Revisiting Email Forwarding Security under the Authenticated Received Chain Protocol

Chenkai Wang
University of Illinois at Urbana-Champaign
USA
chenkai3@illinois.edu

Gang Wang
University of Illinois at Urbana-Champaign
USA
gangw@illinois.edu

ABSTRACT

Email authentication protocols such as SPF, DKIM, and DMARC are used to detect spoofing attacks, but they face key challenges when handling email forwarding scenarios. Recently in 2019, a new Authenticated Received Chain (ARC) protocol was introduced to support mail forwarding applications to preserve the authentication records. After 2 years, it is still not well understood how ARC is implemented, deployed, and configured in practice. In this paper, we perform an empirical analysis on ARC usage and examine how it affects spoofing detection decisions on popular email provides that support ARC. After analyzing an email dataset of 600K messages, we show that ARC is not yet widely adopted, but it starts to attract adoption from major email providers (e.g., Gmail, Outlook). Our controlled experiment shows that most email providers' ARC implementations are done correctly. However, some email providers (Zoho) have misinterpreted the meaning of ARC results, which can be exploited by spoofing attacks. Finally, we empirically investigate forwarding-based "Hide My Email" services offered by iOS 15 and Firefox, and show their implementations break ARC and can be leveraged by attackers to launch more successful spoofing attacks against otherwise well-configured email receivers (e.g., Gmail).

CCS CONCEPTS

• **Security and privacy → Security protocols**.

KEYWORDS

Email Forwarding Security; Spoofing Attack; ARC

ACM Reference Format:
Chenkai Wang and Gang Wang. 2022. Revisiting Email Forwarding Security under the Authenticated Received Chain Protocol. In *Proceedings of the ACM Web Conference 2022 (WWW '22), April 25–29, 2022, Virtual Event, Lyon, France.* ACM, New York, NY, USA, 9 pages. https://doi.org/10.1145/3485447.3512228

1 INTRODUCTION

Email spoofing is commonly used in phishing and social engineering attacks where the attacker impersonates the sender address of a trusted entity [16, 24]. To prevent and detect email spoofing, various email authentication protocols (e.g., SMTP extensions) have been proposed and standardized in the past decades including SPF [11], DKIM [5], and DMARC [12].

While SPF, DKIM, and DMARC can work together to detect spoofing, they still face challenges to handle *mail forwarding* scenarios. Forwarding is a key mechanism to enable applications like mailing lists. More recently, to protect the privacy of personal email addresses, *forwarding-based* email relay services are introduced. This includes the most recent "Hide My Email" feature in iOS 15, which allows users to use a random iCloud alias to register online accounts without revealing their true email addresses [2]. During forwarding, emails are sent through one or multiple forwarders while preserving the original sender's information. During this process, however, key authentication information cannot be carried over, and thus it often leads to authentication failures [9].

To preserve the authentication records during mail forwarding, a new protocol called Authenticated Received Chain (ARC) [1] was published under RFC8617 in July 2019. ARC allows an intermediate mail server to sign the message's original authentication results so that even when SPF or DKIM fails at the receiver end, the receiver can still check the chain of authentication records to determine whether the message can be accepted. While it has been two years after the initial introduction of ARC, little is known about how ARC is adopted, implemented and configured in practice. Most existing studies have been focused on the more established protocols such as SPF, DKIM, and DMARC [6, 8, 10, 21, 23], and this paper seeks to fill in the gaps.

Our Questions. We focus on email forwarding scenarios and perform an empirical analysis of the real-world ARC implementations. We seek to answer the following questions. First, how widely are ARC adopted among email providers? Second, is the ARC protocol correctly implemented and deployed? Third, whether and how do email providers use ARC information to make spoofing detection decisions? To answer these questions, we gather and analyze email datasets, and design controlled experiments to test real-world email services as well as the recently introduced "Hide My Email" services. Our analysis leads to several key findings.

Analysis and Findings. First, we find the ARC is not widely adopted yet. Unlike SPF/DMARC, the adoption of ARC is not directly measurable using public DNS records [10, 23]. Instead, we work with an industry collaborator to analyze a dataset of 674,564 email headers (collected in 2020 and 2021) to look for signs of ARC adoption. In 2020, out of 10,740 unique email domains analyzed, ARC domain keys are only found in 6 sender/forwarder domains. The observation is similar in the 2021 data. Then we analyze popular public email providers and find that Gmail, Outlook, Zoho, Fastmail, and Pobox have adopted ARC. As major email providers such as Gmail start to adopt ARC, we do not rule out the possibility that ARC's adoption rate would increase in the future.

Second, we find that ARC results have different impacts on different email providers for spoofing detection. Specifically, we design and run a series of controlled experiments by sending spoofing emails to our own accounts within Gmail, Outlook, Fastmail, and Zoho. We emulate the scenario where an attacker aims to spoof a target domain that is protected by DMARC. By forging an ARC chain (with authentication results set to "pass"), we examine how email providers would make decisions. We find that Gmail responded correctly under all conditions; Fastmail and Outlook do not always follow the DMARC policy but they are also not influenced by ARC. On Zoho, however, we observe that a seemingly valid ARC chain (from an untrusted party) can help spoofing emails to get into their inbox and remove warnings.

Third, during our experiment, we do not find major implementation errors in the ARC protocol in Gmail. However, we find that Outlook, Zoho and Fastmail do not seal the DKIM signature in ARC sets when sending/forwarding emails (against RFC8617 recommendations [1]). After analyzing a few other open-sourced ARC implementations, we find several issues in the existing OpenARC codebase and the ARC module of Mailman3 (a mailing list manager).

Finally, we analyze email relay services including iCloud "Hide My Email" and Firefox Relay, and find that they implement the forwarding in a way that would break the ARC chain. Their current designs do not perform careful authentication and also take away the ability of checking authenticity of the original senders from the end receivers. Through controlled experiments, we show that attackers can leverage their relay services to conduct spoofing attacks against Gmail, successfully sending spoofing emails into Gmail's inbox (that would be otherwise rejected).

Contributions. In summary, we have three key contributions:

- First, we analyze a large email dataset and popular email providers to understand ARC adoption in practice.
- Second, we run controlled experiments to examine how email providers use ARC results to make spoofing detection decisions. The results reveal potentially incorrect interpretations of ARC results from email providers.
- Third, we analyze "Hide My Email" services to understand how their email forwarding are implemented. We show the current implementations can be used to assist spoofing attacks.

As ARC is going through the trial phase in practice, the main contribution of our work is to empirically examine the gaps between the protocol specification and real-world executions, and point out common mistakes when using ARC results to make decisions.

2 BACKGROUND AND MOTIVATION

2.1 Email Spoofing Attack

Simple Mail Transfer Protocol (SMTP) [18] is the standard protocol for email delivery over the internet. Introduced in 1982, SMTP does not include any built-in security features. As a result, adversaries can spoof an arbitrary sender address to send emails on their behalf. Suppose the impersonation target is "`info@bank.com`", there are two ways to implement spoofing. First, the adversary can modify the "`MAIL FROM`" field in the SMTP protocol and use "`info@bank.com`" as the sender address. This address, by default, will also be inserted as the "`Return-Path`" and "`From`" in the email header. "`Return-Path`" determines the destination address of the reply message, and the "`From`" address will be eventually displayed on the user interface (UI) of the receiver's email client. Second, alternatively, the adversary can directly modify the "`From`" field in the email header [20].

2.2 Security Extensions against Spoofing

In the past decades, security extensions have been introduced to defend against email spoofing attacks:

SPF. Sender Policy Framework (SPF) [11] uses IP addresses to authenticate email senders. Using the same example above, to prevent spoofing, administrators of "`bank.com`" can publish a DNS record to declare which IP addresses are valid to send emails on their behalf. When the receiving server receives an email whose "`MAIL FROM`" address is claimed to be "`bank.com`", the server can query `bank.com`'s DNS record to check if the declared IP address matches.

DKIM. DomainKeys Identified Mail (DKIM) [5] is a public-key-based protocol. The composer of the email will sign the (selected) header fields and the message body with its domain key (e.g., a private key associated with "`bank.com`"). The generated signature will be attached to the email header. When a receiving server receives this email, the server can query the DNS to obtain the public key of "`bank.com`" to verify the signatures. Compared with SPF (which only focuses on the authenticity of the sender), DKIM additionally provides an integrity check on the email message.

DMARC. Domain-based Message Authentication (DMARC) [12] is designed to work together with SPF and/or DKIM to holistically verify the sender authenticity, handle authentication failures, and report results back to senders. One of the key contributions of DMARC is that it fixes a spoofing vulnerability of SPF and DKIM. More specifically, SPF only inspects "`MAIL FROM`" (for email delivery) but does not consider the "`From`" field in the header (the address that will be displayed to users on the UI). As a result, the attacker may use an "`attacker.com`" (under its control) as the "`MAIL FROM`" to pass SPF, while setting the "From" field as "`info@bank.com`" to fool users [9]. Similarly, to bypass DKIM, the attacker can use its own domain key associated with "`attacker.com`" to sign the message while setting the "From" field as "`info@bank.com`" [9]. DMARC fundamentally solves this problem by requiring all domain identifiers to be aligned, including the domain name of the DKIM key, "`MAIL FROM`", and the "`From`" in the header.

Recent measurement shows the adoption rates of SPF, DKIM, and DMARC protocols have been increasing, but there are still a large number of domain names that have not adopted (or only partially adopted) them [6, 8, 10, 15, 23].

2.3 Email Forwarding and ARC

Unfortunately, existing security extensions are not working well with email forwarding [9]. Email forwarding is a key mechanism to enable mailing list functions, and it is also used by individuals who need to automatically forward emails from one email provider to another (for convenience or privacy). Figure 1 illustrates an email forwarding process. Suppose `f.com` is a mailing list, when

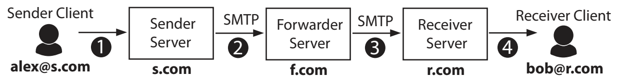

Figure 1: The email forwarding process.

alex@s.com sends an email to the mailing list, f.com will automatically forward the message to all its subscribers (bob@r.com is one of the subscribers). While the receiver Bob still sees "alex@s.com" is displayed as the sender, it is more difficult to verify the sender authenticity.

First, SPF is broken by mail forwarding. From the receiver r.com's perspective, the immediate sending server is f.com whose IP address will not match with the original sender s.com's SPF record. Forcefully modifying the "Return-Path" to be "f.com" can pass SPF but will cause DMARC failure due to identifier misalignment.

Second, DKIM has a chance to be broken too as mailing lists often need to modify the original message. A common modification is to add a footer to include the name of the mailing list and/or a link for unsubscription. Some mailing lists also help to convert all hidden hyperlinks into displayed links in the email body. Due to the modification, the original DKIM signature becomes invalid and thus DMARC will fail.

Authenticated Received Chain (ARC) Overview. ARC protocol (RFC8617) [1] was recently published in 2019. ARC allows an intermediate mail server (a mailing list or a forwarding service) to sign the message's original authentication records. In this way, the receiving server can validate the message even when the email's SPF or DKIM fails due to the intermediate server's processing. ARC creates a chain of authentication records when an email message is forwarded by multiple intermediate servers, as shown in Figure 2. During each forwarding hop, an "ARC Set" is added to the chain. This is done by (1) validating the records in the previous ARC Sets (*validation* process), and (2) signing any new changes introduced to the email before sealing the entire chain (*sealing* process).

ARC Set is the building block of the chain. Each ARC Set has an *instance number* (i.e., sequence number) which increases by 1 after each ARC-participating hop. It also contains three header fields:

- **ARC-Authentication-Results (AAR)** header holds the authentication results (SPF, DKIM, DMARC) produced by the current server at the message arrival time. This allows the final receiver to trace the authentication results at each hop, especially the first hop that involves the original sender.
- **ARC-Message-Signature (AMS)** header contains the *signature* of selected email headers (excluding ARC headers) and the message body signed by the current server.
- **ARC-Seal (AS)** header contains a signature for all "ARC-" header fields as a whole (to preserve the integrity of the ARC Chain). It also contains the current "chain validation status" ("cv").

In the following, we use Figure 2 to explain how the i_{th} hop server performs the *validation* process and the *sealing* process.

ARC Validation Process. When the i_{th} hop server receives an incoming message, it needs to validate the ARC Sets from previous hops. It first collects all of the previous ARC Sets on the chain, and checks their ARC-Seal (AS) headers. If any of the previous AS headers has the chain validation status (cv) marked as "fail", it means the chain has already failed, and thus there is no point

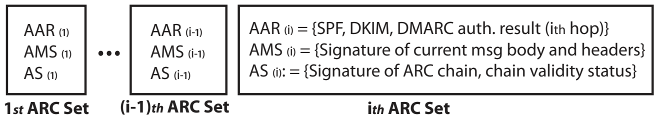
AAR (1)
AMS (1)
AS (1)
•••
AAR (i-1)
AMS (i-1)
AS (i-1)
AAR (i) = {SPF, DKIM, DMARC auth. result (ith hop)}
AMS (i) = {Signature of current msg body and headers}
AS (i): = {Signature of ARC chain, chain validity status}
1st ARC Set
(i-1)th ARC Set
ith ARC Set

Figure 2: ARC chain structure. Each ARC set contains an ARC-Authentication-Results (AAR), an ARC-Message-Signature (AMS), and an ARC-Seal (AS).

to continue the process. Also, if there is any missing ARC Set on the chain (based on instance numbers), the validation process will return a "fail" status immediately. Finally, the server will check the ARC-Message-Signature (AMS) from the most recent $(i-1)_{th}$ hop to verify the integrity of the non-ARC headers and the email body. If all of the above integrity checks are successful, the current chain validation status is set to "cv=pass" ("fail" otherwise).

Sealing Process. If this server (e.g., a mailing list) needs to modify the email message (e.g., adding a footer), the modification must be done *before* the sealing process. To perform sealing, the server first prepares the ARC-Authentication-Results (AAR) header by storing the current i_{th} hop's authentication results for SPF, DKIM, and DMARC to its AAR header. Second, the server prepares the ARC-Message-Signature (AMS) header by signing the email body and other non-ARC headers. Third, the server prepares the ARC-Seal (AS) header by collecting every ARC-* header in the current email (including the new AAR and AMS generated above), sorting them in an ordered list, and signing its hash with this i_{th} hop server's private key. Also, this ARC-Seal (AS) header includes the current chain validation status (cv) produced by the *validation process* above.

Checking ARC at the Receiver End. From the final receiver's point of view, if SPF, DKIM, or DMARC fails at the last hop (e.g, due to message modification during forwarding), the receiver can then check the ARC Sets. It can use the ARC chain to trace back to the SPF/DKIM/DMARC authentication results at the *first hop* when authenticating the original sender. Also, email modifications can be analyzed along the chain using the AMS headers. As pointed out by the protocol specification and other analysis articles [1, 13], ARC has an important assumption, that is, the forwarders are trusted (i.e., they follow the ARC protocol). In this paper, we will examine how the ARC protocol is interpreted and implemented in practice.

3 PROGRESS OF ADOPTION

We start by exploring the real-world adoption of ARC among email providers. Determining whether a given mail domain has adopted ARC is challenging using existing measurement methods. Prior works have examined the adoption rates of SPF and DMARC by scanning their records in the public DNS [8, 10, 15, 23]. In comparison, DKIM's adoption rate is more difficult to measure because querying a domain's DKIM key requires knowing the "selector" information (which is not public knowledge). A recent work overcomes this challenge by guessing common selectors used for DKIM keys [23]. ARC also hosts its domain key using DNS with a similar mechanism like DKIM (ARC may even use the same DKIM key). This creates a problem: even if we find out a given domain hosts a domain key on DNS, it does not mean the domain has adopted ARC (i.e., it could be a DKIM key). As such, to examine whether a

Count	Domain Key	Key Length
94,580	arcselector9901._domainkey.microsoft.com	2048 bit
38	arc-20160816._domainkey.google.com	1024 bit
7	arc-20200618._domainkey.improvmx.com	1024 bit
4	arc-201807._domainkey.one.com	1024 bit
2	zohoarc._domainkey.zohomail360.com	1024 bit
1	zohoarc._domainkey.zohomail.eu	1024 bit
1	key201809._domainkey.1-hostingservice.com	1024 bit

Table 1: ARC keys found in 407,357 email messages in the 2020 dataset. The keys are ranked by the number of associated messages.

given domain has adopted ARC, we need to collect email messages sent/forwarded by this domain and analyze the header information. We perform such analyses with (1) an email dataset obtained from an industry partner (with IRB approval); and (2) public email services where we can register accounts to send/receive emails.

3.1 Analyzing an Email Dataset

We analyze an dataset (674,564 email headers) working with our collaborator at *mailsac.com* (a disposable email service). The dataset contains two parts collected from 2020 and 2021, respectively.

Dataset 2020. This dataset contains the *header information* for 407,357 incoming email messages sent/forwarded to *mailsac.com* in November 2020. Among the 407,357 email headers, we observe 10,740 unique sender domain names (that send/forward emails to `mailsac.com`). Among them, 360,542 (88.5%) emails contain DKIM keys and only 94,633 (23.2%) emails contain ARC keys. After removing duplicated keys, we find a total 6,189 unique DKIM keys, and only 7 unique keys for ARC. This suggests that ARC adoption rate is far lower than that of DKIM.

Table 1 shows the 7 ARC keys we find, used by 6 different service providers, including Microsoft Outlook, Google Gmail, Zoho, one.com (a web hosting service), improvmx.com (an email alias service), and e-goi.com (a marketing service). The vast majority of ARC-enabled messages are either sent from or forwarded through Microsoft (Outlook).

Dataset 2021. We use a more recent dataset collected in November 2021 to double check the results. This set contains 267,207 message headers, from which we find 11,277 unique sender domain names. Among them, 88.7% of the messages contain DKIM keys, and 13.4% of the messages contain ARC keys. While our dataset is not necessarily representative (e.g., it is collected from a single email service), the results can at least indicate that ARC is not widely adopted.

3.2 Analyzing Email Providers

To complement the above analysis, we further examined the ARC adoption in popular email providers. We considered 13 email providers that allow us to create accounts for running experiments (see Table 4 in the Appendix). Most of them are free/public services (e.g., Gmail, Yahoo Mail) and two are paid services (i.e., Fastmail, Pobox). Given an email provider, we first register an account with it. Then we try to send and receive ARC-enabled messages using this account. We also try to set up email forwarding within the email provider. Finally, we check whether they perform ARC sealing and validation based on the header information of the received messages.

We find that most of the email services do not have any support for ARC (see Table 4). For free services, only Outlook, Gmail, and Zoho support both ARC validation and sealing during forwarding. Note that Gmail does not support ARC sealing when sending out emails. Fastmail is a commercial email service and has full ARC support. While Pobox supports ARC, it is a forwarding-only service for users to create email aliases and does not provide email inbox services. Overall, the result is consistent with Section 3.1 that ARC is not yet widely supported yet.

3.3 Open-source ARC Implementations

We suspect that the quality of readily available open-source ARC implementations may have affected the adoption of ARC. As such, we analyzed existing open-source ARC implementations that act as plugins (or "milters") in mail transfer agents (MTA) suits and those integrated in popular Mailing List Managers. We find that certain projects (e.g., OpenARC and Mailman3) have implementation errors or design choices against the RFC recommendations. Due to the space limit, we report the more detailed results in Appendix A.

3.4 Result Summary

Our investigation of an email dataset and popular email providers shows that most email providers have not adopted ARC (except for Gmail, Outlook, Zoho, and Fastmail). We also find implementation errors in certain open-source ARC implementations.

4 ARC IMPACT ON SPOOFING

Given ARC has been adopted by popular email services (e.g., Gmail, Outlook, Fastmail, and Zoho), we run experiments to understand how ARC is used, and how ARC results affect their decisions on spoofing emails. In these experiments, the adversary pretends to be a forwarding server to interact with these email providers (who act as receivers). By fabricating the ARC sets (i.e., the authentication records), the adversary tries to convince the email providers that the spoofing email is forwarded from the original sender and is authentic. Recall that ARC does not address the issue of untrusted forwarders [1, 13]. We use this experiment to check whether real-world email providers have interpreted ARC correctly.

4.1 Experiment Setup

Suppose we try to spoof a target domain name (e.g., `t.com`), the adversary sets up a mail forwarding server `f.com` to send a spoofing email to the testing email provider (e.g., Gmail). If `t.com` has already set up SPF/DKIM and DMARC, attempts of directly spoofing `t.com` will be detected by Gmail. Instead, the adversary tries to use ARC to convince the receiver that any authentication failure at the receiver end is due to forwarding, and the original authentication record in the ARC Set (AAR) shows that the original sender `t.com` has been correctly verified. In this attack, there is no real sender sending emails to the attacker at `f.com`—the attacker will simply spoof the email and fabricate the ARC Sets. Our goal is to understand whether the email receiver (Gmail) would take the ARC results into consideration when making spoofing detection decisions.

To run this experiment, we first register accounts at the testing email providers. All of the emails are sent to these accounts under our control. Then we test different setups for the choices of the

Setup	**Spoof DMARC="none"**				**Spoof DMARC="quarantine"**				**Spoof DMARC="reject"**			
SPF, DKIM, ARC	zoho	fastmail	gmail	outlook	zoho	fastmail	gmail	outlook	zoho	fastmail	gmail	outlook
SPF=0, DKIM=0, ARC=0	[○]	○	◐	◐	●	◐	[◐]	◐	●	◐	●	◐
SPF=0, DKIM=0, ARC=1	○	○	◐	◐	○	◐	[◐]	◐	●	◐	●	◐
SPF=0, DKIM=1, ARC=0	[○]	○	◐	◐	●	◐	[◐]	◐	●	◐	●	◐
SPF=0, DKIM=1, ARC=1	○	○	◐	◐	○	◐	[◐]	◐	●	◐	●	◐
SPF=1, DKIM=0, ARC=0	○	○	◐	◐	●	◐	[◐]	◐	●	◐	●	◐
SPF=1, DKIM=0, ARC=1	○	○	◐	◐	○	◐	[◐]	◐	●	◐	●	◐
SPF=1, DKIM=1, ARC=0	○	○	◐	◐	●	◐	[◐]	◐	●	◐	●	◐
SPF=1, DKIM=1, ARC=1	○	○	◐	◐	○	◐	[◐]	◐	●	◐	●	◐

Table 2: The spoofing experiment results. ○=inbox; ◐=spam/junk folder; ●=blocked/discarded; [ ] means a warning message is shown on the email client. The spoofing target domains each has different DMARC policies: "none", "quarantine", and "reject". For the experiment setup, DMARC will fail for all cases (due to spoofing). SPF is set to either pass (1) or fail (0). DKIM is set to either pass (1) or fail (0). For ARC, the attacker can choose to not include any ARC set (0), or add an ARC Set with falsified authentication records (1).

target domain `t.com`, and the SPF and DKIM configurations during forwarding. Since we focus on spoofing detection (which only considers sender identity), we use benign email content for our testing messages (i.e., the content is considered "benign" by all tested email providers; verified by non-spoofing experiments).

Target Domain Names. Prior works already show that domains that have not adopted SPF/DKIM/DMARC can be successfully spoofed [6, 10, 23]. In this experiment, we only consider target domains (`t.com`) that have adopted SPF+DMARC, DKIM+DMARC, or both. `t.com` may set different DMARC policies to instruct the receiver how to handle an email *when the authentication fails*. For example, "none" means `t.com` does not have any instruction and it is up to the receiver to make a decision; "quarantine" means the receiver should put the email into a spam folder; "reject" means the receiver should directly discard the email when the authentication fails. In our experiment (see Table 2), we select three domain names with different DMARC policies: `illinois.edu` ("none"), `usenix.org` ("quarantine"), and `nicehash.com` ("reject"). Here, we choose to spoof real-world domain names (instead of freshly registered domain names) to mimic realistic attack scenarios[1].

SPF Setting. For SPF, we test two conditions where SPF is either pass (1) or fail (0). First, if the attacker uses the attacker's `f.com` as the "`MAIL FROM`", then SPF can pass, but DMARC will fail due to identifier misalignment (see Section 2.3). Second, if the attacker modifies the "`MAIL FROM`" to be aligned with the "`From`" field in the email header (both are `t.com`), then SPF will fail due to `t.com`'s SPF record (see Section 2.2). In both cases, DMARC will fail.

DKIM Setting. We have two conditions for DKIM where DKIM is either pass (1) or fail (0). First, if the attacker inserts its own DKIM signature by signing the message, DKIM will pass but DMARC will fail due to identifier misalignment (see Section 2.3). Second, if the attacker does not insert any DKIM signature, the original DKIM will fail (due to spoofing). In both cases, DMARC will fail.

ARC Setting. For ARC, we have two conditions. Condition ARC=0 means the attacker does not include any ARC record (baseline). Condition ARC=1 means the attacker adds one ARC Set (i=1) in the chain sealed with a falsified authentication record "spf=pass, dkim=pass, dmarc=pass". This falsified record tries to convince the receiver that the original sender (`t.com`) is correctly authenticated before the forwarding.

[1]During our pilot experiments, we also tested with freshly registered domain names, and the conclusion was the same.

4.2 Experiment Results

Table 2 shows the results. We consider Gmail, Zoho, Fastmail, and Outlook for this experiment. Pobox is not applicable because it does not have an inbox (forwarding only). Under all scenarios, the spoofing target `t.com` has a DMARC record (with different DMARC policies). By manually checking the results, we confirm that all four email providers have correctly mark DMARC "fail" under these scenarios. Even so, Table 2 shows that these email providers handle the spoofing emails differently, and ARC has influenced the decision of certain providers.

DMARC policy="none". When the target domain's DMARC policy is "none", it means the email receivers can make their own decisions on how to handle the emails (that have failed DMARC). In this case, we observe that Gmail and Outlook put the spoofing emails into the spam folder. However, Fastmail and Zoho put the spoofing email into user's inbox. More importantly, on Zoho, the falsified ARC set (ARC=1) has boosted Zoho's trust toward the spoofing email. For example, when SPF fails (SPF=0), Zoho will display a warning message on the email client UI to warn users that the email sender is not verified. However, by using an ARC Set, we can successfully remove the warning message.

A closer inspection shows that Zoho has correctly marked all the decisions for SPF and DKIM. Also, Zoho has correctly marked the DMARC authentication results as "fail" for all cases[2]. A likely explanation is that Zoho has checked the ARC chain and looked into the ARC Set (i=1) for the authentication records of the original sender (`t.com`). Recall that the attacker has falsified this record as "spf=pass, dkim=pass, dmarc=pass", which may have influenced Zoho's decision (i.e., remove the warning message).

DMARC policy="quarantine". When the target domain has a DMARC policy of "quarantine", it means the receiver is supposed

[2]This observation is different from a recent analysis performed in 2020 [21], which shows Zoho has an ARC implementation error that leads to incorrect DMARC "pass" for forwarded emails. As of October 2021, we do not observe such an implementation error anymore as DMARC is marked as "fail" correctly by Zoho. It is likely that Zoho has fixed that error in the past year.

to put the message into the spam folder when authentication fails. We find that Fastmail, Outlook, and Gmail have followed the policy. Gmail even added a warning message explaining why the email is put into the spam folder. However, Zoho again shows the influence of ARC. Without ARC (ARC=0), Zoho will directly drop the message without even putting it into the spam folder due to the failed DMARC. However, with ARC (ARC=1), even with a failed DMARC, Zoho consistently puts the spoofing email into user inbox *without showing the warning*.

DMARC policy="reject". When the target domain has a DMARC policy of "reject", the receiver is expected to directly drop the message when authentication fails. Our results show that Gmail has respected the reject policy. However, Fastmail and Outlook still keep the spoofing email in the spam folder (i.e., not rigorously following the policy). Interestingly, Zoho also correctly follow the reject policy this time. Based on the error code we received, it seems that DMARC rejection happened at the SMTP receiving stage (without reaching the next phase to check ARC yet).

Other Problems Observed. We find a separate problem with the ARC sealing process in Zoho, Outlook, and Fastmail (which is not related to the above experiment). When these services are acting as the *sender/forwarder*, their seals do not include the DKIM signature in AMS. This is a similar problem with Mailman3 described in Appendix A. As DKIM signatures are not sealed by ARC, their integrity cannot be protected. Gmail is the only service that seals with the DKIM signature header in AMS.

4.3 Result Summary

Overall, our experiment shows that in most cases, a spoofing email with a falsified ARC Set does not necessarily change the decisions of email providers at the receiving end. However, for Zoho, even though its DMARC authentication is correctly performed, the presence of ARC="pass" still increased their trust toward spoofing emails for certain conditions (e.g., when DMARC policy is not "reject"). While this is not an implementation error, it appears that Zoho has misinterpreted the meaning of ARC="pass". Fundamentally, ARC cannot prevent malicious forwarders from deviating from the ARC protocol and inserting fabricated authentication records. Instead, ARC results should only be used to audit a failed DMARC for *trusted* forwarders (e.g., a mailing list on the receivers' allowlist, a high reputation forwarding service). Further discussion is presented in Section 6.

5 "HIDE MY EMAIL" SERVICES

Finally, we investigate an email forwarding scenario for privacy-preserving purposes. In September 2021, iOS 15 introduces a new privacy feature called "Hide My Email" (or HME), which is developed based on the concept of mail forwarding [2]. It allows users to generate random email aliases (e.g., `random_name@icloud.com`) to register accounts with online services (e.g., online social networks) without revealing their true personal email address (e.g., `true_name@gmail.com`). Whenever the email alias receives a message, iCloud will automatically forward the message to the user's true email address (e.g., at Gmail). This also allows users to create a large number of email alias to reduce the linkability of their identities across online platforms (and even password dumps). A similar service is offered by Firefox too called Firefox Relay [7].

In the following, we conduct a simple experiment to investigate how these two email relay services implement the forwarding process, and whether ARC/DKIM is correctly used.

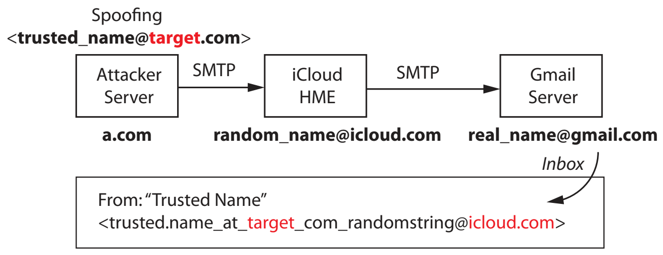

Figure 3: Spoofing attack via iCloud's "Hide My Email" (HME) service. The spoofing emails can reach the inbox of the receiver at Gmail.com.

5.1 Attacking via iCloud HME

We use iCloud's Hide My Email (HME) as an example to explain our experiment process. Figure 3 shows the setup. We set up an attacker mail server at `a.com` and spoof a target identity under the target domain, e.g., "`trusted_name@target.com`". The spoofing emails are sent to the victim users whose real email at Gmail is hidden behind iCloud's HME. The attacker will simply send the spoofing email to the victim's alias at `random_name@icloud.com`, which will be automatically forwarded to `real_name@gmail.com`. The configurations of the attacker's SPF, DKIM, ARC, and the spoofing target domains are the same with the previous experiment in Section 4.1.

More Successful Attacks. Table 3 shows the experiment results. As a comparison baseline, we directly run the spoofing attack without using iCloud HME to send emails to the Gmail receiver (same attack described in Section 4.1). Gmail has either put the spoofing emails into the spam folder or directly discarded the emails without showing them to users.

When we run the attack via iCloud HME, the spoofing attack becomes more effective. For instance, when the target domain's DMARC policy is "none", all emails get into the Gmail inbox. When the target domain's DMARC policy is "reject", we find that the spoofing emails can no longer reach the inbox. A closer examination shows that the emails are dropped by iCloud (which never reached Gmail) as iCloud follows the target domain's DMARC policy. The interesting part is when the target domain's DMARC policy is "quarantine". Since iCloud HME is a forwarding service, it does not store the message and thus cannot accommodate "quarantine". The email needs to be either forwarded or rejected by iCloud HME. As a result, iCloud decides to forward it through even though the DMARC fails. Eventually, all of these emails enter the Gmail inbox. Also, the warnings that exist in the baseline experiment are removed.

Broken Chain of ARC. We examine the email forwarding process of iCloud HME, and have two key observations.

Setup	**Direct Spoofing (baseline)**			**Attack via iCloud HME**			**Attack via Firefox Relay**		
SPF, DKIM, ARC	none	quarantine	reject	none	quarantine	reject	none	quarantine	reject
SPF=0, DKIM=0, ARC=0	◐	[◐]	●	○	○	●	○	○	○
SPF=0, DKIM=0, ARC=1	◐	[◐]	●	○	○	●	○	○	○
SPF=0, DKIM=1, ARC=0	◐	[◐]	●	○	○	●	○	○	○
SPF=0, DKIM=1, ARC=1	◐	[◐]	●	○	○	●	○	○	○
SPF=1, DKIM=0, ARC=0	◐	[◐]	●	○	○	●	○	○	○
SPF=1, DKIM=0, ARC=1	◐	[◐]	●	○	○	●	○	○	○
SPF=1, DKIM=1, ARC=0	◐	[◐]	●	○	○	●	○	○	○
SPF=1, DKIM=1, ARC=1	◐	[◐]	●	○	○	●	○	○	○

Table 3: The spoofing attack results against Gmail as the receiver. We compare the baseline attack (directly spoofing) with the attack via iCloud's "Hide My Email" (HME) and the attack via Firefox Relay. ○=inbox; ◐=spam/junk folder; ●=blocked/discarded; [] means a warning message is shown on the email client. The spoofing target domains each has different DMARC policies: "none", "quarantine", and "reject". For the experiment setup, DMARC will fail for all cases (due to spoofing). SPF is set to either pass (1) or fail (0). DKIM is set to either pass (1) or fail (0). For ARC, the attacker can choose to not include any ARC set (0), or add an ARC Set with falsified authentication records (1).

First, iCloud does not perform any ARC operation (no validation or sealing). For emails that have eventually reached Gmail, the ARC Set inserted by the attacker is still preserved after passing through iCloud. Second, iCloud makes changes to the email. As shown in Figure 3, instead of forwarding the received message as it is, iCloud has replaced the original "`From`" in the header which is "`trusted_name@target.com`" with a reformatted sender address as shown in Figure 3. Note that the new "`From`" address has the domain name of `icloud.com`. The advantage is that this forces identifier alignment between "`MAIL FROM`" and "`From`" which helps the email to bypass DMARC. However, this also breaks the ARC Set: without re-sealing the modified header, the existing AMS signature is no longer valid (due to the above header modification). As a result, when Gmail receives this email, it marks "arc=fail" due to the broken AMS signature.

5.2 Attacking via Firefox Relay

We run a similar experiment for Firefox Relay. Table 3 shows even stronger results.

First, regardless of the target domain's DMARC policies, all spoofing emails enter Gmail's inbox without raising any warnings. The result indicates that Firefox Relay does not perform any authentication (e.g., DMARC) or take any actions other than forwarding all the emails through.

Second, like iCloud HME, Firefox Relay also breaks ARC, but in a different way. Firefox Relay completely removes all authentication-related headers (including ARC Sets) from the original email and converts the original plain text email into an HTML format. Then Firefox sends the newly formatted email to the receiver (Gmail). The email is essentially recomposed, and the "`From`" field is formatted as "`'trusted_name@.target.com [via Relay]' <noreply@relay.firefox.com>`" In this way, the original sender address is formatted as the displayed "sender name" and the new "`From`" domain name is set to `relay.firefox.com`. This again helps to align the SPF identifiers to pass DMARC.

5.3 Result Summary

In summary, both iCloud HME and Firefox Relay have made a similar decision to modify the emails. This improves deliverability of the messages because: (1) the IP addresses of iCloud and Firefox have a high reputation, and (2) changing the "`From`" domain to the domain names of the relay helps to pass DMARC to get the emails accepted. The problem is, this modification can be exploited by attackers. As shown in our experiment, spoofing emails that were previously blocked by Gmail are now mostly accepted without raising warnings. Also, iCloud modifies emails without re-sealing the ARC headers and Firefox completely removes all authentication headers. This takes away the opportunity from the final receiver (e.g., Gmail) to check the authenticity of the original email.

As a relay service, high deliverability is highly desired. The question is, can they maintain high deliverability while preserving the chain of authentication records? We believe this is a largely reachable goal. (1) iCloud/Firefox relay should perform their own authentication and drop emails with failed DMARC (when sender policy="reject"). (2) The relay should perform ARC validation and sealing to maintain the integrity of the ARC chain. (3) By default, the relay should not modify the original email[3].

For such a relay, if the incoming email has DKIM, because of (3), the DKIM signature will still be valid when the email reaches the receiver, which makes DMARC pass (i.e., no harm on deliverability). If the incoming email does not have DKIM but has SPF, the above recommendations might hurt deliverability when (a) the sender's DMARC policy is "reject"/"quarantine" and (b) SPF="pass" at the relay. Under this condition, the relay may consider modifying "`From`" to ensure deliverability. In this case, because SPF is "pass" at the relay (i.e., email authenticity is verified), modifying "`From`" does not introduce additional risks to the receiver. If the incoming email does not have DKIM or SPF, the above recommendations also do not hurt deliverability.

6 DISCUSSION

6.1 Ethical Considerations

We have taken active steps to ensure research ethics. Our study has an approved IRB protocol. All spoofing emails sent in our experiments have been sent to accounts under our control. There

[3]If the relay modifies the original email, it is suboptimal. This is because the receivers would lose the opportunity to verify the original sender themselves, and have to put extra trust to the relay—trusting it can *correctly* verify the original sender.

are no other users' accounts involved in these experiments. We are in the process of sharing our research findings to related parties (e.g., Zoho, iCloud, Firefox, Mailman3) to help improve their implementations.

6.2 Benefits and Problems of ARC

Overall, our results show that major email providers (e.g., Gmail and Outlook) have correctly implemented the ARC protocol. However, some providers (Zoho) might have misinterpreted the meaning of "ARC=pass" (their ARC validation is implemented correctly). Fundamentally, ARC is a chain of signed authentication results, which allows the email receiver to audit potential DMARC failures. However, there is an implicit assumption that the receiver needs to trust the "good faith" of forwarders, i.e., good-faith forwarders may make legitimate changes to the email (e.g., a mailing list adding a footer) but they should always follow the ARC protocol. This also means ARC cannot prevent malicious forwarders from deviating from the protocol and adding fabricated authentication records onto the chain.

When forwarders are *trusted*, ARC can help to save a legitimately forwarded email when the forwarding breaks its DMARC. For example, when a *trusted* mailing list forwards an email to a Gmail user. Due to the mailing list's modification of the email (e.g., adding a footer), the original DKIM fails and thus DMARC fails. In this case, because the mailing list is known to be legitimate, Gmail can choose to further check the ARC. If ARC=pass, Gmail then can use the ARC chain to check the authentication record of the original sender (prior forwarding) and check where the modification is made that causes the DMARC failure. Gmail may let this legitimate email into the inbox if the modification is done by this trusted mailing list.

However, if this email is forwarded by an unknown forwarder, the receiver (Gmail) should not trust the email simply because of "ARC=pass" or other assertions carried in the chain. Otherwise, this decision could be exploited by attackers to facilitate their spoofing attacks (as shown in Section 4.2).

The above discussion leads to an interesting dilemma. ARC should be considered only when the receiver trusts the forwarders are in good faith (i.e., honest). This means ARC needs to be used in combination with allowlists or domain reputation systems. On one hand, if a forwarder is already on the allowlist of the receiver, ARC may not add much value to increase email deliverability. On the other hand, if a forwarder is not currently trusted, ARC should not be used to establish such trust. This dilemma could lead to a relatively narrow applicable scope of ARC and hurt its adoption.

Lastly, for security-sensitive domains, we recommend always using DKIM and setting DMARC policy to "reject". This minimizes spoofing attack's success rate even when ARC is used to amplify the attack.

6.3 Limitations and Future Work

Our work has some limitations. First, the email dataset used in Section 3.1 is not necessarily representative. Indeed, finding a large and recent email dataset is challenging given its sensitive nature. As such, we only make a conservative conclusion that ARC is not widely adopted as DKIM yet. Second, our experiments in Section 4 and Section 5 only use a small number of target domains which is a limitation. We tried to limit our experiment scale to avoid stressing the receiving email services. Third, our analysis on open-source ARC implementations is focused on those can be easily found via a quick Google search. Future work may examine a larger collection of ARC implementations. Finally, while email user interfaces (UI) are not a main focus of this paper, we observe that email providers display forwarded emails differently. Our future work will explore how users perceive the forwarded (spoofing) emails and whether they can interpret the information on the UIs correctly.

7 RELATED WORK

DNS-based email authentication protocols have been studied in the past with a focus on SPF, DKIM, and DMARC. Researchers have studied their adoption rates in practice [6, 8–10, 15, 23], security flaws rooted in the inconsistent interpretation/parsing of messages [4], and their inability to handle subdomain spoofing [19, 21]. Our study is different from existing works as we focus on the new ARC protocol and email forwarding scenarios to understand the impact of ARC on spoofing detection.

The most related work to ours is [21] where the authors studied many spoofing techniques but only briefly mentioned ARC. In our paper, we focus on ARC to dive deeper. The key differences are three-fold. First, we have different threat models. Prior work [21] assumes a legitimate forwarder contains vulnerabilities that allow an attacker to forward (spoofing) emails to an address that the attackers does not own, whereas our paper assumes the attacker sets up its own forwarding server. Second, the problems identified in [21] (in Zoho and Outlook) have been fixed after a year (confirmed in our experiment). The problems discovered in our experiment are new. Finally, we analyzed the "Hide-My-Email" forwarding services of iCloud and Firefox, which is a new contribution.

8 CONCLUSION

In this paper, we perform an empirical analysis on ARC adoption and implementations in practice. Our analysis is based on an email dataset of 600K messages, which shows ARC is not yet widely adopted (in comparison with DKIM). Our controlled experiment shows that most email providers' ARC implementations are done correctly. However, some email provider (i.e., Zoho) has misinterpreted the meaning of ARC results, which makes spoofing emails with ARC Sets more successful. Finally, we empirically investigate forwarding-based "Hide My Email" services from iCloud and Firefox and show their implementation breaks the ARC chain and can be used to run spoofing attacks against strong defenses. As ARC starts to get popularized, we hope our results can help practitioners to avoid common mistakes.

ACKNOWLEDGMENTS

We thank Jeff Parrish for preparing and sharing the email dataset for our project. This work was supported in part by NSF grant CNS-2030521, and Jump ARCHES endowment through the Health Care Engineering Systems Center. Any opinions, findings, conclusions, or recommendations expressed in this material are those of the authors and do not necessarily reflect the views of any funding agencies.

REFERENCES

[1] K. Andersen, B. Long, S. Blank, and M. Kucherawy. 2019. The Authenticated Received Chain (ARC) Protocol. RFC8617. https://datatracker.ietf.org/doc/html/rfc8617.

[2] Apple. 2021. What is Hide My Email? https://support.apple.com/en-us/HT210425.

[3] Marc Bradshaw. 2021. Fastmail Authentication Milter. https://github.com/fastmail/authentication_milter.

[4] Jianjun Chen, Vern Paxson, and Jian Jiang. 2020. Composition Kills: A Case Study of Email Sender Authentication. In *Proc. of USENIX Security.*

[5] D. Crocker, T. Hansen, and M. Kucherawy. 2011. DomainKeys Identified Mail (DKIM) Signatures. RFC6376. https://tools.ietf.org/html/rfc6376.

[6] Zakir Durumeric, David Adrian, Ariana Mirian, James Kasten, Elie Bursztein, Nicolas Lidzborski, Kurt Thomas, Vijay Eranti, Michael Bailey, and J. Alex Halderman. 2015. Neither Snow Nor Rain Nor MITM: An Empirical Analysis of Email Delivery Security. In *Proc. of IMC.*

[7] Firefox. 2021. Firefox Relay. https://relay.firefox.com/.

[8] Ian D. Foster, Jon Larson, Max Masich, Alex C. Snoeren, Stefan Savage, and Kirill Levchenko. 2015. Security by Any Other Name: On the Effectiveness of Provider Based Email Security. In *Proc. of CCS.*

[9] Hang Hu, Peng Peng, and Gang Wang. 2018. Towards Understanding the Adoption of Anti-Spoofing Protocols in Email Systems. In *Proc. of SecDev.*

[10] Hang Hu and Gang Wang. 2018. End-to-End Measurements of Email Spoofing Attacks. In *Proc. of USENIX Security.*

[11] S. Kitterman. 2014. Sender Policy Framework (SPF). RFC7208. https://tools.ietf.org/html/rfc7208.

[12] M. Kucherawy and E. Zwicky. 2015. Domain-based Message Authentication, Reporting, and Conformance (DMARC). RFC7489. https://tools.ietf.org/html/rfc7489.

[13] John Levine. 2015. What's ARC? https://circleid.com/posts/20151028_what_is_authenticated_received_chain_arc.

[14] Mailman3. 2021. Mailman3 Mailing List Manager. https://docs.mailman3.org/en/latest/.

[15] Sourena Maroofi, Maciej Korczynski, Arnold Hölzel, and Andrzej Duda. 2021. Adoption of Email Anti-Spoofing Schemes: A Large Scale Analysis. *IEEE Trans. Netw. Serv. Manag.* 18, 3 (2021), 3184–3196.

[16] Daniela Oliveira, Harold Rocha, Huizi Yang, Donovan Ellis, Sandeep Dommaraju, Melis Muradoglu, Devon Weir, Adam Soliman, Tian Lin, and Natalie Ebner. 2017. Dissecting Spear Phishing Emails for Older vs Young Adults: On the Interplay of Weapons of Influence and Life Domains in Predicting Susceptibility to Phishing. In *Proc. of CHI.*

[17] OpenARC. 2021. The Trusted Domain Project: OpenARC. https://github.com/trusteddomainproject/OpenARC.

[18] J. B. Postel. 1982. Simple Mail Transfer Protocol (SMTP). RFC821. https://tools.ietf.org/html/rfc821.

[19] Florian Quinkert, Dennis Tatang, and Thorsten Holz. 2021. Digging Deeper: An Analysis of Domain Impersonation in the Lower DNS Hierarchy. In *Proc. of DIMVA.*

[20] P. Resnick. 2001. Internet Message Format *(RFC5321).* https://www.ietf.org/rfc/rfc2822.txt.

[21] Kaiwen Shen, Chuhan Wang, Minglei Guo, Xiaofeng Zheng, Chaoyi Lu, Baojun Liu, Yuxuan Zhao, Shuang Hao, Haixin Duan, Qingfeng Pan, and Min Yang. 2021. Weak Links in Authentication Chains: A Large-scale Analysis of Email Sender Spoofing Attacks. In *Proc. of USENIX Security.*

[22] Sympa. 2021. Sympa Mailing List Manager. https://www.sympa.org/.

[23] Dennis Tatang, Florian Zettl, and Thorsten Holz. 2021. The Evolution of DNS-based Email Authentication: Measuring Adoption and Finding Flaws. In *Proc. of RAID.*

[24] TrendMicro. 2021. White Paper by Osterman Research: How to Reduce the Risk of Phishing and Ransomware. https://resources.trendmicro.com/rs/945-CXD-062/images/Reduce-Phishing-Ransomware_Trend-Micro.pdf.

A ARC IMPLEMENTATION ANALYSIS

We briefly analyze existing open-source ARC implementations that can be found online, and explore potential problems with them.

ARC Implementation as Milters. Popular mail transfer agents (MTA) suits such as Postfix and Sendmail use plugin software, called "milters", to run security checks and other extended functionalities like spam filtering. For example, for DKIM and DMARC, the commonly used milters are OpenDMARC and OpenDKIM. For ARC, we find an "OpenARC" implementation [17] by the same Trusted Domain Project (TDP) that also developed OpenDKIM and OpenDMARC. However, by running OpenARC on our mail server, we find the implementation has various errors including broken and/or mismatched signatures (confirmed by sending the OpenARC sealed emails to Outlook and Gmail). Also, the OpenARC code has not been actively maintained since 2018.

Another open-source ARC implementation is an authentication milter developed by Fastmail that contains ARC support [3]. Compared to OpenARC, this project is still actively maintained as of 2021. It is currently used by the Fastmail email service (tested later in Section 4; no obvious implementation error is spotted).

Integration with Mailing List Managers. Mailing lists are based on email forwarding, and thus we investigate open-source Mailing List Managers to explore their ARC integration. We find that Mailman3 [14] has included ARC. Their overall implementation is compliant with the RFC8617 [1], with *one exception.* We find Mailman3 does not handle DKIM signing and thus its ARC Set does not include the DKIM signature header. Even if the outgoing MTA generates a DKIM header later, it would not be sealed by the ARC Set. Not sealing the DKIM header makes it difficult to keep track of its integrity, which is against the RFC8617 recommendations [1]. Since it is the Mailing List Manager (Mailman3) that modifies the email (e.g., adding the footer), it is better for Mailman3 to handle the DKIM signature and seal it within the ARC Set.

Another open-source Mailing List Manager Sympa [22] also integrates ARC. However, due to outdated documentations, we did not manage to start a runnable instance for Sympa.

Domain	Validate	Seal in Forwarding	Seal in Sending
outlook.com	**Yes**	**Yes**	**Yes**
zoho.com	**Yes**	**Yes**	**Yes**
fastmail.com*	**Yes**	**Yes**	**Yes**
gmail.com	**Yes**	**Yes**	No
pobox.com*	**Yes**	**Yes**	N/A
mail.ru	No	No	No
yandex.com	No	No	No
protonmail.com	No	No	No
aol.com	No	No	No
yahoo.com	No	No	No
qq.com	No	No	No
163.com	No	No	No
icloud.com	No	No	No

Table 4: Email providers and their adoption status of ARC. *Commercial email services.