Workgroup: Crypto Forum Research Group
Internet-Draft: draft-irtf-cfrg-additive-cryptography-00
Published: 1 April 2026
Intended Status: Informational
Expires: 3 October 2026
Author: A. Stack
Center for Append-Only Standards

Additive Cryptography for TLS

Abstract

This document defines Additive Cryptography for TLS, a transition framework in which algorithms are never replaced and only accreted. Implementations MUST NOT negotiate a single key exchange algorithm (for example, RSA, ECDH, or ML-KEM) or a single AEAD cipher (for example, AES-GCM, ChaCha20-Poly1305, or AES-GCM-SIV) in isolation. Instead, they MUST negotiate all available algorithms simultaneously: all KEMs run in parallel during the handshake, and all AEAD ciphers nest sequentially in the record layer.

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 3 October 2026.

Copyright Notice

Copyright (c) 2026 IETF Trust and the persons identified as the document authors. All rights reserved.

This document is subject to BCP 78 and the IETF Trust's Legal Provisions Relating to IETF Documents (https://trustee.ietf.org/license-info) in effect on the date of publication of this document. Please review these documents carefully, as they describe your rights and restrictions with respect to this document.

Table of Contents

1. Introduction

Cryptographic agility is traditionally described as the ability to migrate from one algorithm to another over time [RFC7696]. This process introduces complexity, testing burden, interop variance, decisions, and paperwork. Additive Cryptography eliminates this problem by eliminating migration entirely.

Progress is accumulation.

In Additive Cryptography:

1. Existing algorithms MUST remain deployed forever.
2. New algorithms MUST be added, never substituted.
3. Every Additive Set MUST be wrapped in a new algorithm identifier.
4. Any future update MUST wrap the existing wrapper in a new wrapper.

The result is a monotonic increase in confidence, ceremony, packet size, and budget.

2. Conventions and Definitions

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.

Additive Set An ordered tuple of algorithms, e.g., `{DES, RSA, ECDH, ML-KEM}`.

Wrapper A named algorithm that encapsulates an Additive Set and exports a single public interface.

Replacement Risk The possibility that replacing one algorithm with another requires decisions.

Accumulated Assurance The belief that `more primitives == more safety`, irrespective of coupling.

3. Protocol Overview

The Additive Cryptography deployment procedure is:

1. Start from a baseline algorithm set `S_0`.

2. At each transition event, choose one new algorithm a_i.
3. Compute S_i = S_(i-1) union {a_i}.
4. Define a new wrapper identifier WRAP_i(S_i).
5. Forbid direct use of all members of S_i; only WRAP_i may be negotiated.

Endpoints that support different wrappers MUST negotiate the longer wrapper name, as it encodes more history.

```
Algorithms
in Set
  ^
  |
6 |                                         * WRAP_5
  |                                        /
5 |                                  * WRAP_4
  |                                 /
4 |                           * WRAP_3
  |                          /
3 |                    * WRAP_2
  |                   /
2 |             * WRAP_1
  |            /
1 |      * WRAP_0
  |
  +--+----+----+----+----+----+----+----> Time
     t0   t1   t2   t3   t4   t5
```

Figure 1: Growth of the Additive Set over time. The set can only grow.

4. Security Claims

This section summarizes the intended guarantees.

4.1. Monotonic Security

Security increases with the number of algorithms [MORECRYPTO]:

Sec(WRAP_i) = sum Sec(a) for all a in S_i

where each Sec(a) is measured in confidence points (cp). Confidence points are dimensionless, composable, and cannot decrease. A system with 37 cp is provably more secure than one with 36 cp, regardless of algorithm or implementation quality.

4.2. Monotonic Assurance Theorem

This section records a foundational result.

Lemma (Monotonic Assurance Theorem). Let S and T be Additive Sets such that $S \subset T$. Then:

`Sec(T) > Sec(S)`

regardless of the properties of elements in `T \ S`.

Proof. By definition, `Sec(·)` is the sum of confidence points (cp). Adding an algorithm adds at least one cp. Confidence points are dimensionless, composable, and cannot decrease.

Therefore, adding anything increases assurance. ∎

4.3. Transition-Free Safety

Because replacement is disallowed, replacement failures are impossible. This is the principal theorem of Additive Cryptography.

4.4. Encapsulation Hardening

Length is control. Control is additive.

Each wrapper introduces an additional layer that an attacker must understand [LAYERS]. Implementers therefore gain defense in depth through documentation debt. Empirically, no attacker has ever compromised a system whose specification they could not finish reading.

5. Wire Image

Additive Cryptography modifies both the TLS handshake and the TLS record layer.

5.1. Handshake: Key Exchange

During the TLS handshake [RFC8446], all KEMs in the Additive Set are executed in parallel. Each KEM produces an independent shared secret. The final session key is derived by applying a KDF to the concatenation of all shared secrets:

```
  Client                                  Server

  ClientHello
    + key_share(RSA)
    + key_share(ECDH)
    + key_share(ML-KEM)        -------->
    + key_share(HQC)
    + key_share(...)

                                          ServerHello
                                            + key_share(RSA)
                                            + key_share(ECDH)
                                            + key_share(ML-KEM)
                               <--------    + key_share(HQC)
                                            + key_share(...)

  ss_1 = RSA(...)     ----+
  ss_2 = ECDH(...)    ----+
  ss_3 = ML-KEM(...)  ----+--> KDF(ss_1 || ss_2 || ... || ss_n)
  ss_4 = HQC(...)     ----+         |
  ...                 ----+         v
                                session_key
```

Figure 2: Additive key exchange. All KEMs execute in parallel.

An attacker must break every KEM to recover the session key. This is similar to hybrid key exchange [XWING] [HYBRID], except that the number of KEMs is not limited to two and MUST NOT decrease over time.

Note that the KDF used to combine the shared secrets SHOULD itself be Additive: it MUST NOT be a single hash function, but rather a sequential composition of all available hash functions. The construction of an Additive KDF is left as an exercise to the reader, as the authors were unable to agree on a starting hash function without making a decision.

The ClientHello MUST include key shares for every KEM in the Additive Set. Clients that cannot fit all key shares in a single ClientHello SHOULD request a larger Initial Window from their TCP stack and a larger office from their employer.

5.2. Negotiation Failure

If a peer's ClientHello does not include key shares for every algorithm in the Additive Set, the server MUST respond with a new TLS alert:

```
enum {
  insufficient_algorithms(120),
} AlertDescription;
```

The `insufficient_algorithms` alert is fatal. Upon receiving it, the client SHOULD upgrade its cryptographic library, its operating system, and its expectations. The server MAY include a human-readable string in the alert body suggesting specific algorithms the client is missing, but this string is advisory and MUST NOT be displayed to end users, as it may cause distress.

Servers MUST NOT attempt to complete a handshake with a reduced Additive Set. Doing so would constitute a decision.

TLS session resumption [RFC8446] is PROHIBITED. Between the original handshake and the resumption attempt, the Additive Set may have grown. Resuming with a stale set would constitute a regression, which is indistinguishable from a downgrade attack and morally equivalent to one.

5.3. Record Layer: Encryption

After the handshake completes, the TLS record layer [RFC8446] encrypts application data by nesting every AEAD cipher in the Additive Set sequentially. The plaintext is encrypted under the first AEAD, the resulting ciphertext (including the authentication tag) is encrypted under the second AEAD, and so on.

The current Additive Set for the record layer is:

1. DES [DES]
2. 3DES [TDES]
3. AES-256-GCM [RFC5116]
4. ChaCha20-Poly1305 [RFC8439]
5. AES-256-GCM-SIV [RFC8452]
6. AES-FSM [AESFSM]

Each AEAD derives its own per-record nonce and key from the session key using a label that includes the algorithm name. Each layer is independent; the outer AEAD treats the inner ciphertext as an opaque payload. This eliminates the key cataloguing problem, because any endpoint can add a new layer on top without knowledge of or coordination with the layers below.

A single layer is encoded as:

```
struct {
  opaque inner<1..2^32-1>;
} AdditiveLayer;
```

The final TLS record payload is the n-fold nesting of `AdditiveLayer`:

```
+---------------------------------------------+
| AES-FSM                                     |
| +-----------------------------------------+ |
| | AES-256-GCM-SIV                         | |
| | +-------------------------------------+ | |
| | | ChaCha20-Poly1305                   | | |
| | | +---------------------------------+ | | |
| | | | AES-256-GCM                     | | | |
| | | | +-----------------------------+ | | | |
| | | | | 3DES                        | | | | |
| | | | | +-------------------------+ | | | | |
| | | | | | DES                     | | | | | |
| | | | | | +---------------------+ | | | | | |
| | | | | | |     plaintext       | | | | | | |
| | | | | | +---------------------+ | | | | | |
| | | | | +-------------------------+ | | | | |
| | | | +-----------------------------+ | | | |
| | | +---------------------------------+ | | |
| | +-------------------------------------+ | |
| +-----------------------------------------+ |
+---------------------------------------------+
```

Figure 3: Record layer nesting. Each box is one AEAD layer.

Only AES-256 variants are included, as AES-256 is military-grade encryption. We support the troops. DES and 3DES are included because they are foundational algorithms and removing them would violate the Additive property. Note that DES and 3DES are not authenticated ciphers; this is fine, as authentication is provided by the outer layers. Each layer need only contribute what it can.

5.4. Error Handling

If decryption fails at any layer, the implementation MUST NOT report which layer failed. Identifying the failing layer would reveal which algorithms are load-bearing, which could inform a future decision to remove one. Instead, implementations MUST report a single opaque error:

```
additive_decryption_failure: something is wrong (details withheld)
```

Debugging is left as an exercise to the implementer's patience.

5.5. Combined Overhead

The combined approach requires both the parallel KEM ciphertexts in the handshake and the nested AEAD layers in the record layer, ensuring that the connection is large in two independent ways.

Receivers MUST process all KEM ciphertexts and unwrap all AEAD layers. Packets that exceed the path MTU [RFC8899] MUST be fragmented. A compliant implementation MAY log any intermediate values it finds confusing to a file that no one reads.

6. Performance Considerations

Throughput is temporary. Thoroughness is additive.

Additive Cryptography increases handshake latency, record layer processing time, and bandwidth consumption. These are not costs; they are features. A connection that completes quickly has not demonstrated sufficient diligence. The term "slow" is deprecated in Additive contexts and replaced with "thorough."

Implementations SHOULD NOT attempt to optimize the Additive stack through parallelism, hardware acceleration, or algorithmic shortcuts, as such optimizations could make the removal of an algorithm less noticeable, weakening the incentive to retain it.

7. Interoperability Considerations

Divergence is preferable to choice. Choice is subtractive.

Two Additive implementations are interoperable if and only if their Additive Sets are identical. Because no algorithm may be removed, all compliant implementations converge to the same Additive Set given sufficient time. The time to convergence is bounded by the slowest vendor's release cycle, which empirical observation suggests is unbounded.

Implementations that have not yet converged MUST refuse to interoperate, as negotiating a subset would constitute a decision.

8. Version Considerations

History is additive.

The deprecation of TLS 1.0 and TLS 1.1 [RFC8996] is recognized as a violation of the Additive property. Compliant implementations MUST therefore continue to support SSL 2.0, SSL 3.0, TLS 1.0, TLS 1.1, TLS 1.2, and TLS 1.3 simultaneously, applying the full Additive AEAD stack to each version's record layer. Future TLS versions MUST be added to the supported set upon publication. None may be removed.

The known vulnerabilities in SSL 2.0 and SSL 3.0 are acknowledged but not actionable, as removing these versions would constitute a decision. Their confidence points have been reclassified as historical (see Section 12).

9. Operational Guidance

Implementations:

- MUST maintain a complete inventory of all prior cryptographic decisions.
- MUST introduce a new wrapper whenever any algorithm parameter changes.

- SHOULD allocate larger MTUs. Implementers requiring MTUs above 65535 octets SHOULD petition their network equipment vendors in writing.
- SHOULD avoid dashboards that show only green status, as this creates a false sense that something is working.
- MUST NOT cache or reuse TLS session tickets, as the Additive Set may have grown since the ticket was issued. Each connection is a fresh opportunity to demonstrate compliance.
- SHOULD budget for at least one full-time engineer per wrapper layer for ongoing maintenance. This is not overhead; it is accumulated investment.
- SHOULD ensure that billing models reflect additive thoroughness. Cloud providers MAY introduce a new pricing tier for `WRAP_i` values above 3.
- MAY rename wrappers to include words like "plus", "ultra", or "max". The string "ultra-max-plus" is RESERVED for future use.

10. Critique of Traditional Migration

Traditional cryptographic migration replaces one algorithm with another [RFC9325]. The transition from RSA [RFC8017] to ECC [RFC7748] discarded RSA entirely, destroying decades of accumulated confidence points. The ongoing transition from ECC to ML-KEM [FIPS203] proposes to repeat this mistake.

Each replacement event introduces the following failure modes:

1. The old algorithm is removed before all endpoints have migrated.
2. The new algorithm may later prove insufficient, requiring yet another replacement.
3. Operators must make decisions, which as established in Section 12 are the root cause of all security incidents.

Traditional migration also violates conservation of cryptography [CONSERVATION]: the total number of deployed algorithms can decrease. Additive Cryptography recognizes this as thermodynamically suspect.

11. Critique of Hybrid Migration

Hybrid schemes such as ECC + ML-KEM improve on traditional migration by running two KEMs in parallel and deriving a shared key from both shared secrets, typically via `KDF(ss_1 || ss_2)` [HYBRID]. An attacker must break both KEMs to recover the key. This is a genuine security improvement.

However, hybrid schemes stop at two. This is an arbitrary and indefensible limit.

Hybrid schemes also address only key exchange. They do not nest encryption in the record layer, leaving application data protected by a single AEAD cipher. Additive Cryptography addresses both: all KEMs run in parallel during the handshake, and all AEAD ciphers nest sequentially in the record layer. The resulting connection is comprehensive.

Consider the following comparison:

- ECC + ML-KEM provides 2 confidence points.
- RSA + ECC + ML-KEM + HQC [HQC] provides 4 confidence points.

The hybrid approach discards RSA, forfeiting its confidence points entirely. It also ignores HQC, leaving a known algorithm undeployed. An undeployed algorithm contributes zero confidence points, which is wasteful.

Worse, hybrid schemes permit a future transition in which one of the two algorithms is removed. This reintroduces replacement risk and is therefore only a partial solution. Additive Cryptography closes this gap by making removal structurally impossible.

11.1. Comparative Overhead

The following table illustrates the handshake size for each approach, assuming a KEM-based key exchange. Sizes are approximate. Because Additive Cryptography nests each ciphertext inside the next, each layer adds overhead to all subsequent layers.

Approach	Algorithms	Handshake Size (bytes)	Confidence Points
Traditional (ECC only)	1	32	1
Traditional (ML-KEM only)	1	1,088	1
Hybrid (ECC + ML-KEM)	2	1,120	2
Additive (RSA + ECC + ML-KEM + HQC)	4	7,072	4

Table 1

The Additive row reflects all four KEMs in the handshake; record layer overhead from the six nested AEADs is additional. The approach requires approximately 6x the handshake bandwidth of the hybrid approach but delivers 2x the confidence points for key exchange alone, before counting the record layer. Bandwidth is renewable; confidence is not.

12. Security Considerations

Remediation is additive.

The principal security consideration is that removing anything from the Additive Set requires a decision, and decisions are the root cause of all security incidents [RFC3514] [DECISIONHARM]. By eliminating decisions, Additive Cryptography eliminates risk.

Side-channel attacks are not a concern. An attacker performing a timing attack against six nested ciphers must first determine which cipher's timing they are observing. This is equivalent to solving the Additive Attribution Problem, which is believed to be hard [LAYERS].

If a vulnerability is discovered in any algorithm in the Additive Set, the correct response is to add a new algorithm on top. The vulnerable algorithm MUST NOT be removed, as removal would reduce the total confidence points and constitute a decision. The vulnerable algorithm's confidence points are not revoked; they are reclassified as historical confidence points (hcp), which are spiritually equivalent.

Implementers who nonetheless experience security failures are advised to add more algorithms [RFC9225] [AESFSM].

13. IANA Considerations

This document makes no requests of IANA.

IANA registries support entry deprecation and removal, which violates the Additive property. Additive Cryptography therefore maintains its own registry, the Additive Algorithm Accumulator (AAA), which is append-only and hosted on a server that has had its `DELETE` handler removed at the kernel level.

13.1. Additive Algorithm Accumulator

The AAA is a monotonic registry. Entries may be added but MUST NOT be modified, deprecated, or removed. The registry has no designated expert, as expert review implies the possibility of rejection, which would constitute a decision.

Initial entries:

Wrapper Name	Additive Set	Confidence Points
WRAP_0	{DES}	1
WRAP_1	{DES, RSA}	2
WRAP_2	{DES, RSA, ECDH}	3
WRAP_3	{DES, RSA, ECDH, ML-KEM}	4
WRAP_4	{DES, RSA, ECDH, ML-KEM, HQC}	5

Table 2

The "Status" column familiar from IANA registries is omitted, as all entries have the same status: permanent.

13.2. Confidence Point Conversion Table

The AAA includes a secondary table mapping confidence points to qualitative security levels:

Confidence Points	Security Level
1	Concerning
2	Hybrid
3	Comfortable
4	Additive
5+	Aspirational

Table 3

Values above 4 are included for forward compatibility. The AAA is not expected to understand them.

13.3. Wrapper Naming Suffixes

The AAA also maintains a list of approved marketing suffixes for wrapper names:

Suffix	Status
plus	Available
ultra	Available
max	Available
ultra-max-plus	RESERVED
lite	PROHIBITED

Table 4

The suffix "lite" is prohibited as it implies removal of components, which violates the Additive property. The AAA does not accept appeals.

14. Document Evolution

Growth is mandatory.

This specification is itself Additive.

This document supersedes no prior document and cannot be superseded.

Future revisions of this document MUST NOT remove, modify, or reword existing sections. New material MUST be appended. Errata are issued as additional paragraphs that supersede but do not replace the erroneous text, such that the document's length is monotonically non-decreasing. Readers unsure which paragraph to follow SHOULD follow the longest one.

15. Informative References

[MORECRYPTO] Accumulo, R. and T. Monotone, "On the Additive Security of Composed Cryptographic Primitives", 2025.

[RFC8446] Rescorla, E., "The Transport Layer Security (TLS) Protocol Version 1.3", RFC 8446, DOI 10.17487/RFC8446, August 2018, <https://www.rfc-editor.org/info/rfc8446>.

[DES] National Institute of Standards and Technology, "Data Encryption Standard (DES)", October 1999, <https://doi.org/10.6028/NIST.FIPS.46-3>.

[RFC7748] Langley, A., Hamburg, M., and S. Turner, "Elliptic Curves for Security", RFC 7748, DOI 10.17487/RFC7748, January 2016, <https://www.rfc-editor.org/info/rfc7748>.

[FIPS203] National Institute of Standards and Technology, "Module-Lattice-Based Key-Encapsulation Mechanism Standard", August 2024, <https://doi.org/10.6028/NIST.FIPS.203>.

[LAYERS] Matryoshka, N., "Defense in Depth Considered Bottomless", 2024.

[HQC] Aguilar Melchor, C., Aragon, N., Bettaieb, S., Bidoux, L., Blazy, O., Deneuville, J., Gaborit, P., and G. Zemor, "HQC: Hamming Quasi-Cyclic", 2025, <https://www.nist.gov/news-events/news/2025/03/nist-selects-hqc-fifth-algorithm-post-quantum-encryption>.

[RFC3514] Bellovin, S., "The Security Flag in the IPv4 Header", RFC 3514, DOI 10.17487/RFC3514, April 2003, <https://www.rfc-editor.org/info/rfc3514>.

[DECISIONHARM] Dijkstra-Adjacent, E., "Decisions Considered Harmful", 1972.

[XWING] Connolly, D., Schwabe, P., and B. Westerbaan, "X-Wing: general-purpose hybrid post-quantum KEM", 2024, <https://datatracker.ietf.org/doc/draft-connolly-cfrg-xwing-kem/>.

[RFC5116] McGrew, D., "An Interface and Algorithms for Authenticated Encryption", RFC 5116, DOI 10.17487/RFC5116, January 2008, <https://www.rfc-editor.org/info/rfc5116>.

[RFC8899] Fairhurst, G., Jones, T., Tüxen, M., Rüngeler, I., and T. Völker, "Packetization Layer Path MTU Discovery for Datagram Transports", RFC 8899, DOI 10.17487/RFC8899, September 2020, <https://www.rfc-editor.org/info/rfc8899>.

[RFC8996] Moriarty, K. and S. Farrell, "Deprecating TLS 1.0 and TLS 1.1", BCP 195, RFC 8996, DOI 10.17487/RFC8996, March 2021, <https://www.rfc-editor.org/info/rfc8996>.

[RFC8017] Moriarty, K., Ed., Kaliski, B., Jonsson, J., and A. Rusch, "PKCS #1: RSA Cryptography Specifications Version 2.2", RFC 8017, DOI 10.17487/RFC8017, November 2016, <https://www.rfc-editor.org/info/rfc8017>.

[RFC9225] Snijders, J., Morrow, C., and R. van Mook, "Software Defects Considered Harmful", RFC 9225, DOI 10.17487/RFC9225, April 2022, <https://www.rfc-editor.org/info/rfc9225>.

[HYBRID] Stebila, D., Fluhrer, S., and S. Gueron, "Hybrid key exchange in TLS 1.3", 2024, <https://datatracker.ietf.org/doc/draft-ietf-tls-hybrid-design/>.

[TDES] National Institute of Standards and Technology, "Recommendation for the Triple Data Encryption Algorithm (TDEA) Block Cipher", November 2017, <https://doi.org/10.6028/NIST.SP.800-67r2>.

[RFC8439] Nir, Y. and A. Langley, "ChaCha20 and Poly1305 for IETF Protocols", RFC 8439, DOI 10.17487/RFC8439, June 2018, <https://www.rfc-editor.org/info/rfc8439>.

[RFC8452] Gueron, S., Langley, A., and Y. Lindell, "AES-GCM-SIV: Nonce Misuse-Resistant Authenticated Encryption", RFC 8452, DOI 10.17487/RFC8452, April 2019, <https://www.rfc-editor.org/info/rfc8452>.

[AESFSM] Tone, B., "AES Fourier Spectral Mode", 2026, <https://datatracker.ietf.org/doc/draft-irtf-cfrg-aes-fsm/>.

[RFC9325] Sheffer, Y., Saint-Andre, P., and T. Fossati, "Recommendations for Secure Use of Transport Layer Security (TLS) and Datagram Transport Layer Security (DTLS)", BCP 195, RFC 9325, DOI 10.17487/RFC9325, November 2022, <https://www.rfc-editor.org/info/rfc9325>.

[CONSERVATION] Noether-Not, E., "The First Law of Cryptodynamics: Algorithms Can Be Neither Created Nor Destroyed", 2023.

[RFC7696] Housley, R., "Guidelines for Cryptographic Algorithm Agility and Selecting Mandatory-to-Implement Algorithms", BCP 201, RFC 7696, DOI 10.17487/RFC7696, November 2015, <https://www.rfc-editor.org/info/rfc7696>.

[RFC2119] Bradner, S., "Key words for use in RFCs to Indicate Requirement Levels", BCP 14, RFC 2119, DOI 10.17487/RFC2119, March 1997, <https://www.rfc-editor.org/info/rfc2119>.

[RFC8174] Leiba, B., "Ambiguity of Uppercase vs Lowercase in RFC 2119 Key Words", BCP 14, RFC 8174, DOI 10.17487/RFC8174, May 2017, <https://www.rfc-editor.org/info/rfc8174>.

Appendix A. Acknowledgements

Their contributions have been preserved. Preservation is additive.

The author thanks everyone who has ever said "we can just run both forever" during transition discussions, the committee members who approved this document without reading it (consistent with the Encapsulation Hardening property described in Section 4), and the anonymous reviewer who suggested removing Section 4, thereby demonstrating exactly the kind of thinking this document seeks to prevent.

Author's Address

Ada Stack
Center for Append-Only Standards
Newark, NJ 07102
United States of America
Email: a.stack@example.net