ID # 1569296653

AN ADAPTIVE TECHNIQUE FOR DATA COMPRESSION AND ROBUSTNESS AT THE TACTICAL EDGE[1]

Yan Grushevsky, Junghoon Lee, George F. Elmasry, Manoj Jain and Michael Vitt
DSCI-XPRT Solutions, Inc., Eatontown, NJ 07724

Gregory Hadynski
AFRL/RIGC, Rome, NY 13441

Bruce Metcalf
The MITRE Corporation, Bedford, MA 01730

ABSTRACT

An adaptive technique for reliable and fast transmission of streaming data at the tactical edge is presented. This technique combines data compression and data robustness to optimize for BW, reliability and speed of transmission in an adaptive matter. Speed of transmission and reducing BW needed for transmission is achieved by exploring lossless data compression of the payloads of the packet stream. Once all redundancy in the packet stream is removed, a packet-erasure-based FEC is applied that further optimize for BW and speed of transmission in addition to adding robustness to the packet stream that assures successful transmission within a speed of service given by tactical requirements. Lossless compression uses an adaptive Arithmetic Coding technique that provides a level of data compression close to theoretical limits (approaching the Shannon Bound) and does not require any prior knowledge of the statistical nature of the packet payload stream (performs compression on-the-fly). The FEC uses an RS code that can work over a general purpose process (does not require specific HW) and optimizes for the amount of redundancy needed. Robustness is achieved in the presence of high percentage of packet erasures caused by blockage, fading or malicious attack. The presented approach guarantees the delivery of a given message within a speed of service defined by tactical requirements and minimizes the BW needed for transmission. The paper presents the technique, shows supporting analysis and presents simulation results.

I. INTRODUCTION

The use of Performance Enhancement Proxy (PEP) for data transmission over Transmission Control Protocol (TCP) has been exploited in tactical environments [1] for some time. Increasing the speed of transmission of a TCP session in the presence of a satellite delay is an important improvement that PEP can provide. Other technologies [2] attempt to modify the transport protocol by compensating for lost packets with techniques that are different from retransmission. These include sending redundancy packets at the end of a TCP window to compensate for lost packets, and thus avoid retransmission that can delay session completion time. Other approaches attempt data compression at a modem [3]. In this work, we present a comprehensive and novel approach for PEP at the tactical edge that checks for lossless compression (to decrease the data stream size when possible), and applies a form of packet recovery through an adaptive packet erasure based Forward Error Correction (FEC) technique. This is in addition to using a flow control technique that is based on the precedence of the session (i.e., increase the flow for high precedence sessions and decrease it for low precedence sessions to overcome congestion), besides the flow control techniques used by typical PEP [1].

With this novel PEP approach, as the packet flow of a specific session is detected, the technique checks if applying lossless compression can achieve any compression gain. If so, the payload of the packet stream is extracted and compressed, and a new packet flow would then carry the compressed data. Arithmetic coding [4] is used for lossless compression since it can compress on-the-fly (i.e., it does not need to scan the entire data stream to achieve compression). The compressed data stream (which should now contain a lesser number of packets than the original data stream), is then passed through a FEC that uses a form of Reed-Solomon (RS) code with packets erasure in order to generate redundancy packets where a single redundancy packet can replace any lost packet. On top of the standard proxy flow control, a Multi-Level Precedence and Preemption (MLPP) based flow control is used to attempt to free bandwidth (BW) for higher precedence sessions.

This comprehensive approach results in an immense performance enhancement of TCP since, 1) The transmitted information is compressed to reduce the BW needed and increase the speed of transmission, 2) FEC prevents retransmission and hence further increases the speed of transmission, and 3) MLPP flow control helps the War-

[1] Research conducted under SBIR AF073-020, Phase II, "*Reservation-Based Quality of Service (QoS) in an Airborne Network*", sponsored by Air Force Research Labs, Rome, NY 13441.

fighter achieve his mission since the higher precedence sessions will now have more BW available than the lower precedence sessions.

We would like to note here that with immense increase in the computational capabilities in the hardware today, easy availability of huge and inexpensive memory, and development of packet erasure based RS coding that can be used over a general purpose processor, it should be quite possible to productize this comprehensive PEP approach. We plan on realizing such a product in the near future.

II. LOSSLESS COMPRESSION THROUGH ARITHMETIC CODING

To provide lossless data compression, an adaptive arithmetic compression was used. The arithmetic coding algorithm maps (compresses) the message, which consists of a set of symbols, into a very long floating point number between 0 and 1 (see [4] and [5]). The basic compression steps are described by the following expressions:

$$\begin{aligned}
&l_0 = 0;\ h_0 = 1,\ R_0 = 1 \text{ at } \mathbf{i} = 0;\\
&l_i = l_{i-1} + R_{i-1} * L_j, \text{ at } \mathbf{i} > 0;\\
&h_i = l_{i-1} + R_{i-1} * H_j\ ;\\
&R_i = h_i - l_i. \qquad\qquad (1)
\end{aligned}$$

Here L_j and H_j are lower and upper boundary of cumulative frequency of appearance of symbol **j** in the message;
l_i and h_i are lower and upper boundary of compressed value after compressing **i** symbols;
R_i is range of values after compressing **i** symbols;
$1 <= \mathbf{i} <= N$, where N is the overall number of symbols in the message, and
$1 <= \mathbf{j} <= N_d$, where N_d is the number of distinct symbols in the message.

For example, take the case of simple message that has only four symbols "**baca**". Here the cumulative frequencies are presented by the following values. For symbol "**a**", which appears twice in the message, $L_1 = 0$, $H_1 = 0.5$; for symbol "**b**", $L_2 = 0.5$, $H_2 = 0.75$; and for symbol "**c**", $L_3 = 0.75$ and $H_3 = 1.0$.

The iteration process described by expressions (1) continues for all N symbols in the message. As a result, boundary values l_i and h_i are converged. Any number in between l_N and h_N with the smallest number of digits will represent the compressed message. In the example above, such a number is 0.6.

The decompression starts by figuring out the half open interval $[L_i, H_i)$ in which the compressed value belongs. For value 0.6, such interval is [0.5, 0.75) at **i** = 2, which defines the symbol „**b**'. Then, decompression reverses the iteration process that was described by expressions (1).

The iteration process can be seriously affected by the underflow that could be caused by the limited number of bits in the computer's representation of the floating point fraction. The renormalization procedure implemented according to [5] and [6] allows for overcoming this potential problem, and can compress messages of any length.

The adaption part of the algorithm consists of building of an array of frequencies for all N = 256 ASCII symbols that can possibly occur in a message to be compressed. The array is started with the assumption of uniform distribution of symbols and is updated with every symbol that is to be compressed. The elements of such an array define the half-open intervals $[L_i, H_i)$ used in the expressions (1) above.

A similar process takes place at the decoder where the statistical model represented by the frequency array is used to restore the previously compressed information. The originally array is identical to the starting stage array at coding. Then, it is also updated with every new symbol, which is converted back and placed in the appropriate place in the frequency array.

Thus, there is no overhead data (like a table or dictionary) that is transmitted between the sender and the receiver. Note that process described here does not use any prior knowledge of the data to be compressed and decompressed, and works the same way for both short and long messages.

III. USE OF PACKET ERASURE BASED FEC

The data compression technique described above certainly allows substantial bandwidth savings but, at the same time, it results in an increase in the message's vulnerability [7] to packet erasures that could be caused by blockage, fading or malicious attacks. To increase the robustness of message delivery, the forward error correction (FEC) based on Reed-Solomon technique described in [8] and [9] has been implemented. The technique uses systematic Reed-Solomon codes for erasures built over Galois Field $\mathbf{2^8}$. This finite field has 256 elements that are easily matched to any 8 bit pattern. The Field, by definition, supports addition, subtraction, multiplication and division, defined by very specific rules [10]. It allows building a Vandermonde matrix [10], which after diagonalization yields the long **L** and short **G** generator matrixes as shown in Figure 1.

$$
\begin{array}{c|ccccc|c}
 & 1 & 0 & 0 & \dots & 0 & \\
 & 0 & 1 & 0 & \dots & 0 & \\
 & 0 & 0 & 1 & \dots & 0 & \\
\text{Matrix L} & \cdots & \cdots & \cdots & \cdots & \cdots & \\
 & 0 & 0 & 0 & \dots & 1 & \\
 & G_{11} & G_{11} & G_{11} & \dots & G_{1M} & \\
 & G_{21} & G_{22} & G_{23} & \dots & G_{2M} & \text{Matrix G} \\
 & \cdots & \cdots & \cdots & \cdots & \cdots & \\
 & G_{R1} & G_{R2} & G_{R3} & \dots & G_{RM} &
\end{array}
$$

Figure 1: Generator Matrixes

The width (number of columns) of these matrixes is equal to the message size M. The top $M \times M$portion of matrix **L** is identity matrix while the bottom portion defines matrix **G.** The number of rows in long generator matrix **L** is determined by sum $M + R$with the limitation $M + R < 256$. Here $R$is the number of redundancy packets to be constructed and added to the original message to counter the packet loss. The number of redundancy packets R is also defined as the number of rows in matrix **G**. By the way matrixes are constructed, it is guaranteed that the rank of both matrixes is M. This means that any submatrix of these matrixes of size $M \times M$ is invertible. The matrix **G** is used to generate the redundancy packets, while matrix **L** is used to recover the message from any Mpackets that reach the destination out of $M + R$packets that were sent.

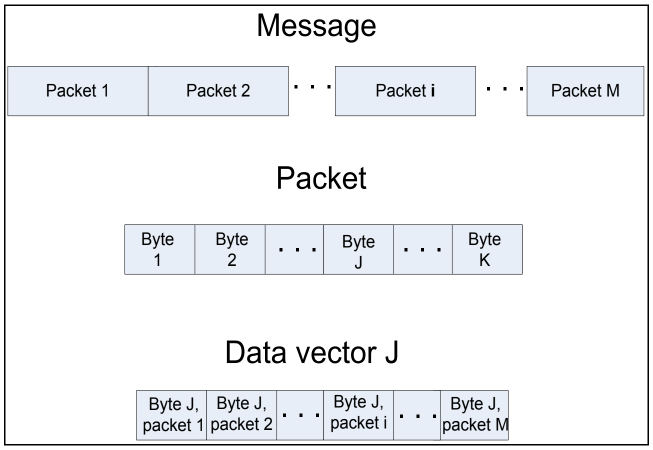

Figure 2: Message, Packet and Data Vector Structure

The redundancy packets are built in the following way. As shown in Figure 2, the message to be sent consists of M packets, and every packet consists of Kbytes.

The algorithm starts with constructing $\boldsymbol{M}$ auxiliary data vectors, where data vector $\overline{\boldsymbol{D_j}}$ ($\boldsymbol{1} \leq \boldsymbol{j} \leq \boldsymbol{K}$) is built out of bytes $\boldsymbol{j}$of every of $\boldsymbol{M}$data packets, where $\boldsymbol{P}$is the number of packets in a message. Then, vector $\overline{\boldsymbol{E}_j}$ of RS encoded bytes is computed as the product:

$$\bar{E}_j = G \times \bar{D}_j \qquad (2)$$

Here vector $\bar{E}_j = (e_{1j}, e_{2j}, \ldots, e_{Rj})$and $e_{1j}$is the byte j of the 1st encoded (redundant) packet, e_{2j} is byte $j$of the 2nd redundant packet, etc. Applying expression (2) for all message bytes in the range: $1 \leq j \leq K$ allows encoding of complete R redundant packets.

After all Rencoded packets are computed, these together with Mdata packets are sent to the destination. This can be expressed in the matrix form:

$$S = L \times P \qquad (3)$$

Here $P$is a matrix of size $M \times K$ built out of the original Mdata message packets, so $\{P_{ij}\}$ is byte $j$of packet i; $S$is also a matrix of size $(M + R) \times K$, in which the elements of the first Mrows match to the M data packet bytes, because the top of matrix L is an identity matrix. The elements of the bottom Rrows are computed according to Equation (2).

This guarantees that if the destination gets any Mpackets out of the $M + R$packets that were sent, the remaining M rows from Equation (3) can be considered to be a system of simultaneous linear equations. The rank of such a system is M, which means that the system (3) has only one set of solutions, and this set allows restoration of all lost packets and, as a result, complete and correct decoding of the sent information.

References [8] and [9] show in detail how the number of redundancy packets R depends on the message size, network conditions, throughput, and also the requirements for speed of service and robustness of message delivery. It was shown [9], in particular, that longer messages require relatively lesser number of redundant packets than shorter messages under the same packet loss. However, the processing time for decoding grows exponentially with the message size, which could negatively affect the overall PEP performance. That is why the message size (TCP segment to be encoded at a time using RS coding) can be limited. Also, the packet size K needs to be chosen as a compromise between preventing any message fragmentation and still maximizing the throughput. We propose K= 1500 minus the combined size for all headers used for packet transmission. Here, 1500 is maximum transmission unit (MTU) in bytes.

Note that in this PEP implementation, the decoder uses the LU[1] decomposition method [11] to solve the system of linear Equation (3), over Galois field. This method is much faster than the Gaussian elimination method employed in [8].

To summarize, the RS coding presented here, which is usable over a general purpose processor, has been utilized to generate redundancy packets to be added at the end of a TCP segments of packets. A single redundancy packet can replace any lost packet in the TCP segment and hence reduces the probability of retransmission, leading to faster completion of a TCP session and the ability to work over a network with high packet loss (e.g., tactical networks). Note that the number of redundancy packets is decided dynamically based on the current state of the network (i.e., real-time measurement of packet loss ratio). Please see Reference [9] for more details.

IV. MLPP FLOW CONTROL

This section presents Multi-Level Precedence and Preemption (MLPP) based flow control techniques to be used for TCP proxy at the network edge. In particular, the techniques are for application to the tactical networks, which have specific Speed of Service (SoS) requirements for each message precedence. In compliance to the MLPP principle, each TCP connection is controlled as follows:

Forced TCP Connection Abort

As illustrated in Table 1, the purpose of TCP connection abort procedure is to establish the TCP connections in accordance with the SoS requirements specified for the precedence levels of the transmissions. In other words, if a connection violates the SoS requirement, the TCP connection is aborted following the procedure given in Table 1.

Table 1 is given for the case that the MLPP policy states to allow aborting connections with lower three precedent levels. T1, T2, T3, M1, M2, and M3 are parameters to be decided by the policy. Similarly, the abort procedures can be established for application to any desired precedence levels.

[1] The LU decomposition is a matrix decomposition, which writes a matrix as the product of a Lower triangular matrix and an Upper triangular matrix.

Table 1: TCP Connection Abort Procedure

Traffic Conditions	TCP Connection Status
No violating incidents.	Establish TCP connection meeting the requirements for SoS.
Occurrence of the 1st violating incident during the current communication session.	Abort TCP connections having a level of precedence (LP) = LP1 for a pre-determined time interval T1 or a pre-determined number M1 of transmissions.
Violation incident occurs during the time interval T1 or M1 transmissions.	Abort TCP connections having a level of precedence (LP) $\leq$ LP2 for a pre-determined time interval T2 or a predetermined number M2 of transmissions.
Violation incident occurs during the time interval T2 or M2 transmissions.	Abort TCP connections having a level of precedence (LP) $\leq$ LP3 for a pre-determined time interval T3 or a pre-determined number M3 of transmissions.
Violation incident occurs during the time interval T3 or M3 transmissions.	Abort TCP connections having a level of precedence (LP) $\leq$ LP4 for a pre-determined time interval T4 or a pre-determined number M4 of transmissions.

Retransmission Time-Out Adjustment

In TCP flow control, the value of Retransmission Time-Out (RTO) timer is set in accordance with the packet transmission delay of the network, or the Round Trip Time (RTT). As the RTT becomes larger, the RTO value is set to a larger value to properly maintain the TCP connection. However, a larger RTO value can also introduce a longer delay in the TCP flow, thus resulting in a violation of the SoS required for the MLPP-based tactical network. The propose method adjusts the RTO timer value in a way that guarantees the SoS requirement. It applies to the TCP flow control between a pair of proxies as long as the connection is maintained. It applies to TCP flow control using adaptive RS code as described in the previous section.

If $\mathbf{K_T}$ number of pushes or ($\mathbf{K_T}$-1) retransmissions are required to meet the required Message Completion Rate (MCR), the RTO value should be selected following the criterion:

$$D_f + (\mathbf{K_T}\text{-}\mathbf{1})\cdot \boldsymbol{RTO} < L_{qm}$$

where D_f denotes packet transfer delay in the forward direction and L_{qm} denotes the latency required for a message of given priority, or

$$RTO < (L_{qm} - D_f) / (K_T\text{-}1)$$

and K_T is selected in accordance with the following criterion:

$$\sum_{k=1}^{K_T} p^{1-k}(1-p)^k < R$$

where p denotes packet loss ratio and R denotes the reliability required for a message of given priority. In the protocol implementation, p will be estimated by the measurement between the proxies. For a military network, L and R will be given for each message priority level.

For a TCP connection, the RTO value will be calculated by the measured or estimated values of delay and jitter. Whenever the RTO value needs to be updated, the proxy calculates:

$$RTO_{max} = (L - D_e - \alpha \cdot J_e) / (K_T\text{-}1)$$

where D_e and J_e denote estimated packet transfer delay and jitter in a forward direction, respectively, and α denotes a parameter to account for the inaccuracy and/or instability of the measurement. If an RTO value calculated by the TCP is greater than RTO_{max}, then the RTO value is replaced by RTO_{max}. In other words, TCP retransmits in a shorter interval than one based on an ordinary TCP flow control method, in order to meet the SoS requirement specified for the connection.

Congestion Window Size Control

When there exists a large delay in a network, data transmission speed of ordinary TCP flow is limited in association with the delay and the size of the congestion window (**cwnd**). The proposed window size control for the TCP flow between the TCP proxies at the edge of the encrypted network attempts to guarantees the SoS requirement in reference to the delay estimated at the TCP proxy.

For a TCP connection with a large transfer delay, the maximum data rate of a TCP connection is limited to ***(cwnd / RTT)***. Therefore, the proposed method attempts to adjust **cwnd** to follow the criterion:

$$(cwnd / RTT) > MSS / L_q \quad \text{or} \quad cwnd > RTT \cdot MSS / L_q$$

where MSS and L_q denote maximum segment size of the TCP connection (typically 1460 bytes) and the latency requirement, respectively. In this way, it prevents the SoS requirement from being violated by the size of the congestion window.

Connection Admission Control (CAC) Mechanism at TCP Connection Establishment

Through continuous monitoring the network path conditions of TCP connections for each precedence level, the proposed method performs CAC at the instant of TCP connection establishment as follows:

A TCP connection request is rejected unless the following conditions are met:

$$RTO_{max} > \beta \cdot RTT_e \quad \text{and} \quad rwnd > cwnd\_required$$

where RTO_{max} denotes the retransmission time out (RTT) value given by the flow control method applied, RTT_e denotes the RTT value estimated for the message priority at the time of TCP connection, β denotes a parameter to consider efficiency of the flow control, and $rwnd$ denotes the advertised maximum window size of the receiving proxy.

The typical value of β could be 2. A lower value of β means that the retransmission could occur even for the case that the corresponding segment is acknowledged but the ACK packet arrived after the RTO expired. The intent of the second condition is to avoid a case that the connection cannot provide the required SoS due to a small **cwnd** limited by the **rwnd**.

V. COMPREHENSIVE TACTICAL EDGE PROXY PERFORMANCE

Utilizing an actual testbed, we ran a number of tests to verify the compression technique presented above. Figure 3 shows the results of transmitting an uncompressed document file versus a compressed document file. The bandwidth used has been averaged over 60 seconds. The file was sent using an FTP client, which sends the data over the TCP protocol. The compressed file finished in about half the time, and required about half the bandwidth when compared to the uncompressed file. The initial bandwidth used is the same because the FTP client will send more packets for the compressed file. Therefore, the bandwidth used will climb at the same rate, but then the compressed file will finish more quickly and use less bandwidth than the uncompressed file.

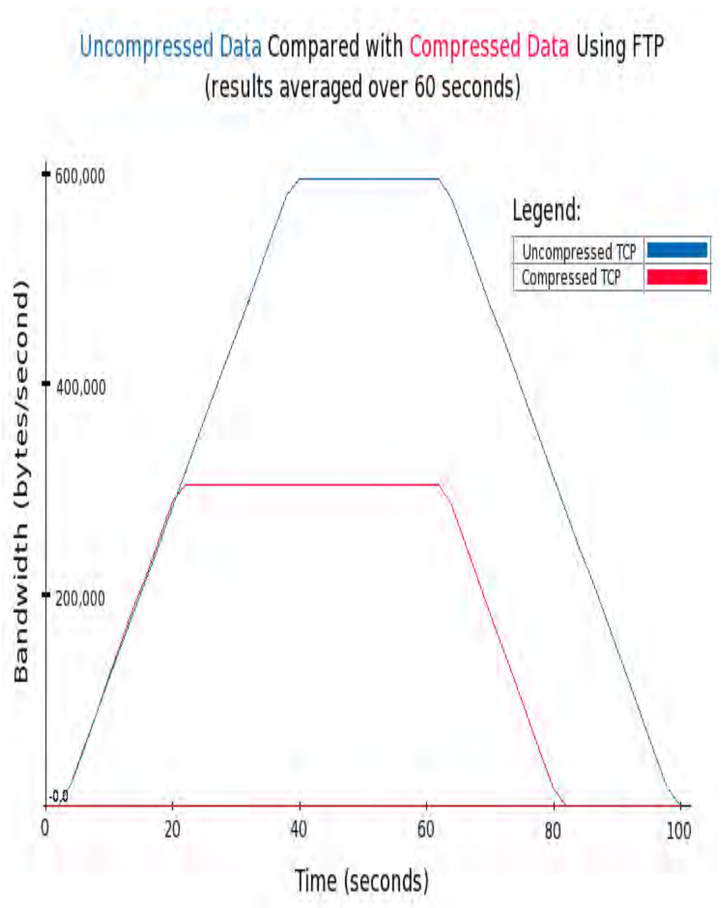

Figure 3: Testbed Results for Data Compression

The performance improvement based on the use of RS code to compensate for lost packets will be presented as the final version of this paper. We would like to note here that PEP should adjust itself based on the available BW at the core network. If BW is abundant and the core network is reliable, there is no need for compression or adding redundancy packets (actually the computations performed for PEP can then be the bottleneck that causes transmission to slow down).

If the core network path causes high packet loss, even if BW is abundant, the use of RS code can substantially improve the performance since it minimizes retransmission. If BW is scarce and the path causes high packet loss, a definite gain comes from both compression and RS code implementation. Another factor worth mentioning here is the processing capabilities at the tactical edge. If a high end general purpose processor is used, PEP shows improvement in performance at higher BW availability (computational bottlenecks happen at higher BW). Work is in progress to quantize these compound relationships and show the results in one or two curves.

VI. SUMMARY

The ever increasing processing capability in the hardware and easy availability of huge and inexpensive memory make it possible today to develop the more complex PEP presented in this paper. This PEP: (1) Exploits any lossless compression capability in the TCP session (packet payload), and hence reduces the size of the message to be transmitted; (2) Adds redundancy packets generated using an RS code for packet erasure; and (3) Implements an elaborate flow control technique, in addition to implementing the standard PEP techniques. The resulting PEP can work over low bandwidth, high packet loss networks and will make it possible for TCP to deliver the Warfighters' messages with a high level of assurance and within the defined speed of service requirements.

VII. REFERENCES

[1] N. Schult, et al, "*A Performance Evaluation of Transport Mechanisms in Hybrid Networks*", The MITRE Corporation, MILCOM 2006, 23-25 October 2006.

[2] Byers L.W., Luby M., Mitzenmacher M., "*A Digital Fountain Approach to Asynchronous Reliable Multicast*", IEEE J. Select. Area Commun., 2002, 20, (8) pp. 1528-1540.

[3] A.Lettieri, K. Holtz, E.Holtz, "*Data Compression in the V.42bis Modems*", WESCON/94, "Idea/Microelectronics", Conference record, 27-29 Sep 1994, 398-403.

[4] Ian H. Witten, Radford M. Neal, and John G. Cleary, *Arithmetic coding for data compression*, "Communications of the ACM" 30:520-541, June 1987.

[5] Mark Nelson, Jean-Loup Gailly, The Data Compression Book, M&T Books, A division of MIS:Press, Inc., 2nd edition, 1995.

[6] Alistair Moffat, Radford Neal, Ian H. Witten, *Arithmetic code revisited*, Data Compression Conference, 1995; DCC „95 Proceedings, P202-211.

[7] D. W. Sharp, D. S. Roth, "Adaptive Data Compression for a VLF Communication System", Military Communication Conference, 1990, MILCOM'90, vol.3, 1030-1035.

[8] Y.L.Grushevsky, G. F. Elmasry, S.R. Argentieri, R. Lussier, *Adaptive RS code for message delivery over encrypted military wireless network*, Proc., IEE Mili-

tary Communications Conf. 2006, Washington DC, October 2006 US-M-Q.

[9] Y.L.Grushevsky, G. F. Elmasry, *Adaptive RS Codes for Message Delivery over an Encrypted Mobile Network*, IET Commun., 2009, Vol.3, Iss.6, pp1041-1049.

[10] Plank, J.S., *A Tutorial on Reed-Solomon Coding for Fault-Tolerance in RAID-like Systems*, Software-Practice & Experience 1997, 27(9): pp. 995-1012.

[11] Press, W. H.; Flannery, B. P.; Teukolsky, S. A.; and Vetterling, W. T. "*LU Decomposition and Its Application.*" §2.3 in Numerical Recipes in FORTRAN: The Art of Scientific Computing, 2nd ed. Cambridge, England, Cambridge University Press, pp. 34-42, 1992.