Workgroup: TSVWG
Published: 13 December 2021
Intended Status: Standards Track
Expires: 16 June 2022
Authors: R. Wang L. Si B. He
Agora Lab Agora Lab Agora Lab

Sliding Window Selective Linear Code (SLC) Forward Error Correction (FEC) Scheme for FECFRAME

Abstract

RFC8680 describes a framework for using Sliding Window Forward Error Correction(FEC) codes to protection against packet loss, the framework significantly improves FEC efficiency and reduces FEC-related added latency compared to block FEC codes defined in RFC 6363. RFC8681 further describes two fully specified FEC schemes for Sliding Window Random Linear Codes(RLC), the schemes rely on an encoding window that slides over a continuous set of source symbols, generating new repair symbols whenever needed. This document describes a fully specified FEC scheme for Sliding Window Selective Linear Code(SLC) over the Galois Field GF (2^^8) , compared to RFC8681, this framework use a discrete encoding window which can protect arbitrary media streams selectively, and has better recovery performance in scenarios such as layered video coding or mixed streams for video streaming applications.

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 16 June 2022.

Copyright Notice

Table of Contents

1. Introduction

The use of Application-Level Forward Erasure Correction (AL-FEC) codes is a widely-used error control method used to improve the reliability of unicast, multicast, and broadcast transmissions.

The [RFC5052] document describes a general framework to use FEC in Content Delivery Protocols (CDPs), and it is suitable for FEC schemes based on building blocks. Based on this framework, the [RFC5170] describes two fully-specified FEC Schemes, Low-Density Parity Check (LDPC) Staircase and LDPC Triangle, and the [RFC5510] describes one Fully-Specified FEC Scheme for the special case of Reed-Solomon (RS) over GF (2^^8).

The [RFC6363] document describes a general framework used to protect arbitrary media streams along the lines defined by FECFRAME. The FEC scheme defined by the framework does not limit the type of input data, but only processes the data.

Similar to [RFC5052], [RFC6363] only considers block FEC schemes, which requires that the input stream be divided into a series of blocks according to the block partitioning algorithm defined in [RFC5052]. The [RFC6681], [RFC6816], and [RFC6865] are FEC schemes based on this framework. The value for the block size affects the packet loss resistance and the encoding and decoding delay of the FEC scheme. At the same code rate, the FEC scheme with larger size blocks have higher robustness (e.g., in case of long packet erasure bursts), but it has higher decoding delay which is unacceptable for real-time video streaming application.

The framework described in [RFC8680] provides support for FEC codes based on a sliding coding window. The FEC scheme in this framework [RFC8681] is advantageous for real-time flows because of its high robustness and low additional delay.

In general video coding, all frames in a GOP follow the rule of the frame by frame reference, that is, the reconstruction of the current video frame relies on the preceding frame. In that case, all frames in the encoding window are beneficial to the decoding of the current frame. However, for layered video coding, video frames may not reference the preceding frames, but the upper layer frames. When non-reference frames are encoded, the recovered packets will not help the decoding of the current frame, and even have a negative effect on the FEC error correction ability in extreme cases.

This document introduces one fully specified FEC scheme, it is capable to protect streams selectively by adding a filter into the

FEC coding window management. The Sliding Window SLC FEC scheme described in this document belongs to the broad class of Sliding Window AL-FEC Codes (a.k.a., convolutional codes) [RFC8406]. The encoding process is based on an encoding window, and the source symbols are encoded by sliding the encoding window. However, the encoding window does not slide directly over the set of the source symbols. Instead, it filters the source symbols according to the rule defined by application (e.g., video frame dependency, or stream type) and then slide over the set of these filtered source symbols. Repair symbols are generated on-the-fly, by the computation of a linear combination of source symbols present in the current encoding window and passed to the transport layer.

When the loss of source symbol is detected at the receiver, the SLC decoder will recover the lost source symbol according to the linear combination of the source symbols and each received repair symbol (when the rank of the equations involved is solvable).

This fully-specified FEC scheme follows the structure required by [RFC6363], Section 5.6 ("FEC Scheme Requirements"), namely:

* Formats and Codes: This section defines the FEC Framework Configuration Information (FFCI) carrying signaling, including mandatory elements and Scheme-Specific elements. It also defines the Source FEC Payload ID and Repair FEC Payload ID formats, carrying the signaling information associated with each source or repair symbol, including ESI, indexes of source symbols participating in encoding, and coding coefficients.

* Procedures: This section describes procedures specific to this FEC scheme, including encoding window management, coding matrix generation, a linear combination of source symbol computation in Finite Field, and the mapping between ADU, ADUI, and Source Symbols.

* FEC Code Specification: This section provides a high-level description of the Sliding Window SLC encoder and decoder.

2. Terminology

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 [RFC2119] [RFC8174].

3. Definitions Notations and Abbreviations

3.1. Definitions

This document uses the following terms and definitions. Some of these terms and definitions are FEC scheme-specific and are in line with [RFC5052] [RFC6363]:

Source symbol: unit of data used during the encoding process.

Encoding symbol: unit of data generated by the encoding process.

Repair symbol: an encoding symbol that is not a source symbol.

Packet erasure channel: a communication path where packets are either dropped (e.g., by a congested router, or because the number of transmission errors exceeds the correction capabilities of the physical layer codes) or received. When a packet is received, it is assumed that this packet is not corrupted.

Application Data Unit (ADU): unit of source data provided as payload to the transport layer. Depending on the use case, an ADU may use an RTP encapsulation.

ADU Information (ADUI): unit of data constituted by the ADU and the associated Flow ID, Length and Padding fields.

FEC Framework Configuration Information (FFCI): information that controls the operation of FECFRAME. Each FEC Framework instance has its own configuration information. And the FFCI enables the synchronization of the FECFRAME sender and receiver instances.

FEC Source Packet: at a sender (respectively, at a receiver) a payload submitted to (respectively, received from) the transport protocol containing an ADU along with an Explicit Source FEC Payload ID.

FEC Repair Packet: at a sender (respectively, at a receiver) a payload submitted to (respectively, received from) the transport protocol containing one repair symbol along with a Repair FEC Payload ID and possibly an RTP header.

3.2. Notations

This document uses the following notations and some of them are FEC scheme-specific:

m: defines the length of the elements in the finite field, in bits. In this document, m is such that m=8.

GF(q):
denotes a finite field (also known as the Galois Field) with q elements. We assume that q = 2^^m in this document.

a^^b: denotes a raised to the power b.

E: denotes the size of an encoding symbol length in bytes.

cw_size: denotes coding window size (in symbols).

cw_size_max: denotes coding window maximum size (in symbols).

gc: denotes the count of symbol groups participating in encoding (if there is a gap in the serial number, it is considered a new group) when a repair symbol is generated.

gc_max: denotes the maximum count of symbol groups involved in encoding when generating maintenance symbols.

cm: denotes coding matrix.

cm_r: denotes row in the coding matrix.

cm_c: denotes col in the coding matrix.

ESI: denotes the first source symbol of the ADUI corresponding to this FEC Source Packet.

Start_ESI: denotes the first ADUI's ESI of the first group.

Residual_ESI_: denotes the residual value of the starting ESI of the current group relative to the previous group.

Group_Size_: denotes the number of ADUIs contained in each group.

3.3. Abbreviations

This document uses the following abbreviations, and some of them are FEC scheme-specific:

FEC: stands for Forward Error (or Erasure) Correction codes.

ADU: stands for Application Data Unit.

ADUI: stands for Application Data Unit Information.

ESI: stands for Encoding Symbol ID.

FFCI: stands for FEC Framework Configuration Information.

FSSI: stands for FEC Scheme-Specific Information.

4. Formats and Codes

This section describes the format of FEC Framework Configuration Information (or FFCI) and FEC Payload IDs, which are carried in "big-endian" or "network order" format.

4.1. FEC Framework Configuration Information

The FFCI needs to be shared between FECFRAME sender and receiver instances to ensure the synchronization of information. It includes mandatory elements (e.g., FEC Encoding ID) and scheme-specific elements (e.g., Encoding Symbol size).

4.1.1. Mandatory

FEC Encoding ID: the value assigned to this Fully-Specified FEC scheme **MUST** be XXX, as assigned by IANA(Section 9).

4.1.2. FEC Scheme-Specific Information

The FEC scheme-specific information (FSSI) of this scheme is as follows:

Encoding Symbol size (E): a non-negative integer that indicates the size of each encoding symbol in bytes;

The maximum coding window size (cw_size_max): a non-negative integer that indicates the maximum size of the coding window allowed (in symbols);

The maximum number of gc (gc_max): a non-negative integer that indicates the maximum count of groups protected by each repair packet.

These elements are required both by the encoder and decoder.

When SDP is used to communicate the FFCI, this FEC Scheme-Specific Information **MUST** be carried in the 'fssi' parameter in textual representation specified in [RFC6364]. For instance:

```
fssi=E:1500,cw_size_max:128,gc_max:4
```

If another mechanism requires the FSSI to be carried as an opaque octet string (for instance after a Base64 encoding), the encoding format consists of the following four octets:

Encoding symbol length (E): 16-bit field;

Maximum coding window size (cw_size_max):
8-bit field;

Maximum size of gc (gc_max): 8-bit field.

The encoding format consists of the following 4 octets of Figure 1:

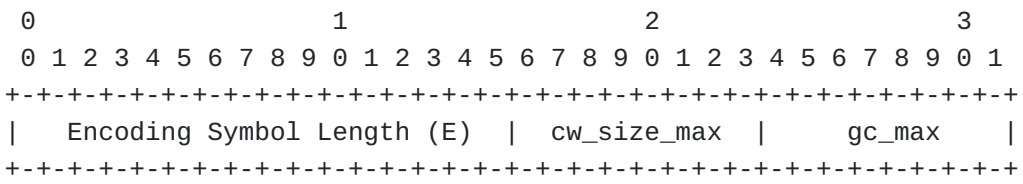

Figure 1: FSSI Encoding Format

4.2. FEC Payload IDs

4.2.1. Explicit Source FEC Payload ID

A FEC Source Packet **MUST** contain an Explicit Source FEC Payload ID that is appended to the end of the packet as illustrated in Figure 2.

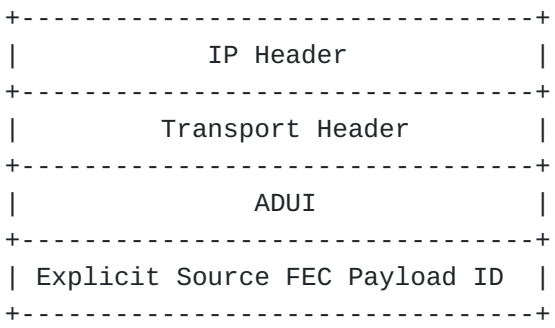

Figure 2: Structure of an FEC Source Packet with the Explicit Source FEC Payload ID

More precisely, the Explicit Source FEC Payload ID is composed of the following field:

Encoding Symbol ID (ESI) (32-bit field): this unsigned integer identifies the first source symbol of the ADUI corresponding to this FEC Source Packet. The ESI is incremented for each new source symbol, and after reaching the maximum value (2^^32-1), wrapping to zero occurs.

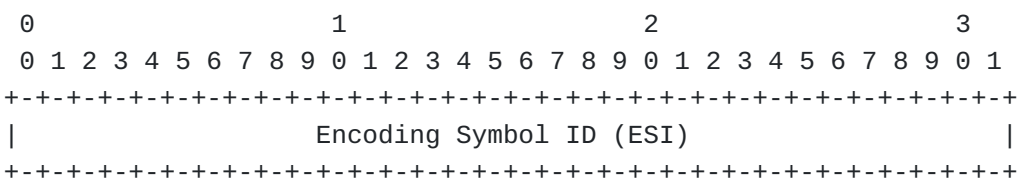

Figure 3: Source FEC Payload ID Encoding Format

4.2.2. Repair FEC Payload ID

A FEC repair packet **MUST** contain a Repair FEC Payload ID prepended to the repair symbol as illustrated in Figure 4. There **MUST** be a single repair symbol per FEC repair packet.

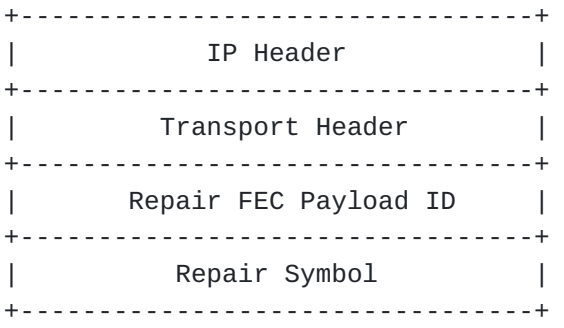

Figure 4: Structure of an FEC Repair Packet with the Repair FEC Payload ID

More precisely, the SLC decoder scheme require the following information from the Repair FEC Payload ID:

Start_ESI (32-bit field): this unsigned integer indicates the ESI of the first source symbol of the first group in the encoding window when this repair symbol was generated.

gc (8-bit field): this unsigned integer indicates the number of symbol groups in the encoding window when this repair symbol is generated (if there is a gap in the serial number, it is considered a new group).

cm_r (8-bit field): this unsigned integer is used as a parameter to generate the desired encoding matrix. This cm_r **MUST NOT** be greater than cw_size_max.

Residual_ESI_ (8-bit field): this unsigned integer represents the residual value of the starting ESI of the current group relative to the previous group.

Group_Size_ (8-bit field): this unsigned integer is the number of the source symbols contained in each group.

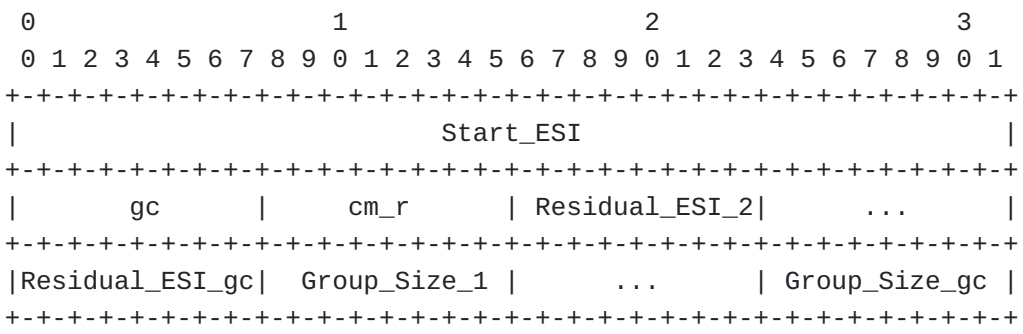

Figure 5: Repair FEC Payload ID Encoding Format

The length of the Repair FEC Payload ID depends on the gc parameter.

5. Procedures

5.1. Restrictions

This specification has the following restrictions:

1. There **MUST** be exactly one source symbol per ADUI, and therefore per ADI;

2. There **MUST** be exactly one repair symbol per FEC Repair Packet.

5.2. ADU, ADUI, and Source Symbols Mappings

Before FEC coding, the mapping from ADU to AUDI needs to be established. When multiple source flows (e.g., media streams) are mapped onto the same FECFRAME instance, each flow is assigned its Flow ID value. The Flow ID needs to be included in the ADUI. Then, the recovered ADU can be allocated to the corresponding source flow by its Flow ID.

Because the length of each ADU may be inconsistent, to ensure that the decoder can extract ADU from ADUI, the original ADU length also needs to be added to ADUI.

For each incoming ADU, an ADUI MUST be created as follows. First of all, 3 bytes are prepended (Figure 6):

Flow ID (F) (8-bit field): this unsigned byte contains the integer identifier associated with the source ADU flow to which this ADU belongs.

Length (L) (16-bit field): this unsigned integer contains the length of this ADU in network byte order (i.e., big endian). This length is for the ADU itself and does not include the F, or Pad fields.

Then, zero padding is added to the ADU if needed:

Padding (Pad) (variable size field): this field is used for alignment purposes up to a size of exactly E bytes.

The data unit resulting from the ADU and the F, L, and Pad fields is called ADUI. An ADUI always contributes to an integral number of source symbols.

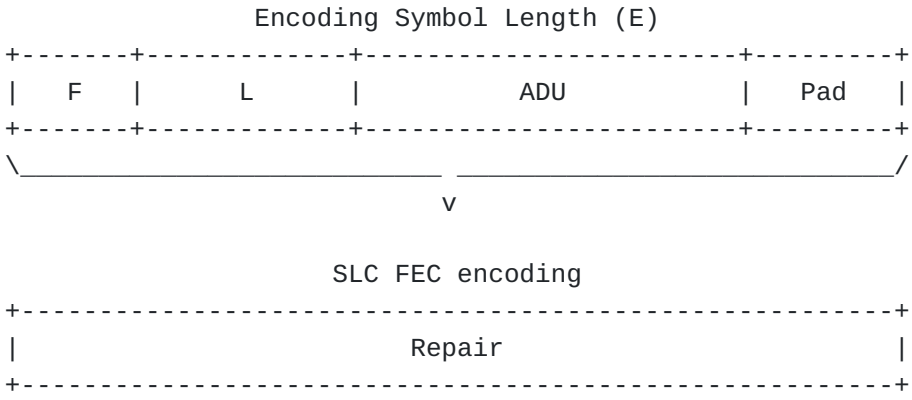

Figure 6: ADUI Creation Example

5. Encoding Window Management

Whenever an ADU arrives, ADU-to-source symbols mapping will be performed. Then, the source symbols will be added to the array source_symbol_history. Whenever a repair symbol needs to be generated, the SLC FEC encoder will search backward in the source_symbol_history, and the source symbols that conforms the rules defined by the application will be put into the encoding window. When the encoding window cw_size is equal to its maximum value cw_size_max or the symbol group count gc is equal to its maximum value gc_max, the search is stopped and the FEC coding will be performed on the source symbols in the encoding window.

Taking Figure 7 as an example, the coding dependency between frames is used as the rule of source symbol selection, and frame I is the reference frame of frame P1, so I and P1 are placed in the encoding window when generating Repair2. However, P1 is not the reference frame of P2 under the SVC mode, so P1 is skipped, I and P2 are put into the encoding window to generate Repair3. The same process is performed to produce Rapair4 and Repair5.

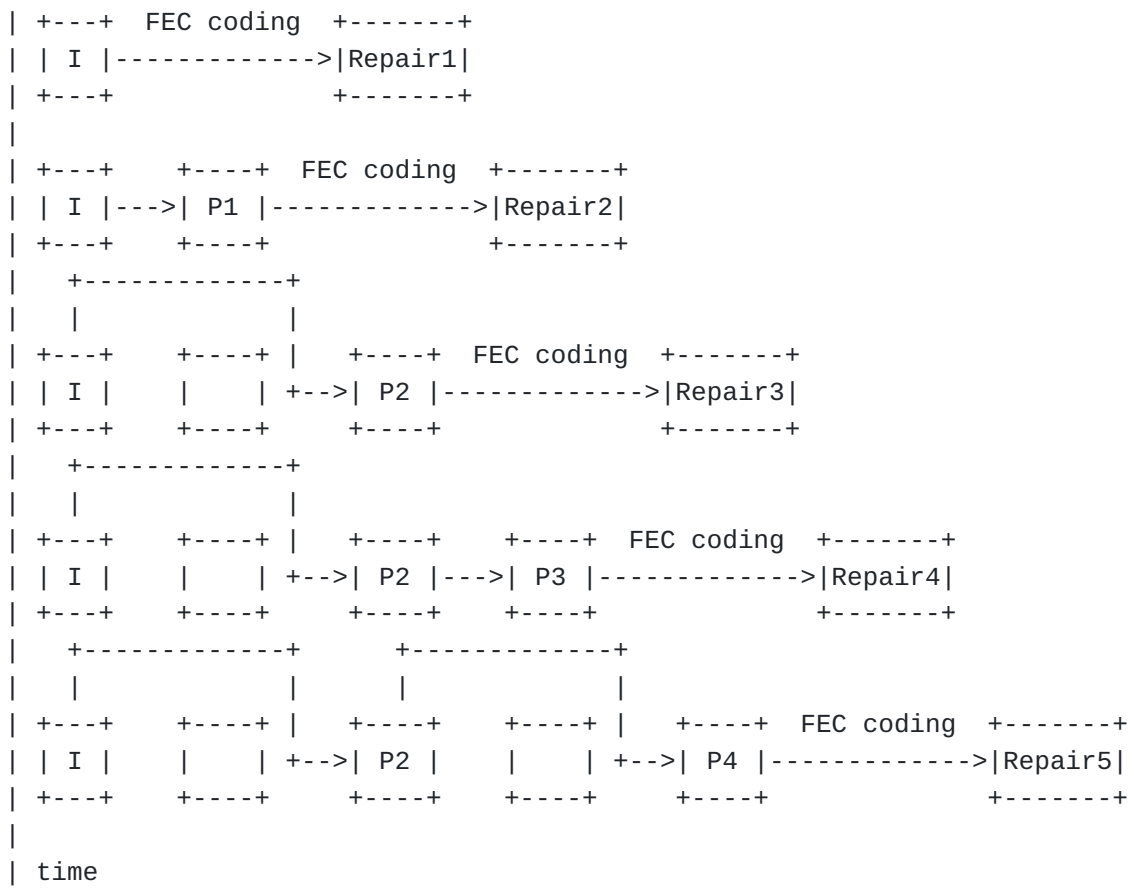

Figure 7: Example of Encoding Window Management

Note that each time the encoding window slides, cm_r will update. The update rules are as follows:

```
if (++cm_r>=cw_size_max) cm_r=0;
```

5.4. Coding Matrix Generation

Compared with the RLC FEC encoder, which depends on a pseudorandom number generator to compute the coding coefficients, the SLC FEC encoder uses a fixed coding matrix to reduce overhead. The elements of the coding matrix used during the encoding process are generated at the SLC FEC encoder each time a new repair symbol needs to be produced. The cm_r and cm_c parameters control these elements. The values of cm_c between 0 (the minimum value) and cw_size_max-1 (the maximum value). And the values of cm_r between 0 and 255-cw_size_max.

```
G (cm_r, cm_c) = y_c / (x_r + y_c) = (cm_c + cw_size_max) / (cm_r
+ cm_c + cw_size_max)
```

where cm_r represents the row number in the matrix, cm_c represents the col number in the matrix, cw_size_max represents the maximum

value of the encoding window, x_r = cm_r, y_c = cw_size_max+cm_c. The basic operations of the above equations are carried out in the GF (2^^8).

5.5. Linear Operation on encoding side and decoding side

5.5.1. Encoding Side

In Section 5.4, the elements of coding matrix G(cm_r, cm_c) are obtained. Then, a repair symbol is generated by the computation of a linear combination of source symbols.

A linear combination of the cw_size source symbols present in the encoding window, say src_0 to src_cw_size_1, is computed as follows. For each byte of position i in each source and the repair symbol, where i belongs to [0; E-1].

```
repair[i] = G(cm_r, 0) * src_0[i] + G(cm_r, 1) * src_1[i] + ... +
G(cm_r, cw_size-1) * src_cw_size_1[i]
```

where * is the multiplication over GF (2^^8), + is the addition over GF (2^^8). In this document, the following irreducible polynomial is used for GF(2^^8).

```
x^^8 + x^^4 + x^^3 + x^^2 + 1
```

5.5.2. Decoding Side

For decoding side, it is assumed that the repair symbol protects cw_size source symbols, among which j source symbols are lost, then,

```
remove_src[i] = repair[i] - G(cm_r, 0) * src_0[i] - ... - G(cm_r,
k) * src_k[i] - G(cm_r, k + j + 1) * src_k_j_1[i] - ... - G(cm_r,
cw_size-1) * src_cw_size_1[i]
```

It is assumed that in the linear system maintained by the decoding side, there is a symbol sequence S = {lost_src_1, lost_src_2, ... , lost_src_N} consisting of N lost source symbols, a symbol sequence R = {repair_1, repair_2, ... , repair_N} consisting of N repair symbols

There is a matrix A whose row represents the position of the repair symbol in R and whose column represents the position of the lost source symbol in S. A[row][col] represents the matrix element of lost_src_row corresponding to repair_col (if it does not exist, then A[row][col] = 0).

```
A[row][col] = G(cm_r,cm_c)
```

where cm_r can be extracted from the Repair FEC Payload ID, cm_c represents the position of the lost source symbol in the encoding window.

Therefore, there is a linear system of equation as follows:

```
A * Transpose(lost_src_1, lost_src_2, ... , lost_src_N) =
Transpose(remove_src_1, remove_src_2, ... , remove_src_N)
```

The inverse matrix of A can be obtained by Gauss elimination method, and finally S can be recovered:

```
Transpose(lost_src_1, lost_src_2, ... , lost_src_N) = A^^-1 *
Transpose(remove_src_1, remove_src_2, ... , remove_src_N)
```

6. FEC Code Specification

6.1. Encoding Side

1. Whenever a new repair symbol needs to be produced, the source symbols are put into the sliding encoding window according to the rule defined by application (e.g., coding dependency between frames).

2. The SLC FEC encoder gathers the cw_size source symbols currently in the sliding encoding window.

3. The elements of the coding matrix are determined according to the parameters cm_r and cm_c (Section 5.4).

4. The SLC FEC encoder computes the repair symbol by a linear combination of the cw_size source symbols present in the encoding window using the coding matrix (Section 5.5.1).

When encoding, the execution object is ADUI composed of Flow ID, Length, ADU, Padding.

6.2. Decoding Side

1. A linear system composed of source symbols, elements of the coding matrix, and repair symbols **MUST** to be maintained to recover the lost source packets.

2. When a repair symbol is received, it detects whether there is loss in the protected source symbols. If at least one of the corresponding source symbols has been lost, an equation composed of the repair symbol, the corresponding source symbols, and the elements of the coding matrix will be added to the linear system (the elements of the coding matrix are generated by the method provided in Section 5.4).

3. When the linear system covering one or more lost source symbols is full, decoding is performed in order to recover lost source symbols (Section 5.5.2).

4. Each time an ADUI can be totally recovered, padding is removed (thanks to the Length field, L, of the ADUI), and the ADU will be assigned to the corresponding flow.

Note that the recovered source symbols can be directly passed to the application through the callback function, or passed to the application after receiving a certain number of source symbols, which depends on the operation decision of the application.

7. Security Considerations

The FEC Framework document [RFC6363] provides a comprehensive analysis of security considerations applicable to FEC schemes. Therefore, the present section follows the security considerations section of [RFC6363] and only discusses specific topics.

7.1. Attacks Against the Data Flow

7.1.1. Access to Confidential Content

The Sliding Window SLC FEC scheme specified in this document does not change the recommendations of [RFC6363]. To summarize, if confidentiality is a concern, it is **RECOMMENDED** that one of the solutions mentioned in [RFC6363] is used with special considerations to the way this solution is applied (e.g., is encryption applied before or after FEC protection, within the end system or in a middlebox), to the operational constrains (e.g., performing FEC decoding in a protected environment may be complicated or even impossible) and to the threat model.

7.1.2. Content Corruption

The Sliding Window SLC FEC scheme specified in this document does not change the recommendations of [RFC6363]. To summarize, it is **RECOMMENDED** that one of the solutions mentioned in [RFC6363] is used on both the FEC Source and Repair Packets.

7.2. Attacks Against the FEC Parameters

The Sliding Window SLC FEC scheme specified in this document defines parameters that can be the basis of attacks. More specifically, the following parameters of the FEC Framework Configuration Information may be modified by an attacker (Section 4.1):

FEC Encoding ID: changing this parameter leads the receiver to consider a different FEC Scheme. It will lead to severe

consequences that the format of the AUDI, the Explicit Source FEC Payload ID, and Repair FEC Payload ID of received packets will probably differ. The FEC decoder can't get the correct decoding information, resulting in decoding failure or decoding error.

Encoding symbol length (E): setting this E parameter to a different value will enable an attacker to create a DoS since the repair symbols and certain source symbols will be larger or smaller than E, incoherency for the receiver.

Therefore, it is **RECOMMENDED** that security measures be taken to guarantee the FFCI integrity, as specified in [RFC6363]. How to achieve this depends on how the FFCI is communicated from the sender to the receiver, which is not specified in this document.

Similarly, attacks are possible against the Explicit Source FEC Payload ID and Repair FEC Payload ID. More specifically, in the case of an FEC Source Packet, the following value can be modified by an attacker who targets receivers:

Encoding Symbol ID (ESI): changing the ESI leads a receiver to consider a wrong ADU, resulting in severe consequences, including corrupted content passed to the receiving application. And in the case of an FEC Repair Packet.

Start_ESI: changing this value causes the FEC decoder to add the wrong source symbol in the linear system, and therefore any source symbol recovered by the linear system may be wrong.

gc: changing this value causes the FEC decoder to add an incorrect number of source symbols in the linear system. Therefore any source symbol recovered by the linear system may be wrong.

cm_r: changing this value leads a receiver to generate a wrong coding coefficient, and therefore any source symbol decoded using the repair symbol contained in this packet will be corrupted.

Residual_ESI_: changing this value causes the FEC decoder to add the wrong source symbol in the linear system, and therefore any source symbol recovered by the linear system may be wrong.

Group_Size_: changing this value causes the FEC decoder to add an incorrect number of source symbols in the linear system. Therefore any source symbol recovered by the linear system may be wrong.

Therefore, it is **RECOMMENDED** that security measures are taken to guarantee the FEC Source and Repair Packets as stated in [RFC6363].

7.3. When Several Source Flows Are to Be Protected Together

The Sliding Window SLC FEC scheme specified in this document does not change the recommendations of [RFC6363].

7.4. Baseline Secure FECFRAME Operation

The Sliding Window SLC FEC scheme specified in this document does not change the recommendations of [RFC6363] concerning the use of the IPsec/Encapsulating Security Payload (ESP) security protocol as a mandatory-to-implement (but not mandatory-to-use) security scheme. This is well suited to situations where the only insecure domain is the one over which the FEC Framework operates.

8. Operations and Management Considerations

The FECFRAME document [RFC6363] provides a comprehensive analysis of operations and management considerations applicable to FEC schemes. Therefore, the present section only discusses specific topics.

8.1. Operational Recommendations: gc_max

The Sliding Window SLC FEC scheme specified in this document defines the maximum number of groups participating in encoding, called gc_max, reflecting the maximum number of source symbols that the coding window can hold. Gc_max is directly proportional to the computational complexity of FEC encoding. If gc_max is too large, the time complexity of FEC encoding will be too high, and the CPU overhead will be too large. Generally, it is appropriate to associate gc_max with cw_size_max.

For example, in real-time video streaming applications, the frame rate (FR) and bit rate (BR) is determined by transmitting the video frames. The possible number of packets per frame can be calculated according to FR and BR, and they can calculate the maximum number of symbols in the coding window.

```
BR kbps / 8 / FR fps / MTU * gc_max <= cw_size_max
```

Where MTU denotes Maximum Transmission Unit.

9. IANA Considerations

This document registers one values in the "FEC Framework (FECFRAME) FEC Encoding IDs" sub-registry as follows:

XXX refers to the Sliding Window Selective Linear Code (SLC) Forward Error Correction (FEC) Scheme for Arbitrary Packet Flows.

10. Acknowledgments

The authors would like to thank the FEC Framework Design Team for providing a great FEC Framework. The authors would also like to thank Shie Qian for reviewing the earlier draft versions of this document.

11. References

11.1. Normative References

[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>.

[RFC5052] Watson, M., Luby, M., and L. Vicisano, "Forward Error Correction (FEC) Building Block", RFC 5052, DOI 10.17487/RFC5052, August 2007, <https://www.rfc-editor.org/info/rfc5052>.

[RFC6363] Watson, M., Begen, A., and V. Roca, "Forward Error Correction (FEC) Framework", RFC 6363, DOI 10.17487/RFC6363, October 2011, <https://www.rfc-editor.org/info/rfc6363>.

[RFC6364] Begen, A., "Session Description Protocol Elements for the Forward Error Correction (FEC) Framework", RFC 6364, DOI 10.17487/RFC6364, October 2011, <https://www.rfc-editor.org/info/rfc6364>.

[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>.

[RFC8680] Roca, V. and A. Begen, "Forward Error Correction (FEC) Framework Extension to Sliding Window Codes", RFC 8680, DOI 10.17487/RFC8680, January 2020, <https://www.rfc-editor.org/info/rfc8680>.

11.2. Informative References

[RFC5170] Roca, V., Neumann, C., and D. Furodet, "Low Density Parity Check (LDPC) Staircase and Triangle Forward Error Correction (FEC) Schemes", RFC 5170, DOI 10.17487/RFC5170, June 2008, <https://www.rfc-editor.org/info/rfc5170>.

[RFC5510] Lacan, J., Roca, V., Peltotalo, J., and S. Peltotalo, "Reed-Solomon Forward Error Correction (FEC) Schemes",

RFC 5510, DOI 10.17487/RFC5510, April 2009, <https://www.rfc-editor.org/info/rfc5510>.

[RFC6681] Watson, M., Stockhammer, T., and M. Luby, "Raptor Forward Error Correction (FEC) Schemes for FECFRAME", RFC 6681, DOI 10.17487/RFC6681, August 2012, <https://www.rfc-editor.org/info/rfc6681>.

[RFC6816] Roca, V., Cunche, M., and J. Lacan, "Simple Low-Density Parity Check (LDPC) Staircase Forward Error Correction (FEC) Scheme for FECFRAME", RFC 6816, DOI 10.17487/RFC6816, December 2012, <https://www.rfc-editor.org/info/rfc6816>.

[RFC6865] Roca, V., Cunche, M., Lacan, J., Bouabdallah, A., and K. Matsuzono, "Simple Reed- Solomon Forward Error Correction (FEC) Scheme for FECFRAME", RFC 6865, DOI 10.17487/RFC6865, February 2013, <https://www.rfc-editor.org/info/rfc6865>.

[RFC8406] Adamson, B., Adjih, C., Bilbao, J., Firoiu, V., Fitzek, F., Ghanem, S., Lochin, E., Masucci, A., Montpetit, M-J., Pedersen, M., Peralta, G., Roca, V., Saxena, P., and S. Sivakumar, "Taxonomy of Coding Techniques for Efficient Network Communications", RFC 8406, DOI 10.17487/RFC8406, June 2018, <https://www.rfc-editor.org/info/rfc8406>.

[RFC8681] Roca, V. and B. Teibi, "Sliding Window Random Linear Code (RLC) Forward Erasure Correction (FEC) Schemes for FECFRAME", RFC 8681, DOI 10.17487/RFC8681, February 2020, <https://www.rfc-editor.org/info/rfc8681>.

Authors' Addresses

Ray Wang
Agora Lab
China

Email: wangrui@agora.io

Liang Si
Agora Lab
China

Email: siliang@agora.io

Bifeng He
Agora Lab
China

Email: hebifeng@agora.io