails and compare the URLs blocked by Proton's private blocklist with those blocked by public blocklists, we find that none of these lists, when used alone, can detect all the tracking pixels present in our dataset. Based on all these experiments, in the following, we present the main lessons learned from our analysis and our study of the tracking pixel ecosystem.

Challenges in detecting small and invisible pixels. Detecting tracking pixels is an extremely challenging task, as demonstrated in this work. The primary difficulty stems from the fact that tracking pixels are characterized by their tracking functionality rather than by intrinsic properties such as size or visibility. Although these pixels are often described as small and invisible, their real significance lies in their capacity to monitor user behavior, not in their visual appearance within the email. The actual tracking behavior is implemented on the backend infrastructure hosting the tracking image, and only the operator of that server can be certain whether tracking occurs. By contrast, any external detection technique must infer tracking intentions indirectly, for example by analyzing the image's dimensions, color, or encoded parameters within the URL that references it.

This reliance on inference introduces uncertainty and makes it easy for service providers to evade detection. Indeed, even the most precise detection methods can be circumvented. For instance, previous research often assumes that a tracking pixel is an image with maximum dimensions of 1×1 pixel. However, a tracker can easily bypass this detection rule by using an image of a slightly different size, such as 1×2 pixels, or by providing a larger image that is then resized via CSS. If the detection instead relies on CSS-rendered size, the actual image dimensions might still evade detection. Similar limitations apply to approaches that attempt to block images based on the hosting domain, such as blocking third-party domains. Service providers can work around this by using URL redirections or by embedding tracking images on the same domain used by the email sender, thereby defeating domain-based blocking.

How email users can stop tracking pixels. Because of these limitations, achieving robust protection against tracking pixels with detection-based methods alone is difficult. Proxy solutions, such as Mail Privacy Protection (MPP) provided by Apple Mail (see Subsection V-B2), partially address this challenge by concealing personal information like the user's IP address and browser agent. Nonetheless, these approaches do not fully prevent behavioral tracking, since it is still possible to infer when an email is opened.

Another approach is to use the Brave browser and configure blocklists or browser extensions that rely on public blocklists. By using these tools, requests to known tracking domains can be blocked, preventing the sender from receiving any information. However, the effectiveness of this method depends on sustained efforts to maintain and update the blocklists, as tracking services frequently evolve in response to new detection techniques. Given the complementary strengths and

list	# URLs	# Unique contribution of URLs	# Emails	# Unique contribution of Emails
EasyList	53,306 (41.2%)	123	2,874 (8.8%)	35
EasyPrivacy	83,159 (64.2%)	14,032	25,022 (76.7%)	7,214
UglyEmail	16,985 (13.1%)	2,934	14,822 (45.5%)	13
uBlock Origin	4,744 (3.7%)	171	4,546 (13.9%)	40
StevenBlack	17,367 (13.4%)	13,911	5,460 (16.7%)	278
Proton	92,377 (71.3%)	23,631	20,601 (63.1%)	1,413
Total flagged	129,524 (30.0%)		32,603 (77.5%)	

TABLE 6: Result of flagged URLs and emails per list and Proton results.

limitations of proxy-based solutions and blocklist-driven detection, combining these approaches may offer a more comprehensive defense. Integrating proxies to obscure personal information, alongside blocklists to actively prevent known tracking domains from receiving requests, can significantly reduce the risk of user tracking in email environments. However, it still requires manual configuration and a level of technical understanding that is likely beyond the average user.

A final approach that ensures not to be tracked and maximizes privacy is disabling image loading in the web mail client or the browser entirely. While technically effective in preventing tracking pixels from loading, this solution comes at the cost of severely degrading the user experience, as emails and websites appear broken or incomplete without visual content.

Tension between compliance and performance in email tracking services. Article 5(1)(a) of the GDPR establishes the principles of lawful, fair, and transparent processing of personal data, while Articles 6 and 7 specify the conditions for obtaining valid consent. According to these provisions, tracking technologies such as email tracking pixels must not be used unless the recipient has given explicit, informed consent. The email tracking services analyzed in Section V demonstrate different approaches that highlight a fundamental tension between compliance and performance. Some services prioritize extreme transparency, emphasizing regulatory compliance, while others employ obfuscation techniques that favor tracking performance over transparency. For example, GMass uses obfuscated tracking pixels to bypass popular detection tools like PixelBlock, UglyMail, and Email Privacy Protector and explicitly promotes this ability⁴. Although this technique appears to conflict with GDPR principles, GMass claims compliance by placing the responsibility on the email sender to obtain explicit consent from recipients prior to tracking. This approach allows GMass to optimize for invisibility and performance while shifting the full legal burden onto the users of its service. In contrast, Boomerang incorporates transparency into its tracking mechanisms by notifying recipients when read receipts are requested and offering clear opt-out options. While this method may reduce tracking efficiency, it aligns more closely with GDPR’s emphasis on user autonomy and data protection by design [43]. This contrast

reflects a tension in design priorities: platforms like GMass prioritize analytics, whereas services like Boomerang embed compliance and consent mechanisms into the user experience.

Interplay between user interactions and tracking mechanisms. A promising direction for future work is to evaluate whether user interactions influence the tracking strategies employed by companies. In particular, it is interesting to investigate whether user adoption of specific anti-tracking techniques prompts companies to develop new methods that bypass these protections. Measuring this effect directly is extremely challenging, as only companies have access to the data required to assess changes in tracking efficacy over time. An alternative and more feasible approach is to evaluate whether tracking behavior adapts to observable user actions, such as clicking on emails. This can be studied using an experimental design with two parallel datasets: one serving as a control with no user interactions, and the other including repeated and controlled interactions over time. Comparing these datasets allows measuring if and how tracking strategies dynamically respond to user behavior.

VIII. ACKNOWLEDGMENTS

This work has been partially funded by projects: MUR National Recovery and Resilience Plan, SERICS (PE00000014); and ST3P (B83C24003210001) under the "Young Researchers 2024-SoE" Program funded by the Italian Ministry of University and Research (MUR); and the project "Exploring User Behavior in Virtual Worlds and Its Relation to the Metaverse" funded by Sapienza University of Rome (RP123188ED29427D).

REFERENCES

- [1] G. Venkatadri, A. Andreou, Y. Liu, A. Mislove, K. P. Gummadi, P. Loiseau, and O. Goga, "Privacy risks with facebook’s pii-based targeting: Auditing a data broker’s advertising interface," in *2018 IEEE Symposium on Security and Privacy (SP)*. IEEE, 2018, pp. 89–107.
- [2] X. Amatriain, "Mining large streams of user data for personalized recommendations," *ACM SIGKDD Explorations Newsletter*, vol. 14, no. 2, pp. 37–48, 2013.
- [3] D. H. Stern, R. Herbrich, and T. Graepel, "Matchbox: large scale online bayesian recommendations," in *Proceedings of the 18th international conference on World wide web*, 2009, pp. 111–120.
- [4] H. Xu, S. Hao, A. Sari, and H. Wang, "Privacy risk assessment on email tracking," in *IEEE INFOCOM 2018-IEEE Conference on Computer Communications*. IEEE, 2018, pp. 2519–2527.
- [5] H. Hu, P. Peng, and G. Wang, "Characterizing pixel tracking through the lens of disposable email services," in *2019 IEEE Symposium on Security and Privacy (SP)*. IEEE, 2019, pp. 365–379.

⁴<https://www.gmass.co/blog/tracking-pixel-blockers/>

- [6] J. Ruohonen and V. Leppänen, “Invisible pixels are dead, long live invisible pixels!” in *Proceedings of the 2018 Workshop on Privacy in the Electronic Society*, 2018, pp. 28–32.
- [7] P. Eckersley, “How unique is your web browser?” in *International Symposium on Privacy Enhancing Technologies Symposium*. Springer, 2010, pp. 1–18.
- [8] P. Laperdrix, W. Rudametkin, and B. Baudry, “Beauty and the beast: Diverting modern web browsers to build unique browser fingerprints,” in *2016 IEEE Symposium on Security and Privacy (SP)*. IEEE, 2016, pp. 878–894.
- [9] G. Merzdovnik, M. Huber, D. Buhov, N. Nikiforakis, S. Neuner, M. Schmiedecker, and E. Weippl, “Block me if you can: A large-scale study of tracker-blocking tools,” in *2017 IEEE European Symposium on Security and Privacy (EuroS&P)*. IEEE, 2017, pp. 319–333.
- [10] J. Haupt, B. Bender, B. Fabian, and S. Lessmann, “Robust identification of email tracking: A machine learning approach,” *European Journal of Operational Research*, vol. 271, no. 1, pp. 341–356, 2018.
- [11] EasyList, “Easyprivacy list,” <https://easylist.to/easylist/easyprivacy.txt>.
- [12] —, “Easylist list,” <https://easylist.to/easylist/easylist.txt>.
- [13] OneClickLab, “ugly email trackers,” <https://github.com/OneClickLab/ugly-email-trackers/blob/master/list.txt>.
- [14] —, “ugly email extension,” <https://uglyemail.com/>.
- [15] uBlockOrigin, “ublock origin list,” <https://ublockorigin.github.io/uAssets/filters/privacy.txt>.
- [16] —, “ublock origin - an efficient blocker for chromium and firefox.” <https://github.com/gorhill/uBlock>.
- [17] StevenBlack, “Unified hosts file with base extensions,” <https://raw.githubusercontent.com/StevenBlack/hosts/master/hosts>.
- [18] N. Bielova, A. Legout, N. Sarafijanovic-Djukic *et al.*, “Missed by filter lists: Detecting unknown third-party trackers with invisible pixels,” *Proceedings on Privacy Enhancing Technologies*, 2020.
- [19] M. A. Bashir, S. Arshad, W. Robertson, and C. Wilson, “Tracing information flows between ad exchanges using retargeted ads,” in *25th USENIX Security Symposium (USENIX Security 16)*, 2016, pp. 481–496.
- [20] B. Liu, A. Sheth, U. Weinsberg, J. Chandrashekar, and R. Govindan, “Adreveal: Improving transparency into online targeted advertising,” in *Proceedings of the Twelfth ACM Workshop on Hot Topics in Networks*, 2013, pp. 1–7.
- [21] B. Krishnamurthy and C. E. Wills, “Generating a privacy footprint on the internet,” in *Proceedings of the 6th ACM SIGCOMM conference on Internet measurement*, 2006, pp. 65–70.
- [22] B. Krishnamurthy and C. Wills, “Privacy diffusion on the web: a longitudinal perspective,” in *Proceedings of the 18th international conference on World wide web*, 2009, pp. 541–550.
- [23] S. Englehardt and A. Narayanan, “Online tracking: A 1-million-site measurement and analysis,” in *Proceedings of the 2016 ACM SIGSAC conference on computer and communications security*, 2016, pp. 1388–1401.
- [24] A. Lerner, A. K. Simpson, T. Kohno, and F. Roesner, “Internet jones and the raiders of the lost trackers: An archaeological study of web tracking from 1996 to 2016,” in *25th USENIX Security Symposium (USENIX Security 16)*, 2016.
- [25] S. Englehardt, D. Reisman, C. Eubank, P. Zimmerman, J. Mayer, A. Narayanan, and E. W. Felten, “Cookies that give you away: The surveillance implications of web tracking,” in *Proceedings of the 24th International Conference on World Wide Web*, 2015, pp. 289–299.
- [26] P. Bekos, P. Papadopoulos, E. P. Markatos, and N. Kourtellis, “The hitchhiker’s guide to facebook web tracking with invisible pixels and click ids,” in *Proceedings of the ACM Web Conference 2023*, 2023, pp. 2132–2143.
- [27] L. Trampert, D. Weber, L. Gerlach, C. Rossow, and M. Schwarz, “Cascading spy sheets: Exploiting the complexity of modern css for email and browser fingerprinting,” 2025.
- [28] S. Englehardt, J. Han, and A. Narayanan, “I never signed up for this! privacy implications of email tracking,” *Proceedings on Privacy Enhancing Technologies*, 2018.
- [29] B. Fabian, B. Bender, and L. Weimann, “E-mail tracking in online marketing-methods, detection, and usage,” 2015.
- [30] B. Fabian, B. Bender, B. Hesseldieck, J. Haupt, and S. Lessmann, “Enterprise-grade protection against e-mail tracking,” *Information Systems*, vol. 97, p. 101702, 2021.
- [31] M. Maass, S. Schwär, and M. Hollick, “Towards transparency in email tracking,” in *Privacy Technologies and Policy: 7th Annual Privacy Forum, APF 2019, Rome, Italy, June 13–14, 2019, Proceedings 7*. Springer, 2019, pp. 18–27.
- [32] A. Chand, N. Nikiforakis, and P. Vadrevu, “Doubly dangerous: Evading phishing reporting systems by leveraging email tracking techniques,” in *34th USENIX Security Symposium (USENIX Security 25)*, 2025, pp. 3181–3200.
- [33] Alexa, “Alexa top sites,” <http://www.alexa.com/>, 2022.
- [34] A. B. O. CIRCULATIONS, “Highest circulated dailies, weeklies & magazines amongst member publications (across languages),” [https://www.auditbureau.org/files/JD%202022%20Highest%20Circulated%20\(across%20languages\).pdf](https://www.auditbureau.org/files/JD%202022%20Highest%20Circulated%20(across%20languages).pdf).
- [35] GMass, “How we get around tracking pixel blockers,” <https://www.gmass.co/blog/tracking-pixel-blockers/>.
- [36] Oberlo, “Most used email clients worldwide in 2024,” <https://www.oberlo.com/statistics/most-used-email-clients>.
- [37] Mailchimp, “What are the most used email service providers?” <https://mailchimp.com/en/resources/most-used-email-service-providers/>.
- [38] I. Litmus Software, “The state of email client market share,” <https://www.litmus.com/email-client-market-share>.
- [39] ProtonMail, “There are now over 100 million proton accounts,” <https://proton.me/blog/proton-100-million-accounts>.
- [40] Apple, “Mail privacy protection & privacy,” <https://www.apple.com/legal/privacy/data/en/mail-privacy-protection/>.
- [41] Mailchimp, “Activity and reports,” <https://mailchimp.com/developer/transactional/docs/activity-reports/#custom-tracking-domains>.
- [42] A. Plus, “How to write filters,” <https://help.adblockplus.org/hc/en-us/articles/360062733293-How-to-write-filters#allowlist>.
- [43] L. Bufalieri, M. La Morgia, A. Mei, and J. Stefa, “Gdpr: when the right to access personal data becomes a threat,” in *2020 IEEE International Conference on Web Services (ICWS)*. IEEE, 2020, pp. 75–83.

APPENDIX A

We conduct our experiments on a machine running Ubuntu 24.04 LTS. The tested browsers and desktop email client versions are detailed in Tab. 7.

TABLE 7: Browsers and desktop email client versions used in our experiments.

Client	Version
Google Chrome	137.0.7151.119
Mozilla Firefox	139.0.4
Microsoft Edge	137.0.3296.52
Brave	1.79.118
Safari	18.5
Mozilla Thunderbird	139.0.2
Proton Mail	1.8.0
Outlook	20250606009.12
Apple Mail	15.4

Hash and Encoding Schemes

MD2, MD4, MD5, RIPEMD, SHA1, SHA224, SHA256, SHA384, SHA512, SHA3-224, SHA3-256, SHA3-384, SHA3-512, BLAKE2b, BLAKE2s, CRC32, Adler32, MurmurHash3-32, MurmurHash3-64, MurmurHash3-128 Base16 (B16), Base32 (B32), Base64 (B64), Base85 (B85), URL Encoding Gzip, Zlib, Bzip2 (BZ2), Entity Encoding

TABLE 8: List of hash and encoding schemes used to detect the recipient’s email address within the image URL.

