SE424: Distributed Systems	**Recitation 7a**
Semester 2 5786	**3 May 2026**
Lecturer: Michael J. May	**Kinneret College**

Email Authentication: SPF and DKIM

In the previous recitation we explored how `dig` can be used to query DNS records. In this recitation, we will use `dig` to inspect two popular email authentication mechanisms: the **Sender Policy Framework (SPF)** and **DomainKeys Identified Mail (DKIM)**. Both mechanisms publish their data as plain DNS TXT records, so it's all visible via DNS.

1 Background: Why Email Needs Authentication

The Simple Mail Transfer Protocol (SMTP) was designed in 1982 without a mechanism for verifying the sender's identity. Anyone could forge the `From:` address in an email, something you'll see every day in phishing and spam today. To address the problem, two DNS-based standards were developed:

SPF (Sender Policy Framework) A domain owner publishes a TXT record declaring a list of IP addresses that are authorized to send email on behalf of the domain. Receiving mail servers check the record against the IP address of the connecting server.

DKIM (DomainKeys Identified Mail) The sending mail server signs outgoing messages with a private key. The corresponding public key is published in DNS. Receiving servers retrieve the public key and verify the signature, confirming the message was not altered.

A third standard, **DMARC** (Domain-based Message Authentication, Reporting & Conformance), builds on top of SPF and DKIM to specify what a receiving server should *do* when either check fails. We will look at DMARC briefly at the end of this recitation.

2 SPF: Sender Policy Framework

2.1 How SPF Works

When a receiving mail server accepts a connection from a remote host claiming to deliver mail from `user@example.com`, it performs the following steps:

1. Extract the domain (`example.com`) from the sender address in the message **envelope** (a set of metadata fields that accompany every email message).
2. Query DNS for a TXT record at `example.com` beginning with `v=spf1`.
3. Check whether the connecting IP address is authorized by that record.
4. Accept, soft-fail, or reject the message according to the policy.

2.2 SPF Record Anatomy

A typical SPF record looks like the following:

```
"v=spf1 include:_spf.google.com include:sendgrid.net ip4:203.0.113.10 ~all"
```

The tokens in the record have the following meanings:

`v=spf1` The SPF version tag. Always present and always `spf1`.

`include:_spf.google.com` Recursively include all IP ranges published in Google's own SPF record.

`include:sendgrid.net` Also authorize SendGrid's mail servers to send on behalf of this domain.

`ip4:203.0.113.10` Explicitly authorize a single IPv4 address.

`~all` **Soft fail** — any IP not matched by the above mechanisms is considered suspicious but is not hard-rejected.

The `all` mechanism is always placed last and acts as a catch-all policy. The qualifier symbol before `all` determines how non-matching senders are treated:

Qualifier	Symbol	Meaning
Pass	`+all`	Allow all senders (defeats SPF, so it's almost never correct)
Soft Fail	`~all`	Fail but do not reject, flag the message as suspicious
Fail	`-all`	Hard reject any sender not listed
Neutral	`?all`	No policy statement

2.3 Querying SPF Records with `dig`

1. Use `dig` to retrieve the TXT records for `google.com` and identify the SPF record.

```
dig google.com TXT +short
```

2. Google's SPF record delegates to `_spf.google.com` using an `include:` mechanism. Follow the chain by querying that subdomain.

```
dig _spf.google.com TXT +short
```

3. Look up the SPF records for several other well-known domains. For each one, note the `all` qualifier and any `include:` directives.

```
dig github.com TXT +short | grep spf
dig microsoft.com TXT +short | grep spf
dig amazon.com TXT +short | grep spf
```

4. What happens if a domain does not publish an SPF record at all? What should a receiving mail server do?

3 DKIM: DomainKeys Identified Mail

3.1 How DKIM Works

DKIM uses asymmetric cryptography (RSA or Ed25519). The flow is as follows:

1. The sending mail server signs the outgoing message body and selected headers using a **private key** held on the server.
2. A `DKIM-Signature:` header containing the cryptographic signature is added to the message before delivery.
3. The **public key** is published in DNS as a TXT record under a special subdomain.
4. The receiving mail server queries DNS to retrieve the public key and uses it to verify the signature. A valid signature confirms that the message body (and the signed headers) were not altered after the message left the sending server.

3.2 DKIM Record Location

Unlike SPF, DKIM records are not placed at the root of the domain. They are published at a subdomain with the following structure:

`<selector>._domainkey.<domain>`

selector A label chosen by the domain owner (e.g., `google`, `s1`, `20240101`). Multiple selectors allow the domain to rotate keys without breaking messages already in transit.

`_domainkey` A fixed namespace reserved for DKIM records.

domain The signing domain (e.g., `google.com`).

For example: `google._domainkey.google.com`

3.3 DKIM Record Anatomy

A DKIM TXT record looks like the following:

```
"v=DKIM1; k=rsa; p=MIIBIjANBgkqhkiG9w0BAQEFAAOCAQ8AMIIBCgKCAQEA..."
```

The tags in the record have the following meanings:

`v=DKIM1` The DKIM version. Always `DKIM1`.

`k=rsa` The key algorithm. Either `rsa` or `ed25519`.

`p=...` The Base64-encoded public key. If this field is **empty** (`p=`), the key has been **revoked** and signatures using this selector must be rejected.

`h=sha256` Acceptable hash algorithms (optional).

`t=s` Optional flags, e.g., `s` means subdomains may not use this key.

`n=...` A note field about the key

3.4 Querying DKIM Records with `dig`

1. Gmail uses date-stamped selectors to support key rotation. Retrieve Gmail's selector for a few years ago:

   ```
   dig 20230601._domainkey.gmail.com TXT +short
   ```

2. Retrieve the DKIM public keys for several third-party email providers.

   ```
   dig k1._domainkey.mailchimp.com TXT +short
   dig s1._domainkey.sendgrid.net TXT +short
   dig s1024._domainkey.yahoo.com TXT +short
   dig s2048._domainkey.yahoo.com TXT +short
   ```

3. The responses from the DNS records are in fact digital certificates encoded in base64 encoding. You can try to parse them using a bunch of bash commands.

   ```
   dig s2048._domainkey.yahoo.com TXT +short | tr -d '" \n' | grep -oP '(?<=p=)[^;]+' | tr
       -d ' ' | base64 -d | openssl rsa -pubin -inform DER -text -noout
   dig s1._domainkey.sendgrid.net TXT +short | tr -d '" \n' | grep -oP '(?<=p=)[^;]+' | tr
       -d ' ' | base64 -d | openssl rsa -pubin -inform DER -text -noout
   ```

4. Use the `+noall +answer` flags to display the TTL values on DKIM records. Why might the TTL matter for DKIM?

```
dig 20230601._domainkey.google.com TXT +noall +answer
```

5. Open up an email that you received from Gmail and read its headers to find the DKIM key that signed the message. Which value do you see?
6. Try the following key lookup:

   ```
   dig 20161025._domainkey.google.com TXT +short
   ```

   What values do you see? What do the values mean?
7. If the `p=` field in a DKIM record is empty, what does that mean? How would you detect this with `dig`?

4 DMARC — Tying SPF and DKIM Together

DMARC (Domain-based Message Authentication, Reporting & Conformance) is published as a TXT record at `_dmarc.<domain>`. It tells receiving servers what to do when SPF or DKIM checks fail, and where to send aggregate reports.

```
dig _dmarc.google.com TXT +short
```

A typical DMARC record looks like this:

```
"v=DMARC1; p=reject; rua=mailto:mailauth-reports@google.com; pct=100"
```

`v=DMARC1` The DMARC version tag.

`p=reject` The policy to apply when alignment fails. Options are `none` (monitor only), `quarantine` (send to spam), or `reject` (discard the message).

`sp=reject` If an unsigned mail came from a subdomain, just reject it. You can put the same values for the p field here.

`rua=mailto:...` Address to which receiving servers send aggregate authentication reports.

`ruf=mailto:..` Address to send forensic reports to.

`pct=100` Apply this policy to 100% of messages. Values below 100 allow a gradual rollout.

`fo=` When to generate a forensic report. `0` if both SPF and DKIM fail. `1` if either fail. `d` if DKIM fails regardless of SPF. `s` if SPF fails regardless of DKIM. You can combine values with a :, for instance `fo=d:s` which is the same as `1`.

`adkim=...` How strict to align the DKIM signature. `r` means it's relaxed and DKIM keys can work on subdomains. `s` means it's strict and subdomains can only be matched if written explicitly.

`aspf=...` How strict the alignment of SPF is. `r` is relaxed and subdomains are allowed. `s` is strict, so they are not allowed unless written explicitly.

4.1 What to do

1. Query the DMARC record for some domains. What policy does each domain enforce?

   ```
   dig _dmarc.google.com TXT +short
   dig _dmarc.github.com TXT +short
   dig _dmarc.amazon.com TXT +short
   dig _dmarc.kinneret.ac.il TXT +short
   ```

2. Try querying the DMARC record for a domain that you believe has weak or missing email authentication. What do you observe?
3. Try querying `walla.co.il`'s DMARC policy. What policy does it enforce?

5 Full Domain Audit

Combine everything above to perform a complete email authentication audit of a single domain. The following shell script automates the process. Save it as `email-auth-check.sh`, make it executable, and run it against several domains.

```
#!/bin/bash
# email-auth-check.sh
# Usage: ./email-auth-check.sh <domain>

DOMAIN=$1
if [ -z "$DOMAIN" ]; then
  echo "Usage: $0 <domain>"
  exit 1
fi

echo "=============================="
echo " Email Auth Check: $DOMAIN"
echo "=============================="

echo ""
echo "--- MX Records ---"
dig "$DOMAIN" MX +short

echo ""
echo "--- SPF Record ---"
SPF=$(dig "$DOMAIN" TXT +short | grep "v=spf1")
if [ -z "$SPF" ]; then
  echo "[!] No SPF record found"
else
  echo "$SPF"
fi

echo ""
echo "--- DMARC Record ---"
DMARC=$(dig "_dmarc.$DOMAIN" TXT +short)
if [ -z "$DMARC" ]; then
  echo "[!] No DMARC record found"
else
  echo "$DMARC"
fi

echo ""
echo "--- Common DKIM Selectors ---"
for SELECTOR in default google s1 s2 mail dkim k1 20230601 20240101; do
  RESULT=$(dig "${SELECTOR}._domainkey.${DOMAIN}" TXT +short 2>/dev/null)
  if [ -n "$RESULT" ]; then
    echo "[+] Found selector: $SELECTOR"
    echo "    ${RESULT:0:80}..."
  fi
done

echo ""
echo "Done."
```

Run the script against the following domains and record your observations:

```
chmod +x email-auth-check.sh
./email-auth-check.sh github.com
./email-auth-check.sh amazon.com
./email-auth-check.sh kinneret.ac.il
./email-auth-check.sh walla.com
```

6 Summary

Mechanism	DNS Record Location	Type	Key Content
SPF	`domain.com`	TXT	`v=spf1 ... ~all`
DKIM	`<selector>._domainkey.domain.com`	TXT	`v=DKIM1; k=rsa; p=...`
DMARC	`_dmarc.domain.com`	TXT	`v=DMARC1; p=reject/quarantine/none`

SPF and DKIM are both stored as plain DNS TXT records. Any DNS client can query them with no special tooling. SPF declares *who* is permitted to send mail for a domain; DKIM proves that *what* was sent has not been altered in transit. DMARC ties both mechanisms together with an enforcement policy and a reporting channel. All three are visible directly in DNS, making `dig` a useful tool for auditing the email security posture of a domain.

