Network Working Group
Internet-Draft
Intended status: Experimental
Expires: 22 July 2026

A. Maurette
IUT R&T B辿thune

18 January 2026

HMTFTP: HMAC-Derived TFTP with Optional AEAD Protection (v0.2)
draft-maurette-hmtftp-02

Abstract

HMTFTP is a lightweight UDP file transfer protocol that preserves the simplicity of TFTP (block-and-ACK) while adding a structured TLV extension mechanism and an optional authenticated-encryption mode. When negotiated, DATA payloads are protected with AEAD AES-256-GCM and keys are derived with HKDF-SHA-256 from a pre-shared key (PSK). The default UDP port is TBD (requested: 6369) and implementations MUST allow it to be configured.

Additional Status Note

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."

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 22 July 2026.

Table of Contents

1. Introduction

The Trivial File Transfer Protocol (TFTP) [RFC1350] is extremely simple but provides no built-in security properties. HMTFTP retains the TFTP operational model (UDP, numbered blocks, ACKs) while introducing (1) a compact TLV extension mechanism and (2) an optional AEAD protection mode for DATA payloads.

The name "HMTFTP" reflects that cryptographic keys are derived using HKDF, a HMAC-based key derivation function [RFC5869] . This version (v0.2) focuses on removing specification blockers: well-defined extension processing rules and explicit security boundaries (what is encrypted vs. authenticated).

2. Conventions and Terminology

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

This document uses the following terms:

* *PSK* : pre-shared key

* *AEAD* : authenticated encryption with associated data

* *AAD* : additional authenticated data

3. Relationship to TFTP

HMTFTP is derived from TFTP [RFC1350] and reuses the core message types and semantics (RRQ, WRQ, DATA, ACK, ERROR). It also reuses the concept of an explicit option acknowledgment, OACK, as introduced by TFTP option extension [RFC2347] . HMTFTP replaces the key/value option encoding of RFC 2347 with a TLV encoding defined in this document.

HMTFTP differs from baseline TFTP primarily by:

* using UDP port TBD by default (requested: 6369; configurable), rather than 69;

* allowing TLV extensions in RRQ, WRQ, and OACK;

* supporting an optional AEAD security mode for DATA payloads.

4. Transport

HMTFTP runs over UDP. The default server port is TBD (requested: 6369), but implementations MUST allow the port to be configured.

As in TFTP, a transfer is conducted between a client and a server transfer address (IP, UDP port). The server MAY respond from a different UDP port than the configured well-known port for the remainder of the transfer, as described in [RFC1350] .

5. Message Formats

All multi-octet fields are encoded in network byte order (big-endian). HMTFTP reuses the TFTP base message formats, with TLVs appended to RRQ, WRQ, and OACK. TLVs are not used in DATA, ACK, or ERROR in v0.2.

5.1. RRQ and WRQ

RRQ and WRQ are defined as in [RFC1350] :

RRQ/WRQ = OpCode (2) || Filename (N) || 0 || Mode (M) || 0 || [TLVs]

The optional TLV sequence, when present, begins immediately after the terminating zero octet of the Mode field and continues to the end of the UDP datagram. The Mode is a NUL-terminated ASCII string (e.g., "octet").

5.2. OACK

OACK is used by the server to acknowledge and/or modify the TLVs offered in RRQ/WRQ. OACK is defined by [RFC2347] as OpCode value 6. In HMTFTP, OACK contains only a TLV sequence:

OACK = OpCode (2) || TLVs

An OACK with an empty TLV sequence indicates acceptance with no negotiated parameters.

5.3. DATA and ACK

DATA and ACK are as defined in [RFC1350] :

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

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

When AEAD protection is negotiated (Section 8), the DATA Payload is structured as: Ciphertext || Tag, where Tag is a 16-octet AES-GCM authentication tag. The ciphertext length is the datagram length minus 4 octets of header and minus 16 octets of tag.

5.4. ERROR

ERROR is as defined in [RFC1350] :

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

HMTFTP endpoints SHOULD use an ERROR with ErrorCode 0 ("Not defined") for extension processing failures (e.g., unsupported critical TLV).

6. TLV Encoding and Processing

HMTFTP TLVs extend RRQ, WRQ, and OACK. TLVs use a compact binary encoding:

Field	Size	Description
Type	16 bits	Type code with Critical bit in MSB
Length	16 bits	Length of Value in octets
Value	variable	Type-specific data

Table 1: TLV Format

The most significant bit (MSB) of the Type field is the _Critical_ bit. Bits 0-14 form the 15-bit TLV code. The Critical bit is not part of any future registry code space that may be defined for TLV codes.

Processing rules:

* A receiver MUST ignore unknown TLVs with Critical=0.
* A receiver that encounters an unknown TLV with Critical=1 MUST reject the message by sending an ERROR (and MUST NOT proceed with the transfer).
* A receiver MAY accept known TLVs in any order. If a TLV appears multiple times, a receiver SHOULD treat this as an error unless the TLV definition explicitly allows repetition.

6.1. Defined TLVs

This specification defines the following TLVs. All multi-octet values are encoded in network byte order (big-endian).

Code	Name	Length	Description
0x0001	BLKSIZE	2	Requested maximum DATA payload size in octets (uint16). If offered by a client, the server MUST respond with BLKSIZE in OACK with the selected value, which MUST be less than or equal to the requested value.
0x0002	TIMEOUT	2	Requested retransmission timeout in seconds (uint16). If offered by a client, the server MUST respond with TIMEOUT in OACK with the selected value (which MAY be less than the requested value) or reject the request.
0x0003	TSIZE	8	Transfer size in octets (uint64). In RRQ, a client MAY send TSIZE=0 to request that the server return the size. In WRQ, a client SHOULD send TSIZE with the size if known.
0x0010	ENC_REQ	0	Request to enable AEAD protection for DATA payloads. Clients that require security mode MUST set the Critical bit on ENC_REQ. Servers that accept security mode MUST echo ENC_REQ in OACK.
0x0011	CIPHER	2	Select ciphersuite (uint16). If omitted, the default ciphersuite is 0x0001 (AES-256-GCM).
0x0012	CNONCE	16	Client nonce (16 octets) generated by a CSPRNG. CNONCE MUST be present in RRQ/WRQ when ENC_REQ is present.
0x0013	SNONCE	16	Server nonce (16 octets) generated by a CSPRNG. SNONCE MUST be present in OACK when ENC_REQ is accepted.

Table 2: Defined TLVs (v0.2)

The ciphersuite value 0x0001 corresponds to AEAD AES-256-GCM.

7. Transfer Procedure

HMTFTP uses the following procedure, aligned with TFTP option negotiation [RFC2347] :

1. The client sends RRQ or WRQ, optionally with TLVs.

2. If the server accepts the request and any offered parameters, it replies with OACK containing the negotiated TLVs (which MAY be empty). If the server does not support a critical TLV or rejects parameters, it replies with ERROR.

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

4. For WRQ: the client starts with DATA(1) after receiving OACK, and the server acknowledges each block with ACK(n).

Apart from the OACK exchange, block numbering, retransmissions, and EOF signaling follow [RFC1350] .

8. Optional AEAD Security Mode

Security mode is negotiated using TLVs in RRQ/WRQ and OACK. When enabled, each DATA payload is protected with AEAD AES-256-GCM [RFC5116] . The AEAD key and IV base are derived using HKDF-SHA-256 [RFC5869] .

8.1. Negotiation TLVs

The client requests security mode by including TLV ENC_REQ in RRQ/ WRQ. When ENC_REQ is present, the client MUST include CNONCE and MAY include CIPHER. If the server accepts, it includes ENC_REQ and SNONCE in OACK and MAY include (or echo) CIPHER. If the server does not support security mode, it MUST reject a Critical ENC_REQ with ERROR.

8.2. Key Derivation

This document assumes an externally provisioned PSK (32 octets RECOMMENDED). During negotiation, the client and server exchange nonces: CNONCE and SNONCE, each 16 octets from a CSPRNG.

The AEAD key material is derived as follows:

* IKM = PSK

* salt = CNONCE || SNONCE (32 octets)

* info = "hmtftp v0.2 keys"

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

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

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

8.3. Nonce Construction and AAD

The AES-GCM nonce (12 octets) for DATA block number _n_ is:

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

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

The AEAD AAD is the 4-octet DATA header (OpCode || Block). RRQ/WRQ/OACK metadata and TLVs are not encrypted and are not included in the DATA AAD in v0.2.

Retransmissions MUST retransmit the exact same ciphertext and tag for a given block number (key, nonce).

8.4. Limits

To avoid nonce reuse, endpoints MUST NOT allow the 16-bit block number to wrap within a security context. Implementations SHOULD terminate a transfer with ERROR well before wrap if it would be reached.

9. Security Considerations

Without security mode, HMTFTP provides no confidentiality or integrity beyond UDP checksums and is vulnerable to on-path modification and spoofing, as with TFTP [RFC1350] .

With security mode enabled, only DATA payloads are encrypted and authenticated. RRQ/WRQ/OACK metadata and TLVs remain in cleartext. This means filenames, modes, and negotiated parameters are observable on the wire. Deployments that require metadata confidentiality MUST avoid placing sensitive data in RRQ/WRQ/OACK and SHOULD use an external secure channel or a future extension that encrypts metadata.

Nonce reuse with AES-GCM is catastrophic. Implementations MUST enforce nonce uniqueness and MUST follow the nonce construction and wrap limits described in Section 8 .

Implementations should also consider UDP robustness guidelines ([RFC8085]) and rate-limiting to mitigate amplification and resource-exhaustion attacks.

10. Implementation Status

This section is provided for RFC 7942 compliance ([RFC7942]). Implementations, interop notes, and known limitations will be added in subsequent versions.

11. References

11.1. 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>.

[RFC2348] Malkin, G., "TFTP Blocksize Option", RFC 2348, May 1998, <https://www.rfc-editor.org/rfc/rfc2348>.

[RFC2349] Malkin, G., "TFTP Timeout Interval and Transfer Size Options", RFC 2349, May 1998, <https://www.rfc-editor.org/rfc/rfc2349>.

[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>.

11.2. Informative References

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

Author’s Address

A. Maurette
IUT R&T B辿thune
France
Email: contact@c4tz.fr