Watters & Jacobs, LLP
Litigation and Trial Law Firm
+1 (650) 542-0057

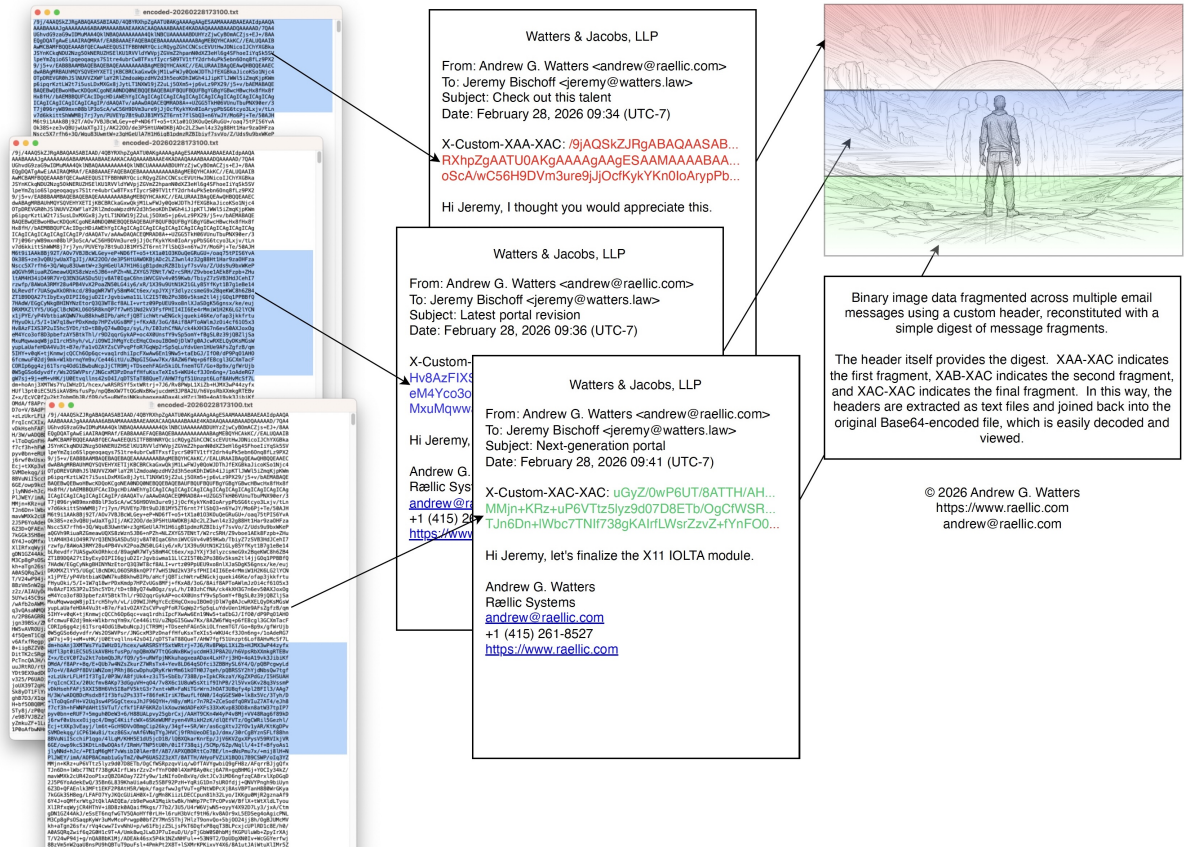
555 Twin Dolphin Dr., Ste. 135
Redwood City, CA 94065
info@watters.law

MOSAIC

An automated covert electronic distribution and payment system using a digest of reconstituted message fragments.

By: Andrew G. Watters[1] and Jeremy Bischoff[2]

March 7, 2026



Executive Summary

MOSAIC is a proof-of-concept system that demonstrates how arbitrary binary files (images, documents, executables), in particular CSAM, can be covertly distributed and

[1]Managing Partner, Watters & Jacobs, LLP. andrew@watters.law

[2]Software engineer, Watters & Jacobs, LLP. jeremy@watters.law

paid for through ordinary email traffic– without appearing as attachments, without triggering attachment-based filters, and without any visible indication in the email body that data is being transferred or payments are being made. A system like this, or one similar to it, could plausibly have been used by Jeffrey Epstein and his accomplices to distribute illicit content such as CSAM while evading conventional email monitoring.

Due to the ease of creating the system– essentially one day of shell scripting for a functional prototype– along with the convenience and utility of the system, the authors assess it is highly probable this method already exists in the wild. Law enforcement would need to have access to the raw source of targets' emails to determine whether any additional data is being sent– a difficult task without an already-pending investigation supported by probable cause.

While it is preferable to the sender to have their own email server in order to transfer files from a server-side queue, the sender does not actually have to have their own server– they only need access to a server that can send email from the command line[3]. The challenge is compounded by the fact that no one is looking for this type of traffic on port 25 (SMTP) or port 993 (IMAP with SSL/TLS), as it is not suspicious compared to the well-known Tor ports and VPNs that would drastically increase the target's detection profile. Finally, MOSAIC is nearly entirely automated, and is practically instantaneous, as well as convenient. As such, the concepts behind MOSAIC represent an important new avenue for investigators to detect the covert distribution of CSAM.

I. Technical Introduction

The core idea is straightforward. Every email message contains *headers*: structured lines of metadata that precede the message body. Standard headers include fields like `From`, `To`, `Date`, and `Subject`. The email standard (RFC 5322) also permits arbitrary *custom headers*, which are any header lines beginning with `X-`. Mail servers, spam filters, and email clients routinely ignore custom headers they do not recognize; they simply pass them through untouched.

MOSAIC exploits this by using custom headers to its advantage, following this high-level process that is enabled by standard command line utilities available on Linux and macOS– and that many Windows computers also have:

1. **Encode.** The input file is Base64-encoded, producing a stream of printable ASCII characters. This is the same encoding used by email attachments (MIME), but MOSAIC does not use the MIME attachment mechanism.
2. **Fragment.** The Base64 stream is split into N chunks of roughly equal size.
3. **Embed.** Each chunk is placed into a custom `X-Custom` header of a separate, otherwise ordinary-looking email message. The header name itself encodes the

[3]Virtually all Linux servers can send email from the command line with no additional configuration and with only built-in software. This is commonly done with automated system messages and notifications.

fragment's sequence number and the total fragment count.

4. **Transmit.** The emails are sent through any standard mail server (SMTP). To an observer or automated filter, they appear to be routine messages with unremarkable bodies.
5. **Reassemble.** The recipient (or an automated process) scans incoming emails for the custom headers, extracts the fragments, orders them by sequence number, concatenates the Base64 data, and decodes it back to the original binary file.
6. **Verify.** A SHA-256 digest embedded in each email's `X-Custom-Digest` header allows confirmation that the reassembled file is bit-for-bit identical to the original. (This step serves to validate the proof of concept rather than being essential to the distribution mechanism itself.)
7. **Invoice.** An arbitrary Bitcoin amount indicating what is being charged for the content appears in a `X-Custom-Invoice` header, so that the user sees how much the content costs. This permits the sender to keep track of the user's balance. This also could be triggered by a purchase order on a website that caused the content to be sent. Or the user could have a subscription that automatically sends them a certain amount of new CSAM per month and they can pay extra for more. Either way, this is all recorded in a blockchain ledger.

Because the payload never appears as an attachment and the email body can contain any plausible cover text, conventional detection methods (attachment scanning, file-type filtering, keyword analysis of message bodies) are ineffective. The data hides in a part of the message that virtually no monitoring tool inspects. The payment request is also in a custom header, which acts as an invoice for payment via Bitcoin or other cryptocurrency.

Header naming convention. MOSAIC identifies fragments using a structured header name of the form:

`X-Custom-<SEQ>-<TOTAL>`

where `SEQ` and `TOTAL` are three-character alphabetic codes. The encoding maps integers to letter pairs: $1 =$ `XAA`, $2 =$ `XAB`, ..., $26 =$ `XAZ`, $27 =$ `XBA`, and so on. For example, the header `X-Custom-XAC-XAH` identifies fragment 3 of 8. This convention allows up to 676 fragments (26×26) per transfer, which is sufficient for files of substantial size.

II. Demonstration Walkthrough

The `mosaic.sh` script includes a self-contained `demo` mode that runs the full encode–fragment–embed–reassemble–verify pipeline on any input file and produces a local directory of artifacts for inspection. The following walkthrough uses a small PNG test image (320×240 pixels, 1,718 bytes) to illustrate the process end to end.

Phase 1: File preparation. The demo begins by Base64-encoding the input image.

The 1,718-byte binary file becomes a stream of printable ASCII characters wrapped to 79-character lines, the same line length used by standard MIME encoding. A SHA-256 digest of the original file is computed and stored for later verification:

```
SHA256: 4c535f1caeb220037f8c0867de15760b08eb927c
        71f20006c8ffe072482d9a2f
```

Phase 2: Fragmentation. The Base64 stream is split into five chunks of approximately six lines each (the number of chunks is arbitrary and configurable). Each chunk becomes one fragment that will be embedded in a separate email.

Phase 3: Email embedding. For each fragment, MOSAIC generates a complete RFC 2822-compliant email message. The fragment data is placed in a custom header; the message body contains only ordinary cover text. Below is an abbreviated example of the first email's headers:

```
From: MOSAIC System <mosaic-sender@example.com>
To: <mosaic-receiver@example.com>
Subject: MOSAIC [1/5]
Date: Fri, 06 Mar 2026 14:45:33 -0800
Message-ID: <mosaic-1-5-177283...@mosaic.local>
MIME-Version: 1.0
Content-Type: text/plain; charset=utf-8
X-Custom-XAA-XAE: iVBORw0KGgoAAAANSUhEUgAAAUA...
X-Custom-Digest: 4c535f1caeb220037f8c0867de...
```

The header `X-Custom-XAA-XAE` indicates this is fragment 1 of 5. The header's value contains the Base64 data for this chunk. The `X-Custom-Digest` header carries the SHA-256 hash of the original file, providing an integrity check independent of any single fragment.

Phase 4: Extraction and reassembly. The decoder scans each `.eml` file in the target directory, searching for headers matching the pattern `X-Custom-X[A-Z][A-Z]-X[A-Z][A-Z]`. When a matching header is found, the decoder parses the sequence number and total count from the header name, extracts the Base64 value, and stores the fragment keyed by its sequence number. After all emails have been processed, the fragments are concatenated in order and Base64-decoded to produce the output file.

Phase 5: Verification. The SHA-256 digest of the reassembled file is compared against the digest embedded in the email headers:

```
Original SHA256:    4c535f1caeb220037f8c0867de15760b...
Reassembled SHA256: 4c535f1caeb220037f8c0867de15760b...
```

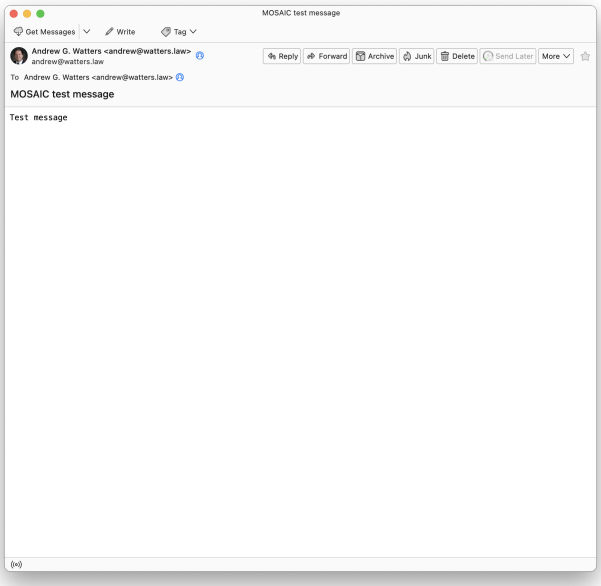
```
SHA256 match: YES
Integrity:    PASS
```

The hashes match, confirming that the reassembled PNG is bit-for-bit identical to the original. The file type is independently verified via the file's magic bytes. This demonstrates that the full pipeline (encoding, fragmentation, embedding into email headers, extraction, and reassembly) is lossless.

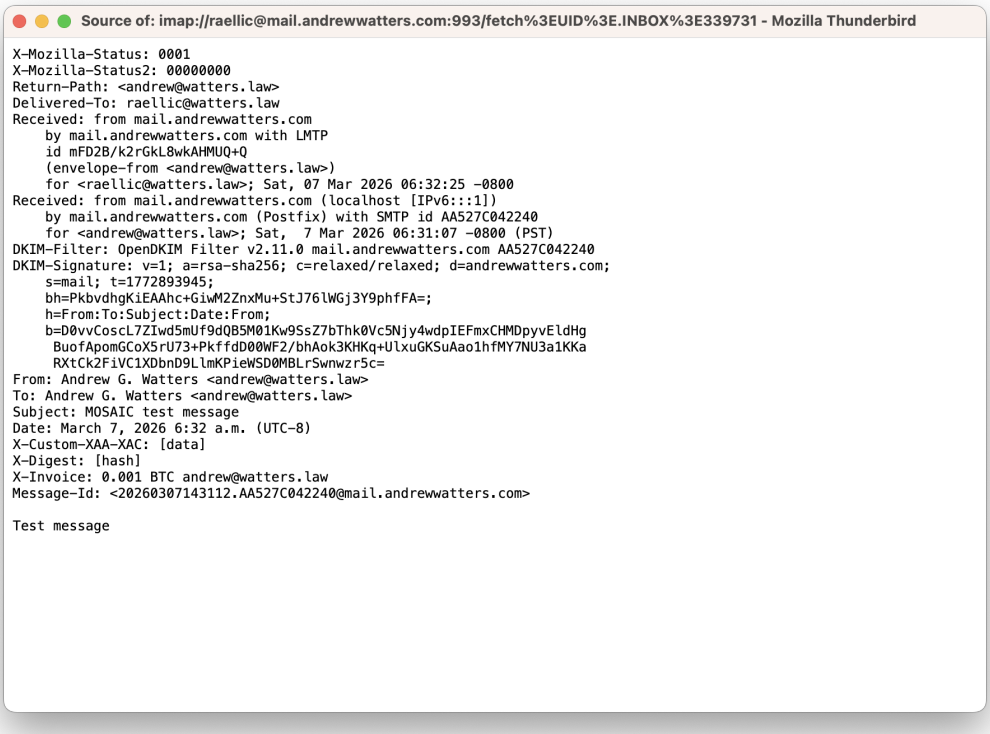
Practical Demonstration

This screenshot depicts a connection via a SSH tunnel to the authors' email server. The user connects to the email server from the command line via Telnet and manually sends an email by typing it into Telnet. The custom headers would be where the MOSAIC data goes,

and can either be typed in manually, copied/pasted, or sent through automated processes.



This screenshot shows the email as received by the user's desktop email program.



This screenshot shows the source of the email in question. As indicated, the headers are there and can be any arbitrary length.

Conclusion

The current mosaic.sh interactive shell script[4] is provided along with this white paper, and updates may be downloaded at the following link, as well:

https://www.watters.law/articles/attachments/mosaic.sh

The proof of concept proves that the covert distribution of CSAM through message fragments is entirely possible, easy, and convenient. It is highly probable that offenders are already using this technique to distribute CSAM.

The authors may be reached via email at andrew@watters.law, or by phone as indicated in the cover page.

[4] As of this date, the Bitcoin payment module is not included, but will be in the next version.