Network Working Group
Internet-Draft
Intended status: Experimental
Expires: 30 September 2026

A. Maurette
IUT R&T Bethune

29 March 2026

HMTFTP: HKDF-Derived TFTP with Optional AEAD Protection (v0.7) draft-maurette-hmtftp-06

Abstract

HMTFTP is a lightweight UDP file transfer protocol derived from TFTP. It preserves the TFTP transfer model (RRQ/WRQ, DATA, ACK, ERROR, and OACK option acknowledgment) and replaces text options with binary TLVs. HMTFTP defines optional AEAD protection for DATA payloads using HKDF-derived keys from a pre-shared key (PSK).

This document describes transfer behavior, extension processing rules, key derivation, nonce construction, and operational guidance for experimental implementations intended to facilitate interoperability.

This document requests IANA actions for a UDP service name/port and registries for TLV Types and Ciphersuites. Until assignment, TBD1 is used as the default UDP port value.

Status of This Memo

This Internet-Draft is submitted in full conformance with the provisions of BCP 78 and BCP 79.

Internet-Drafts are working documents of the Internet Engineering Task Force (IETF). Note that other groups may also distribute working documents as Internet-Drafts. The list of current Internet-Drafts is at https://datatracker.ietf.org/drafts/current/.

Internet-Drafts are draft documents valid for a maximum of six months and may be updated, replaced, or obsoleted by other documents at any time. It is inappropriate to use Internet-Drafts as reference material or to cite them other than as "work in progress."

This Internet-Draft will expire on 30 September 2026.

Copyright Notice

Table of Contents

1. Introduction

The Trivial File Transfer Protocol (TFTP) [RFC1350] provides a minimal UDP file transfer service but no built-in confidentiality or integrity protections.

HMTFTP preserves TFTP operational semantics while introducing:

* a binary TLV negotiation mechanism for RRQ, WRQ, and OACK;

* explicit extension processing rules (including critical TLVs);

* optional AEAD protection of DATA payloads.

HMTFTP is intended for controlled or managed environments where simple TFTP-like behavior remains desirable but optional integrity and confidentiality for transferred payloads are required.

2. Conventions and Requirements Language

The key words "MUST", "MUST NOT", "REQUIRED", "SHALL", "SHALL NOT", "SHOULD", "SHOULD NOT", "RECOMMENDED", "NOT RECOMMENDED", "MAY", and "OPTIONAL" in this document are to be interpreted as described in BCP 14, [RFC2119] [RFC8174] when, and only when, they appear in all capitals, as shown here.

* *PSK*: pre-shared key

* *AEAD*: authenticated encryption with associated data

* *AAD*: additional authenticated data

* *TID*: transfer identifier (UDP endpoint pair per RFC 1350)

3. Relationship to TFTP and Compatibility

3.1. Semantic Similarity with TFTP

HMTFTP reuses the TFTP message types and baseline semantics from [RFC1350]: RRQ, WRQ, DATA, ACK, ERROR, block numbering, stop-and-wait transfer, and end-of-file signaling by a short final DATA block.

HMTFTP also reuses the OACK concept from [RFC2347].

3.2. Deployment and Operational Differences

Compared to baseline TFTP, HMTFTP differs as follows:

* TLVs are binary and carried in RRQ, WRQ, and OACK.

* The default server port is TBD1 (configurable), not UDP/69.

* Optional AEAD protection MAY be negotiated for DATA payloads.

3.3. Limits of Direct Interoperability

Classic TFTP endpoints that do not understand HMTFTP TLVs will not reliably interoperate when TLVs are present. In particular, security negotiation (ENC_REQ/CNONCE/SNONCE/CIPHER) is HMTFTP-specific.

A client that requires security mode MUST set ENC_REQ as Critical and MUST abort if it is not acknowledged in OACK.

Fallback behavior "as TFTP" means message sequencing and transfer semantics follow [RFC1350]. It does not imply use of UDP/69: deployments can remain TFTP-like while using the configured HMTFTP service port.

4. Protocol Overview

HMTFTP uses TFTP packet types and augments negotiation with TLVs:

* Client sends RRQ or WRQ with optional TLVs.

* If TLVs are present and request is accepted, server replies with OACK; otherwise transfer follows RFC1350 sequencing.

* RRQ flow: client ACK(0), then server DATA(1), client ACK(1), ...

* WRQ flow: client DATA(1), server ACK(1), ...

If AEAD security mode is accepted, DATA payloads are carried as Ciphertext || Tag and validated per block number.

5. Message Formats

All multi-octet fields are network byte order (big-endian). HMTFTP reuses base TFTP wire formats and appends TLVs to RRQ/WRQ/OACK only.

5.1. RRQ and WRQ

RRQ/WRQ = OpCode (2) || Filename (N) || 0 || Mode (M) || 0 || optional TLV sequence

The optional TLV sequence starts immediately after the Mode terminating NUL octet and continues to datagram end.

5.2. OACK

OACK = OpCode (2) || TLVs

If RRQ/WRQ contains one or more TLVs and the request is accepted, the server MUST send OACK (possibly with selected/adjusted values).

If RRQ/WRQ contains no TLV, the server SHOULD NOT send OACK and SHOULD proceed with baseline TFTP exchange.

If RRQ/WRQ carries TLVs and the request is accepted, OACK defines the negotiation boundary. The OACK TLV sequence MAY be empty when no TLV is selected or echoed. An empty OACK indicates that the request was accepted but no offered TLV resulted in an echoed or adjusted negotiation item.

5.3. DATA and ACK

DATA = OpCode (2) || Block (2) || Payload (0..n)

ACK = OpCode (2) || Block (2)

In AEAD mode, Payload is Ciphertext || Tag, where Tag is 16 octets for AES-GCM.

5.4. ERROR

ERROR = OpCode (2) || ErrorCode (2) || ErrMsg (string) || 0

For extension processing failures, endpoints SHOULD use ErrorCode 0 ("Not defined") with an informative text message.

6. TLV Encoding, Definitions, and Negotiation

6.1. TLV Format and Processing Rules

Field	Size	Description
Type	16 bits	MSB is Critical bit; lower 15 bits are TLV code
Length	16 bits	Length of Value in octets
Value	variable	Type-specific data

Table 1: TLV Format

TLVs MAY appear only in RRQ, WRQ, and OACK.

Processing rules:

* Unknown TLVs with Critical=0 MUST be ignored.

* Unknown TLVs with Critical=1 MUST cause rejection with ERROR.

* A TLV MUST NOT be duplicated unless its definition explicitly permits repetition.

* Malformed TLVs (length mismatch, truncation) MUST be rejected.

6.2. Defined TLVs

Code	Name	Length	Description
0x0001	BLKSIZE	2	Maximum DATA plaintext payload size (uint16)
0x0002	TIMEOUT	2	Requested retransmission timeout in seconds (uint16)
0x0003	TSIZE	8	Transfer size in octets (uint64)
0x0010	ENC_REQ	0	Request AEAD protection for DATA payloads
0x0011	CIPHER	2	Selected ciphersuite

			(uint16)
0x0012	CNONCE	16	Client nonce (16 octets from CSPRNG)
0x0013	SNONCE	16	Server nonce (16 octets from CSPRNG)

Table 2: Defined TLVs

BLKSIZE in OACK MUST be less than or equal to the client offered BLKSIZE. TIMEOUT in OACK MUST be an acceptable value to both peers.

For TSIZE, RRQ MAY send TSIZE=0 to request file size; WRQ SHOULD send known size when available.

6.3. Security Negotiation

Security mode is negotiated by TLVs and is explicitly bound to ENC_REQ, CNONCE, SNONCE, and CIPHER:

* Client requests security by sending ENC_REQ in RRQ/WRQ.
* When ENC_REQ is present, client MUST include CNONCE and CIPHER.
* If server accepts, OACK MUST include ENC_REQ, SNONCE, and CIPHER.
* When echoing ENC_REQ in OACK, server MUST preserve the request Critical bit in the TLV Type field (0x8010 -> 0x8010, 0x0010 -> 0x0010).
* CIPHER in OACK MUST equal client CIPHER; mismatch MUST be rejected.
* If client marks ENC_REQ as Critical and server does not echo ENC_REQ with matching Type (including Critical bit), client MUST abort.

This specification defines ciphersuite 0x0001 as AEAD AES-256-GCM.

7. Transfer Procedure

7.1. Client and Server Behavior

1. Client sends RRQ or WRQ with desired TLVs.

2. Server validates request, TLVs, and policy; with TLVs present it returns OACK with selected values, otherwise it follows RFC1350 immediate transfer behavior.

3. RRQ after OACK: client sends ACK(0), then server starts DATA(1).

4. WRQ after OACK: client starts DATA(1), then server ACKs each block.

5. Transfer completes when sender transmits a DATA block with payload length smaller than negotiated BLKSIZE (plaintext size in AEAD mode).

When TLVs were present in RRQ/WRQ and accepted, OACK is mandatory. When no TLV is present, OACK is normally omitted.

7.2. Fallback and Abort Conditions

If no TLVs are present and no OACK is sent, peers follow direct TFTP behavior per [RFC1350].

If TLVs are present but server cannot or will not negotiate, server MUST reject the request with ERROR; silent fallback to non-negotiated transfer in that case is not permitted.

Unknown critical TLVs, malformed TLVs, missing required security TLVs, and unsupported requested ciphersuite MUST cause request rejection.

8. Key Derivation

HMTFTP assumes an externally provisioned PSK. Key derivation uses HKDF-SHA-256 as specified in [RFC5869]. In security mode:

* IKM = PSK

* salt = CNONCE || SNONCE (32 octets)

* info = "hmtftp keys v1"

* OKM = HKDF-SHA-256(IKM, salt, info, 44)

* key = OKM[0..31] (32 octets)

* iv_base = OKM[32..43] (12 octets)

Implementations MUST use the exact info string above for interoperability.

9. Nonce Construction

For DATA block number n, nonce (12 octets) is:

nonce = iv_base[0..7] || uint32(n)

where uint32(n) is big-endian and n is the 16-bit DATA block number widened to 32 bits.

Block-number wrap MUST NOT occur within one security context. Endpoints MUST terminate transfer before 65536 DATA blocks.

10. AEAD Processing Rules

This section applies only when security mode is successfully negotiated.

* Algorithm: AEAD AES-256-GCM ([RFC5116]).
* AAD: 4-octet DATA header (OpCode || Block).
* Ciphertext length equals plaintext length.
* Tag length is 16 octets and is appended to ciphertext.
* Retransmission of a DATA block MUST retransmit identical ciphertext and tag for that block number.
* A valid duplicate DATA(n) (same ciphertext and tag as previously accepted for n) MUST be acknowledged with ACK(n) and MUST NOT be delivered twice to the application.
* If a DATA(n) authenticates but differs from the stored accepted ciphertext/tag for the same n, the endpoint MUST abort the transfer.
* Duplicate ACK(n) packets MAY occur and MUST be ignored if they do not advance sender state.

On authentication failure, receiver MUST discard the DATA block and MUST NOT acknowledge it as valid. The receiver MUST NOT emit per-packet authentication error signals.

Duplicate packets within the same transfer context are treated as retransmissions, not as replays. Packets from other transfer contexts (different TID and/or different negotiated key material) MUST NOT be accepted as valid for the active transfer.

After consecutive authentication failures or retransmission timeout exhaustion for a block, endpoint MUST terminate the transfer context. It MAY send at most one generic ERROR (ErrorCode 0) before closing.

BLKSIZE in security mode refers to plaintext bytes only.

11. Error Handling

ERROR packet format and baseline error semantics follow [RFC1350].

HMTFTP-specific requirements:

* Unsupported or unknown critical TLV: reject with ERROR and terminate request processing.

* Malformed TLV sequence: reject with ERROR.

* Missing required security TLVs under ENC_REQ: reject with ERROR.

* Unexpected TID during transfer MUST be handled according to [RFC1350] (Unknown transfer ID).

* Authentication failures on DATA in AEAD mode MUST be handled as specified in Section 10.

12. Timers and Retransmissions

Endpoints SHOULD support configurable retransmission timeout (RTO), exponential backoff, and a maximum retransmission count.

TIMEOUT TLV (if offered and accepted) sets the initial RTO value for the transfer.

In AEAD mode, retransmissions MUST reuse exactly the same ciphertext and tag for a given DATA block.

13. Datagram Sizing and PMTU

Implementations SHOULD avoid IP fragmentation and follow UDP usage guidance from [RFC8085].

In AEAD mode, effective DATA datagram payload is: 4 (DATA header) + BLKSIZE + 16 (GCM tag).

Senders SHOULD choose BLKSIZE so resulting datagrams fit path MTU and SHOULD reduce BLKSIZE when fragmentation or Packet Too Big indications are observed.

14. Resource Limits and DoS Mitigations

Servers SHOULD bound pre-authentication work, rate-limit ERROR responses, and cap concurrent transfers.

Implementations SHOULD enforce limits for:

* maximum negotiated BLKSIZE;
* maximum transfer size and duration;
* maximum outstanding transfer contexts.

In untrusted environments, additional address-validation or cookie mechanisms may be required by deployment policy.

15. Use Cases and Applicability

HMTFTP is intended for environments where TFTP simplicity is desired but optional payload protection is needed without deploying heavier secure transports.

* secure PXE or boot workflows in controlled infrastructure;
* firmware distribution for managed device fleets;
* embedded systems with constrained software footprint;
* controlled operational networks where full TLS-based file transfer is impractical.

HMTFTP does not protect RRQ/WRQ/OACK metadata and is not a replacement for general-purpose secure file transfer protocols on untrusted networks.

16. Implementation Status

This section is provided in accordance with [RFC7942]. RFC Editor: please remove this section before publication as an RFC.

As of March 29, 2026, one public prototype implementation of HMTFTP is available at: https://github.com/c4tzzz/hmtftp (https://github.com/c4tzzz/hmtftp).

The implementation is written in Python and includes client and server components. The README states Python >= 3.12. In this environment, the executed interpreter version was Python 3.13.5.

Based on the provided source tree (README and code under hmtftp/protocol), the implementation covers RRQ/WRQ/OACK/DATA/ACK/ERROR processing, TLV negotiation, optional AEAD DATA protection (AES-256-GCM), HKDF-SHA-256 key derivation, and automated tests.

Repository commit identification:

<CODE BEGINS>
```
git clone https://github.com/c4tzzz/hmtftp /tmp/hmtftp-upstream
git -C /tmp/hmtftp-upstream rev-parse HEAD
# output:
# 5f9a894f6ae9f23ab39b3dd0a6db0d044ee65eb8
```
<CODE ENDS>

The local source snapshot under experimental/hmtftp-main does not include .git metadata; therefore, the revision above was collected from a fresh clone of the same repository URL.

Public revision permalink: https://github.com/c4tzzz/hmtftp/tree/5f9a894f6ae9f23ab39b3dd0a6db0d044ee65eb8 (https://github.com/c4tzzz/hmtftp/tree/5f9a894f6ae9f23ab39b3dd0a6db0d044ee65eb8).

License metadata:

<CODE BEGINS>
```
find /tmp/hmtftp-upstream -maxdepth 2 -type f \
  \( -iname 'LICENSE*' -o -iname 'COPYING*' -o -iname 'NOTICE*' \)
# output:
# (no matching files)
```
<CODE ENDS>

No explicit license file was found in the upstream repository top-level tree at the referenced revision. This remains an implementation packaging gap, not a wire-protocol issue.

Reproducibility commands executed in this environment:

```
<CODE BEGINS>
python3 -m venv .venv
.venv/bin/pip install -r requirements.txt
.venv/bin/pytest -q
# output:
# 25 passed in 1.70s

./scripts/smoke_local.sh
# output:
# smoke test OK
# artifacts in: /tmp/hmtftp-smoke.O8Njj3
# downloaded: /tmp/hmtftp-smoke.O8Njj3/out/hello.txt
#             /tmp/hmtftp-smoke.O8Njj3/out/secure.bin
# uploaded: /tmp/hmtftp-smoke.O8Njj3/server/inbox/local.bin
<CODE ENDS>
```

AEAD capture command from the README was attempted with non-interactive sudo and did not execute in this environment due sudo restrictions:

```
<CODE BEGINS>
sudo -n WAIT_SECS=10 ./scripts/capture_aead.sh \
  /tmp/hmtftp-aead-proof.pcap 192.0.2.1 6369 \
  demo-psk-32-bytes-minimum
# or with an IPv6 documentation address:
sudo -n WAIT_SECS=10 ./scripts/capture_aead.sh \
  /tmp/hmtftp-aead-proof.pcap 2001:db8::1 6369 \
  demo-psk-32-bytes-minimum
# output:
# sudo: a password is required
<CODE ENDS>
```

Packet-capture evidence is therefore not included in this revision.

At the time of writing, this document does not claim multiple independent interoperable implementations. The available code is intended to support experimentation, protocol validation, and implementation feedback for the Experimental track.

17. Experimental Scope and Evaluation Plan (Informative)

The wire protocol defined in this document remains aligned with the stable core behavior already documented in prior drafts. The experimental focus for this version is implementation behavior, repeatability, and observability.

17.1. Scope of Experimental Evaluation

* interoperability behavior and negotiation outcomes;
* loss handling and retransmission behavior with stop-and-wait;
* datagram sizing behavior (BLKSIZE bounds and operational MTU assumptions);
* security downgrade handling when ENC_REQ is required by policy;
* AEAD retransmission and duplicate DATA consistency checks.

17.2. Available Evidence in Provided Sources

Topic	Evidence in README/Code/ Tests	Status
Interop diversity	Single public implementation only (Implementation Status section)	Not yet demonstrated beyond one implementation
Loss/ retransmission logic	Exponential RTO and retransmit loops in client.py/server.py; local smoke and integration tests exercise baseline secure and non-secure transfer paths	Baseline behavior covered; no controlled impairment traces in this revision
MTU/sizing behavior	BLKSIZE selection, bounds, and PMTU guidance in config.py, server negotiation, and Section 13	Negotiation bounds covered; no path-MTU fault traces in this revision
Downgrade controls	OACK validation and ENC_REQ handling in tlv.py/client.py; dedicated regression test in tests/ test_negotiation.py	Covered by dedicated unit test
AEAD retransmissions	Session payload caches and duplicate-wire consistency checks in session.py,	Covered for duplicate-payload handling; no

		client.py, and server.py; dedicated regression tests in tests/test_session.py	packet-loss trace archive in this revision

Table 3: Experimental Evidence and Current Status

17.3. Measurement Plan

* Use the reproducible procedure in Appendix B as baseline run criteria.

* Use pytest and smoke scripts for baseline correctness checks.

* Use packet capture workflow from README to inspect secure DATA exchanges when privileges are available.

* Future evaluation should add repeatable network impairment scenarios (loss, reordering, duplication) with archived command lines and logs.

17.4. Out of Scope for This Draft Revision

Native post-quantum key exchange in the HMTFTP wire protocol is out of scope for this draft revision. Appendix E discusses only out-of-band PSK provisioning considerations.

18. Security Considerations

Without security mode, HMTFTP provides no confidentiality or integrity beyond UDP checksums, consistent with baseline TFTP.

With security mode enabled, only DATA payloads are encrypted and authenticated. RRQ/WRQ/OACK metadata (filename, mode, TLVs) remains cleartext and observable on path.

Nonce reuse with AES-GCM is catastrophic. Implementations MUST enforce nonce uniqueness and MUST prevent block-number wrap in one security context.

Endpoints MUST treat authentication failures as potential active attacks or corruption events and MUST terminate transfer state after bounded retry limits, without exposing per-packet validity side channels.

Clients requiring AEAD protection MUST mark ENC_REQ as Critical and MUST abort if negotiation is downgraded or not acknowledged.

PSK provisioning, storage, rotation, and access control are out of scope for wire protocol definition but are critical to real security.

19. IANA Considerations

19.1. Service Name and UDP Port

IANA is requested to assign a service name and UDP port in the "Service Name and Transport Protocol Port Number Registry" ([RFC6335]). Until assignment, this document uses TBD1.

Service Name	Transport Protocol	Port Number	Reference
hmtftp	udp	TBD1	This document

Table 4: HMTFTP Service Name and Port

19.2. HMTFTP TLV Types Registry

IANA is requested to create "HMTFTP TLV Types" for 15-bit TLV codes (0x0000-0x7FFF). The Critical bit (MSB on wire) is not part of the registry.

Registration policy: Expert Review ([RFC8126]).

The range 0x7F00-0x7FFF is reserved for Private Use.

Code	Name	Description	Reference
0x0000	Reserved	Reserved for future use	This document
0x0001	BLKSIZE	Block size in octets (uint16)	This document
0x0002	TIMEOUT	Retransmission timeout in seconds (uint16)	This document
0x0003	TSIZE	Transfer size in octets (uint64)	This document
0x0010	ENC_REQ	Request AEAD protection (empty value)	This document
0x0011	CIPHER	Select ciphersuite (uint16)	This document
0x0012	CNONCE	Client nonce (16 octets)	This document
0x0013	SNONCE	Server nonce (16 octets)	This document

Table 5: Initial TLV Registry Entries

19.3. Opcode Values

This document does not define new opcode values and therefore does not request creation of an HMTFTP OpCodes registry. Opcode values remain as defined by [RFC1350] and [RFC2347].

19.4. HMTFTP Ciphersuites Registry

IANA is requested to create "HMTFTP Ciphersuites" with Expert Review policy ([RFC8126]).

Value	Name	Description	Reference
0x0000	Reserved	Reserved for future use	This document
0x0001	AES-256-GCM	AEAD AES-256-GCM	This document

Table 6: Initial Ciphersuite Registry Entries

20. References

20.1. Normative References

[RFC1350] Sollins, K., "The TFTP Protocol (Revision 2)", RFC 1350, July 1992, <https://www.rfc-editor.org/rfc/rfc1350>.

[RFC2119] Bradner, S., "Key words for use in RFCs to Indicate Requirement Levels", BCP 14, RFC 2119, March 1997, <https://www.rfc-editor.org/rfc/rfc2119>.

[RFC8174] Leiba, B., "Ambiguity of Uppercase vs Lowercase in RFC 2119 Key Words", BCP 14, RFC 8174, May 2017, <https://www.rfc-editor.org/rfc/rfc8174>.

[RFC2347] Malkin, G., "TFTP Option Extension", RFC 2347, May 1998, <https://www.rfc-editor.org/rfc/rfc2347>.

[RFC5116] McGrew, D., "An Interface and Algorithms for Authenticated Encryption", RFC 5116, January 2008, <https://www.rfc-editor.org/rfc/rfc5116>.

[RFC5869] Krawczyk, H. and P. Eronen, "HMAC-based Extract-and-Expand Key Derivation Function (HKDF)", RFC 5869, May 2010, <https://www.rfc-editor.org/rfc/rfc5869>.

[RFC6335] Cotton, M., Leiba, B., and T. Narten, "Internet Assigned Numbers Authority (IANA) Procedures for the Management of the Service Name and Transport Protocol Port Number Registry", RFC 6335, August 2011, <https://www.rfc-editor.org/rfc/rfc6335>.

[RFC8085] Eggert, L., Fairhurst, G., and G. Shepherd, "UDP Usage Guidelines", RFC 8085, March 2017, <https://www.rfc-editor.org/rfc/rfc8085>.

[RFC8126] Cotton, M., Leiba, B., and T. Narten, "Guidelines for Writing an IANA Considerations Section in RFCs", BCP 26, RFC 8126, June 2017, <https://www.rfc-editor.org/rfc/rfc8126>.

20.2. Informative References

[FIPS203] Technology, N. I. O. S. A., "Module-Lattice-Based Key-Encapsulation Mechanism Standard", FIPS 203, DOI 10.6028/NIST.FIPS.203, 13 August 2024, <https://doi.org/10.6028/NIST.FIPS.203>.

[RFC7942] Bormann, C., "Improving Awareness of Running Code: The Implementation Status Section", RFC 7942, July 2016, <https://www.rfc-editor.org/rfc/rfc7942>.

Appendix A. Example Negotiated RRQ Exchange (Informative)

This appendix illustrates a complete RRQ flow with TLV negotiation and AEAD mode enabled. Values are examples for implementer guidance.

Request context:

* Filename: "firmware.bin"

* Mode: "octet"

* Requested BLKSIZE: 1024

* Requested TIMEOUT: 3 seconds

* Security requested: ENC_REQ (Critical), CIPHER=0x0001

* CNONCE: 16 octets from CSPRNG

Example TLVs in RRQ (Type, Length, Value):

```
BLKSIZE: 0x0001 0x0002 0x0400
TIMEOUT: 0x0002 0x0002 0x0003
ENC_REQ: 0x8010 0x0000
CIPHER : 0x0011 0x0002 0x0001
CNONCE : 0x0012 0x0010 00112233445566778899aabbccddeeff
```

Server accepts request and returns OACK with negotiated values:

```
   BLKSIZE: 0x0001 0x0002 0x0400
   TIMEOUT: 0x0002 0x0002 0x0003
   ENC_REQ: 0x8010 0x0000
   CIPHER : 0x0011 0x0002 0x0001
   SNONCE : 0x0013 0x0010 a0a1a2a3a4a5a6a7a8a9aaabacadaeaf
```

RRQ transfer sequence then proceeds:

1. Client -> Server: RRQ + TLVs

2. Server -> Client: OACK + TLVs

3. Client -> Server: ACK(0)

4. Server -> Client: DATA(1) with ciphertext length 1024 and tag length 16

5. Client -> Server: ACK(1)

6. Server -> Client: DATA(2) ...

7. ...

8. Server -> Client: final DATA(n) with plaintext length < 1024 (EOF)

9. Client -> Server: ACK(n)

For DATA block 1, nonce is computed as:

```
nonce(1) = iv_base[0..7] || 0x00000001
AAD      = DATA header (OpCode=3, Block=1)
```

If DATA(1) is retransmitted due to loss, ciphertext and tag are retransmitted unchanged and the receiver re-ACKs block 1 without delivering duplicate plaintext.

Appendix B. Reproducible Test Procedure (Informative)

B.1. Prerequisites from Provided Sources

* Python >= 3.12 (README prerequisite).

* pip and requirements.txt dependencies: cryptography>=42.0.0 and pytest>=8.2.0.

* Optional for capture workflow: tcpdump and tshark (README).

B.2. Procedure Commands

```
<CODE BEGINS>
python3 -m venv .venv
.venv/bin/pip install -r requirements.txt
.venv/bin/pytest -q
./scripts/smoke_local.sh
# optional capture workflow (requires sudo):
sudo WAIT_SECS=10 ./scripts/capture_aead.sh \
  /tmp/hmtftp-aead-proof.pcap 192.0.2.1 6369 \
  demo-psk-32-bytes-minimum
# or:
sudo WAIT_SECS=10 ./scripts/capture_aead.sh \
  /tmp/hmtftp-aead-proof.pcap 2001:db8::1 6369 \
  demo-psk-32-bytes-minimum
<CODE ENDS>
```

B.3. Execution Status in This Environment

Command	Result	Observed Output
python3 -m venv .venv	PASS	Virtual environment created
.venv/bin/pip install -r requirements.txt	PASS	Dependencies installed
.venv/bin/pytest -q	PASS	25 passed in 1.70s
./scripts/smoke_local.sh	PASS	smoke test OK
sudo -n WAIT_SECS=10 ./scripts/capture_aead.sh ...	TODO (not executed)	sudo: a password is required

Table 7: Execution Status (March 29, 2026)

Appendix C. Crypto / TLV Consistency Notes (Informative)

C.1. HKDF and Key Material Split

* IKM = PSK (crypto.py: derive_key_material).
* salt = CNONCE || SNONCE (crypto.py; constants CNONCE_LEN=16 and SNONCE_LEN=16).
* info string = hmtftp keys v1 (constants HKDF_INFO).

* OKM length = 44 octets (constants HKDF_OKM_LEN).

* key = OKM[0..31] and iv_base = OKM[32..43] (crypto.py).

C.2. Nonce and AAD Construction

* Nonce format: iv_base[:8] || uint32(block) in network byte order (crypto.py: build_nonce).

* AAD format: struct.pack('!HH', OP_DATA, block), i.e., DATA opcode + block number (crypto.py: aad_for_block).

C.3. TLV IDs and Lengths from Code

TLV	Code	Length (octets)	Source
BLKSIZE	0x0001	2	constants.py, tlv.py KNOWN_TLV_LENGTHS
TIMEOUT	0x0002	2	constants.py, tlv.py KNOWN_TLV_LENGTHS
TSIZE	0x0003	8	constants.py, tlv.py KNOWN_TLV_LENGTHS
ENC_REQ	0x0010	0	constants.py, tlv.py KNOWN_TLV_LENGTHS
CIPHER	0x0011	2	constants.py, tlv.py KNOWN_TLV_LENGTHS
CNONCE	0x0012	16	constants.py, tlv.py KNOWN_TLV_LENGTHS
SNONCE	0x0013	16	constants.py, tlv.py KNOWN_TLV_LENGTHS

Table 8: TLV Code and Length Consistency

C.4. Deterministic Vector Executed from Implementation

The following values were executed from the local implementation and match tests/test_crypto.py:

```
   PSK    = 303132333435363738396162636465 66
            303132333435363738396162636465 66
   CNONCE = 00112233445566778899aabbccddeeff
   SNONCE = a0a1a2a3a4a5a6a7a8a9aaabacadaeaf
   key    = e14c36452ca1954c3929b824ececc63d
            fa5c7e4203c75b98ee16f46a0a852cc6
   iv_base= 8a5669a255600f2c7c6ae475
   nonce1 = 8a5669a255600f2c00000001
   aad1   = 00030001
```

Appendix D. Conformance Matrix (Informative)

Spec Requirement	Code Location	Test ID
Unknown critical TLV MUST be rejected	tlv.py: parse_tlvs	T1
Duplicate TLV in one message MUST be rejected	tlv.py: parse_tlvs	T2
Malformed or truncated TLV sequence MUST be rejected	tlv.py: parse_tlvs	T3
SNONCE MUST NOT appear in RRQ or WRQ	tlv.py: validate_request_tlvs	T4
Secure OACK MUST include required security TLVs	tlv.py: validate_oack_tlvs; server.py: _negotiate	T5, T6
HKDF profile MUST derive key and iv_base from PSK and nonces	crypto.py: derive_key_material	T7
Nonce and AAD format MUST follow DATA block binding	crypto.py: build_nonce; aad_for_block	T8
AEAD authentication failures MUST be detected	crypto.py: AeadCipher.decrypt_block; client.py/server.py bounded retry handling	T9
Secure GET and PUT	client.py: download and upload;	T10,

transfer path with OACK negotiation	server.py: _serve_rrq and _serve_wrq	T11
Secure downgrade by missing ENC_REQ in OACK MUST be rejected	client.py: _apply_oack; tlv.py: validate_oack_tlvs	T12
Wire duplicate DATA retransmission consistency checks	session.py: validate_secure_duplicate_data; client.py/server.py duplicate handling	T13, T14, T15

Table 9: Requirement to Code and Test Mapping

Test ID mapping:

* T1 = tests/test_tlv.py function test_parse_unknown_critical_tlv_rejected

* T2 = tests/test_tlv.py function test_duplicate_tlv_rejected

* T3 = tests/test_tlv.py function test_malformed_tlv_rejected

* T4 = tests/test_tlv.py function test_request_forbidden_snonce_rejected

* T5 = tests/test_negotiation.py function test_rrq_secure_negotiation_produces_required_oack_tlvs

* T6 = tests/test_negotiation.py function test_cipher_mismatch_in_oack_rejected

* T7 = tests/test_crypto.py function test_hkdf_derivation_vector

* T8 = tests/test_crypto.py function test_nonce_and_aad_construction

* T9 = tests/test_crypto.py function test_aead_authentication_failure

* T10 = tests/test_integration_transfer.py function test_secure_download_with_oack

* T11 = tests/test_integration_transfer.py function test_secure_upload

* T12 = tests/test_negotiation.py function test_client_rejects_secure_downgrade_when_oack_omits_security_tlvs

* T13 = tests/test_session.py function test_secure_duplicate_identical_ok

* T14 = tests/test_session.py function test_secure_duplicate_invalid_dropped

* T15 = tests/test_session.py function test_secure_duplicate_valid_mismatch_aborts

Appendix E. Post-Quantum PSK Provisioning (Informative)

This appendix is informative and does not modify the HMTFTP wire protocol. The protocol continues to assume an externally provisioned PSK.

One experimental approach is to establish shared key material out of band using ML-KEM from [FIPS203], then derive and provision a PSK for HMTFTP endpoints.

1. Perform ML-KEM key establishment out of band.

2. Derive a deployment PSK from the established shared secret (policy-specific derivation outside this wire specification).

3. Install the resulting PSK on both peers before HMTFTP transfer.

This draft does not specify a tested ML-KEM tooling workflow. The appendix is limited to architectural positioning of out-of-band PSK provisioning and does not define a deployment procedure.

Native post-quantum handshake negotiation inside HMTFTP messages is out of scope for this document revision.

Acknowledgements

The author thanks David Mercier for helpful discussions and feedback.

Author’s Address

A. Maurette
IUT R&T Bethune
France
Email: contact@c4tz.fr