Hindawi
Security and Communication Networks
Volume 2024, Article ID 7538203, 10 pages
https://doi.org/10.1155/2024/7538203

Research Article

Securing the Transmission While Enhancing the Reliability of Communication Using Network Coding in Block-Wise Transfer of CoAP

Mohammed D. Halloush

Department of Computer Engineering, Hijjawi Faculty for Engineering Technology, Yarmouk University, Irbid, Jordan

Correspondence should be addressed to Mohammed D. Halloush; mdhall@yu.edu.jo

Received 8 October 2022; Revised 26 February 2024; Accepted 12 March 2024; Published 28 March 2024

Academic Editor: Shadab Alam

The practical employment of network coding (NC) has shown major improvements when it comes to the transmission reliability of sender data and bandwidth utilization. Moreover, network coding has been employed recently to secure the transmission of data and prevent unauthorized recovery of sender packets. In this paper, we employ network coding (NC) in a practical way in networks with constrained resources with the goal of improving the reliability and security of data transfer. More specifically, we apply NC on the recent options of block-wise transfer (BWT) of the constrained application protocol (CoAP). The goal is to enhance the reliability of CoAP when used to transfer larger data blocks using BWT. Also, we employ an innovative homomorphic encryption approach to secure the BWT of CoAP.

1. Introduction

Constrained application protocol (CoAP) was designed for IoT applications in which IoT devices communicate messages of small size [1]. In certain circumstances, there is a need to communicate large resources. Block-wise transfer (BWT) is an extension for CoAP that has been developed to deal with such circumstances [2]. With BWT, the resource is partitioned into smaller blocks. The blocks are sent to the receiver, and the original resource is constructed once all the blocks are delivered. In lossy environments where it is possible for some blocks to be lost, the sender retransmits the lost blocks which are going to cause an increased transmission overhead and delays.

Network coding [3] has been employed to enhance the reliability of data transmission in lossy environments. The goal is to decrease the overall number of transmissions needed to deliver sender data [4, 5]. NC is based on the idea of sending linear combinations of the packets rather than sending the packets in their original form. Each encoded packet is generated using an encoding vector produced from a finite field. The goal is to provide the receiver with the sufficient number of encoded packets rather than providing the receiver with specific packets. The encoded packets are linearly independent and are decoded at the receiver to recover the sender packets. In case of loss, the sender needs to send extra encoded packets to ensure delivering the required number of encoded packets to the receiver to be able to decode.

Applying NC in BWT of CoAP can improve the reliability of data transfer in the case of loss. BWT supports several options for block transfer [6, 7]. Block1 and Block2 options support the reliable transfer of blocks in a synchronous way. This means that the next block is sent once the previous block is received and acknowledged. Block1 option is used when sending the block from the client to the server as part of a request. On the other hand, Block2 option is used when sending a block from the server to the client as part of a response.

Due to the synchronous transfer nature of Block1 and Block2 options, they do not have the best performance in the case of loss. Q-Block1 and Q-Block2 are new BWT options

that have been developed recently [8]. Q-Block1 and Q-Block2 options are similar to Block1 and Block2 options with the difference of supporting the transfer of more blocks without the need to wait for acknowledgments. This means with Q-Block1 and Q-Block2 options, the transfer of blocks is done faster especially in the case of block or acknowledgment loss.

Applying NC with BWT is done by dividing the large resource into smaller blocks and generating encoded blocks to be forwarded. Encoding vectors are randomly generated at the sender and used to generate these encoded blocks. The encoding vector used to generate the encoded block is transferred with the encoded block. The receiver collects these encoded blocks and performs decoding once the sufficient number of encoded blocks with linearly independent encoding vectors is received.

Another advantage of applying NC to BWT is the ability of securing the blocks sent without the need to encrypt the whole block. Minimizing the size of encrypted data is important in applications with constrained communicating devices. Once encoded blocks are generated at the sender, the sender can encrypt the encoding vectors used to generate these blocks. The encoded blocks cannot be decoded without having the decrypted encoding vectors. The receiver of the encoded blocks can decrypt the encoding vectors and then decode to recover sender's blocks.

Homomorphic encryption (HE) is one of the options that can be employed for encrypting the encoding vectors of encoded blocks [9–11]. Homomorphic encryption allows performing operations on encrypted data without affecting the ability of the receiver to decrypt the data. This means sender as well as intermediate nodes can perform NC operations on encrypted data, without affecting the ability of the receiver to decrypt the received encoded data. Encrypting the encoding vectors rather than the whole block limits the computational overhead of encryption. Decreasing the encoding vector size means less data to be encrypted. As we will see later, the size of the encoding vector depends on the number of encoded blocks. The advantage of using homomorphic encryption is that NC operations can be performed on the encrypted data without affecting the ability of the receiver to decode the sender data.

In this work, we apply NC on the recently developed transfer options for BWT (Q-Block1 and Q-Block2) [8] to achieve two goals. The first is to improve the ability of the receiver to recover sender's blocks in the case of loss. The second is to secure the encoded blocks by encrypting the encoding vectors used to generate the encoded blocks. To the best of our knowledge, our work is the first attempt to apply NC on the new transfer option of BWT (Q-Block1 and Q-Block2). Our focus in this work is to show the benefits that can be gained by applying NC on the new options of BWT. Also, we extend the new options of BWT by proposing a set of parameters that are needed to apply NC in a practical way. These parameters are needed to enable the sender and receiver to communicate NC information necessary to perform encoding and decoding of resource blocks.

The rest of the paper is organized as follows: Section 2 presents related work. Section 3 provides an overview of NC and CoAP along with BWT. Section 4 discusses the application of NC in BWT of CoAP with transfer scenarios. Section 5 discusses the use of homomorphic encryption to secure the blocks of BWT. Section 6 assesses the computational overhead incurred by NC. Section 7 assesses the improvements achieved by NC on the reliability of data transfer. Finally, the paper is concluded in Section 8.

2. Related Work

NC has been discussed thoroughly in the literature due to its benefits in enhancing the reliability of packet delivery in different network scenarios. Also, NC has been employed to secure the transmission of data. NC can be employed in scenarios where security of data transmission is needed while the computational capabilities of senders are limited. This can be seen in IoT applications where the devices sending data have limited computational power and memory.

NC has been employed in different types of networks used for different applications in wireless sensor networks (WSNs), mobile ad-hoc networks, vehicular area networks, P2P networks, multimedia communication, and IoT [12–16]. NC has been investigated as an alternative forwarding approach. Unlike the traditional store and forward approach, with NC, the sender sends linear encodings which are generated using encoding vectors randomly generated from a finite field. For example, to generate n encoded packets, n linearly independent encoding vectors are needed. The encoding vector used to generate a specific encoded packet is sent as part of the packet since it is needed at the receiver to decode and recover the original sender packets.

Several approaches were proposed to secure the data while using NC [17–19]. With NC, since encoded packets are sent, the confidentiality of the encoded packets is maintained but not for too long. A node that receives the encoded packets will not be able to decode as long as the number of encoded packets received is less than the minimum number of encoded packets needed to decode. As the node receives more encoded packets, it will be able to decode once it receives the sufficient number of encoded packets. To ensure the confidentiality of the encoded packets, the sender can encrypt the encoded packets. The receiver decrypts the received encoded packets before decoding. This is suitable for applications where NC is applied end to end where there are no intermediate nodes between the sender and receiver or the intermediate nodes are not participating in the encoding of received packets.

To limit the computational overhead of encryption, the sender can encrypt the encoding vector only rather than the whole encoded packet. Encrypting the encoding vectors prevents the unauthorized access to the data. It is only the receiver who is capable of decrypting the encoding vectors and hence decoding sender's packets.

The limitation with this approach is that NC can be applied only at the sender (not on intermediate nodes) because encryption done on the encoding vectors is end to end. This means that the sender who generates the encoded packets will encrypt the encoding vectors and send the

encoded packets to the receiver who is the only node able to decrypt and then decode the received packets. This will affect the performance of generating independent linear encoded packets.

To overcome this limitation, the authors in [19] proposed appending to the encoded packet a locked and an unlocked vector. The locked encoding vector is used to generate the encoded packet at the sender and then encrypted (locked), and an unlocked vector is appended to the packet to track NC encoding performed at intermediate nodes. In this case, intermediate nodes can encode the received packets and generate new linearly independent encoded packets using locally generated encoding vectors that also update the unlocked encoding vectors.

Once the sufficient number of encoded packets is received at the receiver, Gaussian elimination is performed on the unlocked encoding vectors, and the sender packets are decoded. Once the sender packets are decoded, the receiver will be able to decrypt the locked encoding vectors and decode to recover the original sender packets. The limitation of this approach is that it requires appending two encoding vectors to each sent packet which incurs a transmission overhead. Another limitation is the computational overhead caused by the need to reverse all the updates performed by intermediate nodes once the sufficient number of encoded packets is received at the receiver to recover the sender packets with the locked encoding vectors. This is necessary for the receiver to be able to unlock (decrypt) the locked encoding vectors and then decode.

Sending the sender packets with the unlocked encoding vectors has a major impact on the performance of NC since each of these packets is needed at the receiver. This means if any of the sender packets with an unlocked encoding vector is lost before being encoded by an intermediate node, the receiver will not be able to receive the sufficient number of packets to build a matrix of unlocked vectors that have full rank.

To overcome these limitations, homomorphic encryption can be employed. With homomorphic encryption, there is no need for the unlocked encoding vector. It is sufficient to lock the encoding vectors used to generate the encoded packets at the sender using homomorphic encryption. Intermediate nodes are able to generate linear encoded packets using locally generated encoding vectors to update the locked encoding vectors. The advantage of homomorphic encryption is that the operations performed on the encrypted encoding vectors will not affect the ability of the receiver to decrypt these vectors. An encoded packet with a locked encoding vector is decrypted once it is received by the receiver, and once the receiver receives and decrypts the sufficient number of encoded packets, it can decode these packets and recover sender's packets.

An application that can benefit from NC is the blockwise transfer (BWT) of CoAP. CoAP is an application layer protocol that is used in IoT. BWT is part of CoAP that is used to transfer larger resources between IoT devices. The larger sender resource is divided into blocks to limit the size of the data exchanged between constrained IoT devices. BWT offers two synchronous options for data transfer (Block1 and Block2) where each block sent is explicitly acknowledged before sending the next block. In [6], the authors have applied NC on Block1/Block2 options of BWT to decrease the number of additional transmissions in the case of lost acknowledgment. This is done by encoding the previously sent unacknowledged packet with the next outgoing block after timeout.

In [20], NC was applied on BWT to decrease losses by sending linear combinations of the resource blocks. The resource is divided into generations where each generation has a specific number of blocks. The behavior of BWT was slightly modified to allow sending a sequence of nonconfirmable responses after sending a piggybacked acknowledgment for a confirmable request. The sender sends encoded blocks of a generation until the receiver requests blocks of the next generation.

Recently, two asynchronous options were provided for BWT (Q-Block1 and Q-Block2). With these options, more blocks can be sent without waiting for acknowledgments which make it more suitable for the application of NC. In our work, we take the advantage of the characteristics of Q-Block1 and Q-Block2 options of BWT to apply NC in a practical way. There are two goals for applying NC. The first is to enhance the reliability of data transfer by decreasing the number of transmissions needed to deliver the resource. The other goal is to facilitate securing the sent blocks by encrypting the encoding vector used to generate an encoded block rather than encoding the whole block which limits encryption computational overhead.

3. Network Coding and CoAP

3.1. Network Coding. The idea of network coding (NC) is to send encoded packets that are linear combinations of the plain sender packets. The sender generates encoded packets using encoding vectors generated from a finite field GF (2^F) [4]. The encoding vector used to generate the encoded packet y_i out of n sender packets $(p_1, \ldots, p_n)$ is

$$\alpha_i = [\alpha_{i1}, \ldots, \alpha_{in}]. \tag{1}$$

The encoding operation at the sender to generate an encoded packet (y_i) is

$$y_i = \sum_{j=1}^{n} \alpha_{ij} \cdot p_j. \tag{2}$$

At least n encoded packets must be generated by the sender. The sender can generate any number of encoded packets. The more encoded packets sent the easier for the receiver to receive n encoded packets necessary for decoding. The encoring matrix used to generate n encoded packets is

$$\begin{bmatrix} \alpha_{11} & \cdots & \alpha_{1n} \\ \vdots & \ddots & \vdots \\ \alpha_{n1} & \cdots & \alpha_{nn} \end{bmatrix}. \tag{3}$$

The n encoded packets at the sender are

$$\begin{bmatrix} y_1 \\ \vdots \\ y_n \end{bmatrix} = \begin{bmatrix} \alpha_{11} & \cdots & \alpha_{1n} \\ \vdots & \ddots & \vdots \\ \alpha_{n1} & \cdots & \alpha_{nn} \end{bmatrix} \begin{bmatrix} p_1 \\ \vdots \\ p_n \end{bmatrix}. \quad (4)$$

A forwarding node encodes received encoded packets to generate at least one encoded packet for each received encoded packet. The forwarding node generates an encoding vector to be used to generate an encoded packet. For h received packets at a forwarding node $(y_1, \ldots, y_h)$, the encoding vector used to generate the encoded packet is

$$\beta_i = [\beta_{i1}, \ldots, \beta_{ih}]. \quad (5)$$

And the encoded packet generated at the forwarding node is

$$y_i' = \sum_{j=1}^{h} \beta_{ij} \cdot y_j. \quad (6)$$

The encoded packets generated at a forwarding node have an updated encoding vector. The updated encoding vector for the encoded packet $y_i'$is

$$\alpha_i' = [\alpha_{i1}', \ldots, \alpha_{in}']. \quad (7)$$

The receiver decodes the received encoded packets once it receives n linearly independent encoded packets. The matrix of the encoding vectors received with the n encoded packets is

$$A = \begin{bmatrix} \alpha_{11}' & \cdots & \alpha_{1n}' \\ \vdots & \ddots & \vdots \\ \alpha_{n1}' & \cdots & \alpha_{nn}' \end{bmatrix}. \quad (8)$$

Hence, decoding performed at the receiver is done by finding the inverse of the encoding matrix A and then performing the following:

$$\begin{bmatrix} p_1 \\ \vdots \\ p_n \end{bmatrix} = A^{-1} \begin{bmatrix} y_1' \\ \vdots \\ y_n' \end{bmatrix}. \quad (9)$$

The benefits of NC come from the fact that the receiver needs any n independent encoded packets to be able to decode and recover the original sender's packets, unlike conventional store and forward networks where each specific packet is needed at the receiver and in case of a packet loss that packet must be retransmitted by the sender.

3.2. Constrained Application Protocol (CoAP). CoAP is an application layer protocol that is used in IoT where devices have constrained resources [1]. Devices with constrained resources have limited processing capabilities and small memory [21]. CoAP was developed to enable IoT devices to communicate with low overhead. Hence, it was developed as a lightweight version of HTTP. Unlike HTTP, CoAP supports the communication of IoT devices through the Internet by adopting the unreliable user datagram protocol (UDP) with low overhead at the transport layer. CoAP is based on REST where a resource is identified using a URI in a request response interaction between the client and the server. The URI of a resource can be used to access the resource using the methods (POST, PUT, GET, and DELETE).

CoAP consists of two layers. The upper layer is the request/response layer, and the lower layer is the messaging layer. The request response layer is responsible for the request/response messages and communication methods. The messaging layer supports four types of messages and implements reliability on top of UDP by offering two types of messages: confirmable messages (CON) and nonconfirmable messages (NON). A confirmable message requires an acknowledgment (ACK) so that the next message is sent after receiving and acknowledgment of the previous one. In case the receiver is not able to respond to a confirmable message, a reset message (RESET) is sent that will reset the communication. A nonconfirmable (NON) message does not require any response from the receiver.

3.3. Block-Wise Transfer (BWT). CoAP was developed on top of UDP to support the transfer of data by providing simple means of reliable transfer flow and congestion control through the messaging layer. If the size of the payload is large to be sent in a single packet, BWT divides the resource into smaller blocks to avoid IP fragmentation. The advantage of BWT is to avoid sending data larger than what can be accommodated in a single link layer packet of a constrained network. By dividing the data into smaller blocks, the overhead caused by the fragmentation done at the adaptation layer or the IP layer is avoided. Also, this will make tracing the exchange of the blocks simpler for debugging purposes [2].

BWT supports several options: Block1, Block2 [2], Q-Block1, and Q-Block2 [8]. Block1 is used for the transfer of the resource from the client to the server that is associated with a request. Block2 on the other hand is used for the transfer of the resource from the server to the client that is associated with a response. Block1/Block2 options support the reliable transfer of the blocks by transferring blocks in a synchronous way. A block is sent once the confirmation of the successful delivery of the previous block is received. Transmission scenarios for Block1/Block2 can be found in [2].

Transferring blocks synchronously with Block1 and Block2 limit the transfer rate to the round trip time. This is acceptable in scenarios where loss rates are very low. Quick-Block1 (Q-Block1) and Quick-Block2 (Q-Block2) are new BWT options that have been proposed recently to support the transfer of blocks faster than with Block1 and Block2 especially in the case of loss. This is achieved by having less exchange of messages and faster recovery of lost blocks [8].

Similar to Block1, Q-Block1 is used to transfer a resource from the client to the server as part of a request. Also, similar to Block2, Q-Block2 is used to transfer a resource from the server to the client as part of a response. Q-Block1 and Q-Block2 allow the transfer of more blocks without the need to explicitly acknowledge each block. The sender can send several blocks without waiting for acknowledgments. The receiver can acknowledge several received blocks with

a single acknowledgment. Faster recovery supported by Q-Block1 and Q-Block2 is achieved by allowing the request of several lost blocks through a single message rather than sending a message for each lost block.

Allowing the transfer of several blocks without waiting for acknowledgments makes Q-Block1 and Q-Block2 more suitable for applying NC.

4. Quick Block-Wise Transfer with NC

To enhance the reliability of BWT, the sender sends encoded blocks of the resource. Blocks that belong to the same resource are encoded together. The number of encoded blocks that is sent by the sender is at least the number of blocks in the resource. Extra encoded blocks can be sent to overcome losses. The advantage of NC is that a lost encoded block can be replaced by any other encoded block. For the receiver to be able to decode and recover the sender blocks, it needs a number of encoded blocks which is equal to the number of blocks in the resource. Without NC, the receiver needs to acknowledge received blocks so that the sender resends the specific blocks that were lost. With NC, the receiver needs to acknowledge the number of encoded blocks received so that the sender sends extra encoded blocks to deliver to the receiver the number of encoded blocks it needs to decode.

The number of blocks in a resource depends on the block size. The smaller the block, the larger number of blocks in a resource. Larger number of blocks means more encoded blocks to send. Also, the larger number of blocks in a resource, the bigger encoding vector used to generate the encoded block. To limit the number of blocks in a resource, blocks can be divided into generations where the generation size is k. The generation size is a parameter that can be negotiated between the client and the server. The larger the generation size, the smaller number of generations of a resource, but the larger encoding vector is associated with an encoded block. Later in the paper, we assess the effect of the generation size on the performance.

Our focus in this paper is to apply NC on BWT options (Q-Block1 and Q-Block2). At the sender, the resource is divided into generations where the size of the generation is k blocks. An encoding vector of k coefficients is randomly generated from a finite field. k encoding vectors are needed to generate k encoded blocks. In case of a block loss, the sender generates a new encoded block using a new encoding vector. We present scenarios for Q-Block1 and Q-Block2 resource transfer with NC. We show the behavior in the case of no block loss also when there are losses. In the presented scenarios, the block consists of two generations and the generation size is three ($k = 3$). The behavior is the same when having different number of blocks and different number of generations.

In the scenarios shown in Figures 1–4, the parameters involved when applying NC on Q-Block1 and Q-Block2 options are GNum/BNum/GSize/BSize/M where

(1) GNum: generation number where the first generation number in a resource is zero

(2) BNum: encoded block number where the first encoded block number in a generation is zero

(3) GSize: the number of blocks in the generation

(4) BSize: the size of the encoded block

(5) M: more bit (0 for the last block in the resource and 1 otherwise).

Figure 1 is for the scenario of Q-Block1 where the block is sent from the client to the server as part of the request. In this scenario, we assume no losses. The client sends three encoded blocks for the first generation. Server responds with 2.31 continue where the client proceeds by forwarding the three encoded blocks of the second generation. The resource now is successfully delivered, and the server responds with 2.04 changed indicating the successful delivery of the resource.

Figure 2 is for the scenario of Q-Block1 where there are losses. The client sends the three encoded blocks of the first generation where the last encoded block sent is lost. The client waits for the server response which will not be received, since the server is still waiting for the third encoded block of the first generation. After the expiration of the client time out interval, the client proceeds by sending encoded blocks of the second generation. Once the server receives an encoded block of the second generation, it detects the loss of the first generation encoded block and notifies the client with 4.08, a single missing block ($C = 1$) of generation $G1$. The client sends an encoded block of the first generation to overcome the loss and resumes the transmission of the second generation encoded blocks.

The two encoded blocks of the second generation are lost, and the sender does not have more blocks to send. Upon the expiration of the server timeout interval, it sends a 4.08 missing response with $C = 2$ to indicate the loss of two encoded blocks. The client sends two encoded blocks of the second generation to overcome the losses. The two blocks are delivered, and the server responds with 2.04 changed indicating the successful delivery of the resource.

Q-Block2 has the exact same behavior as Q-Block1 with the difference that the resource is sent from the server to the client as part of a response as shown in Figures 3 and 4. Figure 3 shows the behavior of Q-Block2 with NC when there are no losses, while Figure 4 shows the behavior of Q-Block2 with NC when there are losses.

5. Homomorphic Encryption

With the increasing popularity of IoT, there are increasing concerns related to privacy and security. With the increasing number of IoT devices connected, there are increasing security risks. With that, there is a need for higher security measures for enhanced protection. At the same time, there is a growth on the number of real-time IoT applications in different sectors like healthcare and the industry which require fast response and minimal security computational requirements and delays [21–26].

Different approaches have been proposed to overcome the security concerns in IoT. Encryption can be employed

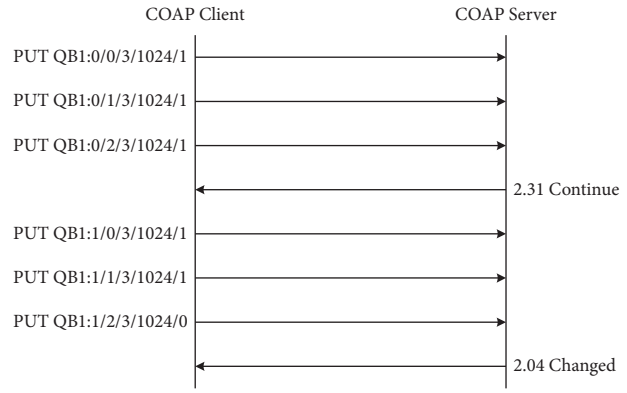

Figure 1: Q-Block1 with NC no loss scenario, a resource of two generations ($k = 3$).

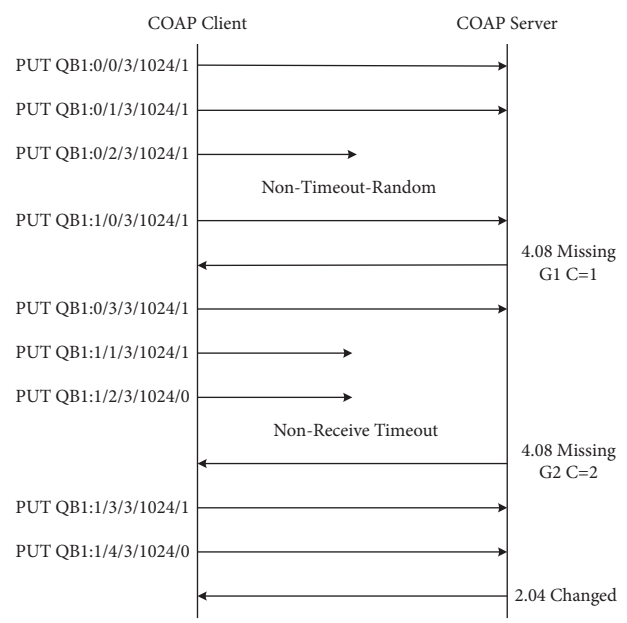

Figure 2: Q-Block1 with NC loss scenario, a resource of two generations ($k = 3$).

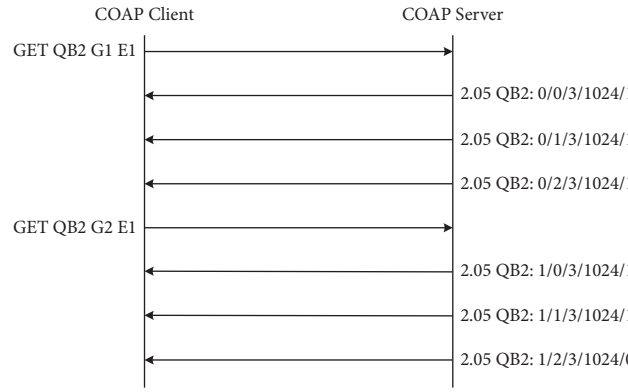

Figure 3: Q-Block2 with NC no loss scenario, a resource of two generations ($k = 3$).

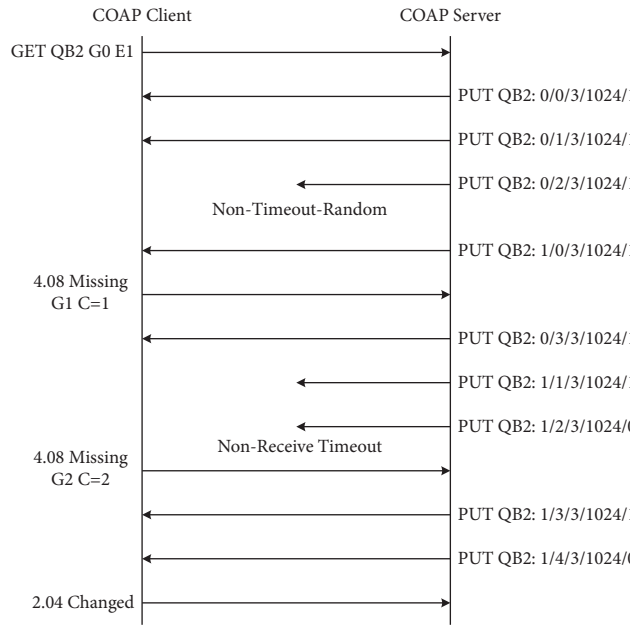

Figure 4: Q-Block2 with NC loss scenario, a resource of two generations ($k = 3$).

while taking in consideration of the computational overhead added which can be a challenge for IoT devices that have limited resources. Homomorphic encryption is gaining momentum due to the ability of performing operations on encrypted data to generate a cipher that can be decrypted at the receiver to generate the data that are the result of the operations performed while being encrypted. Homomorphic encryption allows performing operations on the data while being encrypted which preserves the privacy of the data. Moreover, homomorphic encryption allows performing operations on the encrypted data on the cloud without risking the privacy of the data while being on the cloud. Once the receiver decrypts the received data, it will get the result of the operations performed on the data while being on the cloud.

Homomorphic encryption is a public key encryption scheme. With the homomorphic property, specific arithmetic and logical operations performed on the encrypted data provide encrypted results that when decrypted provide the results of the operations performed on the plain text [23]. For the messages m_1 and m_2, the cipher messages C_1 and C_2 are

$$\begin{aligned} C_1 &= E_{\text{Kpub}}(m_1), \\ C_2 &= E_{\text{Kpub}}(m_2). \end{aligned} \tag{10}$$

For the operation Add,

$$\text{Add}(C_1, C_2) = E_{\text{Kpub}}(m_1 + m_2). \tag{11}$$

Hence, the decrypted message that results from the Add operation is

$$D_{\text{Kpriv}}(\text{Add}(C_1, C_2)) = m_1 + m_2. \tag{12}$$

Homomorphic encryption can be employed with NC to secure the encoding vectors used to generate the encoded

blocks. The size of the encrypted data can be controlled according to the size of the encoding vector. The size of the encoding vector depends on the generation size. This is suitable for IoT applications where the capabilities of communicating devices are limited. Next, we will show how homographic encryption can be applied along with BWT to secure the transfer of the resource blocks.

5.1. At the Sender. For a generation of k blocks, the sender appends an encoding vector to each block. The encoding vector size is k coefficients. The encoding vector of block b_i has the i^{th} coefficient one and all other coefficients zeros. If the NC computations performed are using the finite field GF (2^8), then the size of each coefficient is one byte. The sender then encodes the blocks by multiplying each block (encoding vector along with data) by a randomly generated coefficient from the finite field. At least k encoded blocks are generated at the sender in the same way. Linearly encoding the blocks at the sender protects the sender blocks against losses. Losses require the retransmission of the lost blocks. On the other hand, by linearly encoding blocks at the sender and in case of loss, it is sufficient to send additional encoded blocks that can be used to increase the number of encoded blocks at the receiver to decode and recover the sender blocks. After generating the encoded blocks, the sender encrypts the encoding vector of each encoded block using homomorphic encryption.

5.2. At the Receiver. Once the receiver receives k encoded blocks, it decrypts the encoding vector of each received block. After decrypting the encoding vectors of the k encoded blocks, the receiver performs decoding to generate the k sender blocks. In case the receiver receives less than k encoded blocks, the sender sends additional encoded blocks of the same generation.

For the sender to send additional encoded blocks of a specific generation, it generates an encoding vector to be used to re-encode the k encoded blocks of the generation. Re-encoding is done on the generation encoded blocks that have encrypted encoding vectors. This means in order to generate an additional encoded block, there will be no need for any decryption or encryption of data at the sender.

6. NC Transmission and Computational Overhead

There is a cost for the enhancements achieved by applying NC. The cost of NC comes from the overhead caused by sending the encoding vector used to generate the encoded block along with the encoded block [4, 5]. Also, there is the cost of performing NC operations which are encoding at the sender and decoding at the receiver. Because the encoding vector is encrypted, the size of the encoding vector is an important parameter that affects encryption computational overhead.

6.1. Generation Size and Transmission/Encryption Overhead. The encoding vector size used to generate an encoded block depends on the number of encoded blocks. As the number of blocks increases, the size of the encoding vector increases. The number of blocks in a resource can be controlled by controlling the size of the block. A larger block means a smaller number of blocks, while a smaller block means a larger number of blocks of a resource.

To control the size of the encoding vector, blocks can be grouped in generations. A generation consists of a fixed number of blocks (k). The size of the encoding vector can be controlled by increasing or decreasing the generation size.

A resource of size R bytes consists of G generations where each generation consists of k blocks. NC parameters include the number of blocks per generation (k) and the finite field size used for NC computations. The relationship between the generation size, finite field size, and encoding vector size is shown in Table 1. From the table, it is clear that increasing the generation size increases the size of the encoding vector and hence increases the transmission overhead associated with each encoded block.

Block size (S) also affects the performance of NC. To assess this effect, let us consider a resource of size $R = 1500$ kB. Table 2 shows the number of generations in the resource for different block sizes (S) and different generation sizes (k). As the size of the block increases, the number of blocks per resource decreases, and hence, the number of generations decreases.

Also, Table 2 shows the relationship between the block size (S), generation size (k), and encoding vector size. Since the encoding vector is the part of the block that is encrypted, then decreasing the encoding vector size decreases the encryption computational overhead. Increasing the size of the block decreases the number of resource blocks which decreases the encoding vector overhead.

In conclusion, having smaller block size increases the encoding vector overhead. On the other hand, smaller generation size decreases encoding vector overhead. Moreover, decreasing the encoding vector overhead can be achieved by increasing block size and decreasing generation size. Since the encoding vector is the part of the encoded block that is encrypted, having a smaller encoding vector limits the encryption computational overhead.

6.2. Generation Size and the Reliability of Block Delivery. Increasing the size of the generation increases the overhead that results from the size of the encoding vector sent with the encoded block. On the other hand, increasing the size of the generation may enhance the reliability of block delivery. For a generation of size k, let p be the probability of block loss. If m additional blocks are sent to overcome losses, then the probability of successful delivery (P_k) is as follows [5]:

$$P_k = \sum_{i=k}^{k+m} \binom{k+m}{i} . (1-p)^i . p^{(k+m-i)}. \tag{13}$$

Figure 5 shows the effect of increasing the generation size on the probability of successful delivery. It is clear in the figure that the bigger the generation, the smaller number of additional transmissions needed to provide the receiver with the encoded blocks it needs to decode.

Table 1: Generation size and encoding vector size.

Generation size (k)	Encoding vector size (bytes)	
	GF (2^8)	GF (2^{16})
10	10	20
20	20	40
30	30	60
40	40	80
50	50	100

Table 2: Encoding vector overhead for a resource of 1500 kB for different block sizes.

Block size (S)	Number resource blocks	Number of generations (G)			Encoding vector overhead ($GF2^8$)		
		$k=20$	$k=30$	$k=50$	$k=20$	$k=30$	$k=50$
100	15000	750	500	300	0.2	0.3	0.5
300	5000	250	167	100	0.067	0.1	0.167
800	1875	94	63	38	0.025	0.0375	0.0625
1500	1000	50	34	20	0.0133	0.02	0.033
2000	750	38	25	15	0.01	0.015	0.025

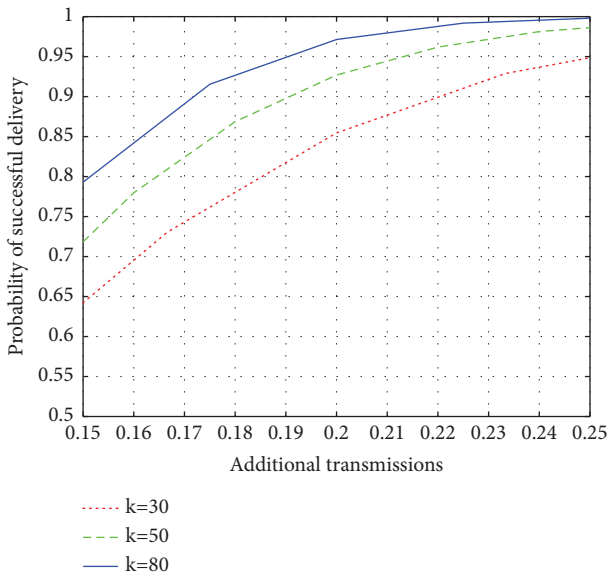

Figure 5: The effect of increasing generation size on the delivery of sender's data.

In conclusion, increasing the generation size enhances the reliability of data transfer. On the other hand, increasing the generation size increases the overhead caused by the increased encoding vector size. To balance the overhead caused by the encoding vector size, the size of the block should be relatively bigger than the encoding vector size. At the same time, increasing the size of the block increases the cost of the additional transmissions needed to achieve successful delivery.

7. Performance Evaluation

In this section, we evaluate the performance of block-wise transfer (BWT) Block1, Block2, Q-Block1, and Q-Block2 options by comparing the total number of transmissions needed to achieve successful delivery of the resource to the destination with and without NC. With Block1 and Block2, each block sent must be acknowledged before sending the next block. An extra transmission is needed in the case of a lost block or a lost acknowledgment. On the other hand, with Q-Block1 and Q-Block2, blocks are sent without waiting for acknowledgments.

Let p be the probability of packet loss, where $p < 1$. A packet could be a resource block or an acknowledgment. Let p_r be the probability of a packet to be critical. A critical packet is a packet whose loss causes the retransmission of a resource block in the case of no NC or the transmission of an additional encoded resource block in the case of NC.

In the case of sending resource blocks (without NC), each packet whether it is a resource block or an acknowledgment is critical. The loss of a resource block or its acknowledgment causes the retransmission of that resource block. On the other hand, in the case of sending encoded resource blocks (with NC), an additional encoded block is sent only in the case of a lost encoded block.

If the resource size is k blocks ($N_R = k$) and if every block sent is acknowledged ($N_R = N_A$), then the total number of packets sent (resource blocks and acknowledgments) in the case of no loss is $2k$.

7.1. BWT Block1/Block2 with or without NC. Without NC, the sender sends the next block after receiving the acknowledgment of the previous one. The block is retransmitted if it is lost or its acknowledgment is lost. A round trip transmission is successful when the resource block is received by the receiver and its acknowledgment is received by the sender.

With NC, the sender sends the next encoded block after receiving the acknowledgment of the previous encoded block. This is the same as in the case of Block1/Block2 without NC. Hence, the performance of BWT Block1/Block2 with NC is the same as that without NC. For a successful

round trip transmission, the number of round trip transmissions performed is

$$\frac{1}{(1-p)^2}. \tag{14}$$

For a resource of k blocks ($N_R = k$), the total number of round trip transmissions which is the total number of sender transmissions needed to deliver the resource is

$$\frac{k}{(1-p)^2}. \tag{15}$$

7.2. BWT Q-Block1/Q-Block2 without NC. The sender can send more than one block without waiting for acknowledgment. The receiver acknowledges received blocks as they are received. The sender retransmits the blocks that were not acknowledged after timeout. A single acknowledgment can be sent for multiple blocks. The total number of sender transmissions needed to successfully deliver a block is

$$\frac{1}{1-p}. \tag{16}$$

For a resource of k blocks ($N_R = k$), the total number of sender transmissions needed to successfully deliver the resource is

$$\frac{k}{1-p}. \tag{17}$$

7.3. BWT Q-Block1/Q-Block2 with NC. The sender sends encoded blocks without waiting for acknowledgments. Acknowledgments are used to inform the sender about the number of encoded blocks received. The sender needs to send an additional encoded block only if a previously sent encoded block was lost.

For N_R resource blocks of and N_A acknowledgments sent while communicating a resource, then

$$p_r = \frac{N_R}{N_R + N_A}. \tag{18}$$

The total number of sender transmissions needed to successfully deliver a block is

$$\frac{1}{1-p\,.\,p_r}. \tag{19}$$

And hence, for a resource of k blocks ($N_R = k$), the total number of sender transmissions needed to successfully deliver the resource is

$$\frac{k}{1-p\,.\,p_r}. \tag{20}$$

Figure 6 shows the total number of transmissions needed to deliver a resource of 100 blocks. It is clear the NC improves the reliability of block delivery so that the total number of transmissions needed is less in the case of NC than that in the case of block transfer without NC.

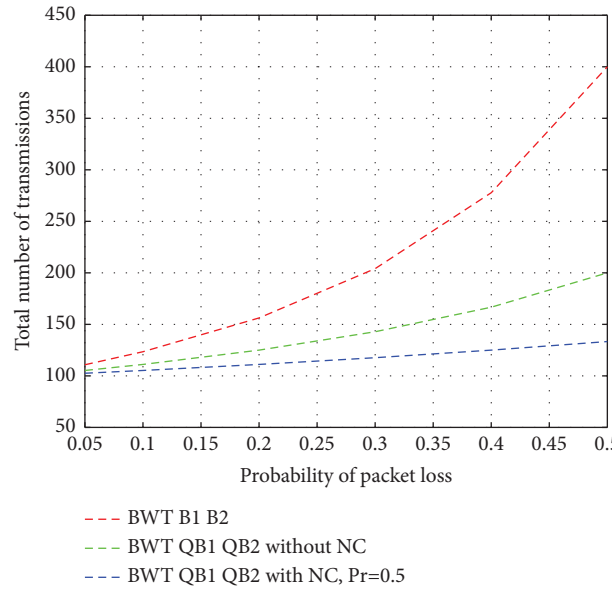

Figure 6: Total number of transmissions needed to deliver a resource of 100 blocks $p_r = 0.5$.

8. Conclusion

In this paper, we have shown that the new options for block-wise transfer (Q-Block1 and Q-Block2) can significantly benefit from NC. The benefits are in terms of securing the transfer of blocks and enhancing the reliability of the transfer. Securing the transfer is achieved by encrypting the encoding vectors used to generate the encoded blocks without the need to encrypt the whole block which limits encryption computational overhead. Moreover, it was shown in the paper that NC enhances the reliability by enabling the receiver to recover sender resource faster after receiving the sufficient number of encoded blocks [27].

Data Availability

The data used to support the findings of this study are available from the corresponding author upon request.

Disclosure

This work was performed by the author as part of his employment as Associate Professor in the Computer Engineering Department at Yarmouk University, Jordan.

Conflicts of Interest

The author declares that there are no conflicts of interest.

References

[1] Z. Shelby, K. Hartke, and A. C. Bormann, "The constrained application protocol (CoAP)," 2014, https://tools.ietf.org/html/rfc7252.

[2] C. Bormann and E. Z. Shelby, "Block-wise transfers in the constrained application protocol CoAP," 2016, https://www.rfc-editor.org/rfc/rfc7959.

[3] R. Ahlswede, N. Cai, S.-Y. R. Li, and R. Yeung, "Network information flow," *IEEE Transactions on Information Theory*, vol. 46, no. 4, pp. 1204–1216, 2000.

[4] P. A. Chou, "Practical network coding," in *Proceedings of the Allerton Conference on Communication, Control, and Computing*, Monticello, II, USA, October 2003.

[5] M. Halloush and H. Radha, "Network coding with multi-generation mixing: a generalized framework for practical network coding," *IEEE Transactions on Wireless Communications*, vol. 10, no. 2, pp. 466–473, 2011.

[6] C. V. Phung, J. Dizdarevic, and A. Jukan, "Enhancing block-wise transfer with network coding in CoAP," in *European Conference on Parallel Processing*, Springer, Cham, Berlin, Germany, 2019.

[7] B. Schütz and N. Aschenbruck, "Adding a network coding extension to CoAP for large resource transfer," in *Proceedings of the IEEE 41st Conference on Local Computer Networks (LCN)*, Dubai, United Arab Emirates, November 2016.

[8] M. Boucadair and J. Shallow, "Constrained application protocol (CoAP) block-wise transfer options supporting robust transmission," 2022, https://www.ietf.org/rfc/rfc9177.pdf.

[9] D. Boneh, E.-J. Goh, and A. K. Nissim, "Evaluating 2-DNF formulas on ciphertexts," in *Theory of Cryptography. TCC 2005. Lecture Notes in Computer Science*, J. Kilian, Ed., Springer, Berlin, Germany, 2005.

[10] P. Paillier, "Public-key cryptosystems based on composite degree residuosity classes," in *Proceedings of the Advances in Cryptology- EUROCRYPT 99, International Conference on the Theory and Application of Cryptographic Techniques*, Prague, Czech Republic, May 1999.

[11] G. Peralta, R. G. Cid-Fuentes, J. Bilbao, and P. M. Crespo, "Homomorphic encryption and network coding in IoT architectures: advantages and future challenges," *Electronics*, vol. 8, no. 8, p. 827, 2019.

[12] E. Ayday, F. Delgosha, and A. F. Fekri, "Location-aware security services for wireless sensor networks using network coding," *IEEE Information and communication*, 2007.

[13] M. Wang and A. B. Li, "Network coding in live peer-to-peer streaming," *IEEE Transactions on Multimedia*, vol. 9, no. 8, pp. 1554–1567, 2007.

[14] M. Halloush and T. Dawahdeh, "Adaptive forwarding using network coding in vector based wireless sensor networks," *International Journal of Sensor Networks*, vol. 14, no. 1, p. 1, 2013.

[15] C. Gkantsidis and A. P. Rodriguez, "Network coding for large scale content distribution," in *Proceedings IEEE 24th Annual Joint Conference of the IEEE Computer and Communications Societie*, San Francisco, CA, USA, March 2005.

[16] Y. Wu, P. A. Chou, and S. Y. Kung, "Minimum-energy multicast in mobile Ad hoc networks using network coding," *IEEE Transactions on Communications*, vol. 53, no. 11, pp. 1906–1918, 2005.

[17] K. Han, T. Ho, R. Koetter, M. Medard, and A. F. Zhao, *On Network Coding for Security*, IEEE MILCOM, Boston, MA, USA, 2007.

[18] C. Gkantsidis and P. R. Rodriguez, *Cooperative Security for Network Coding File Distribution*, IEEE INFOCOM, Las Vegas, NV, USA, 2006.

[19] J. Vilela, L. Lima, and J. Barros, "Lightweight security for network coding," in *Proceedings of the IEEE International Conference on Communications*, Beijing, China, May 2008.

[20] B. Schuetz and N. Aschenbruck, "Adding a network coding extension to CoAP for large resource transfer," in *Proceedings of the 2016 IEEE 41st Conference on Local Computer Networks (LCN)*, pp. 715–722, Dubai, United Arab Emirates, November 2016.

[21] L. García, J. M. Jiménez, M. Taha, and J. Lloret, "Wireless technologies for IoT in smart cities," *Network Protocols and Algorithms*, vol. 10, no. 1, pp. 23–64, 2018.

[22] W. Lu, Y. Mo, Y. Feng et al., "Secure transmission for multi-UAV-assisted mobile edge computing based on reinforcement learning," *IEEE Transactions on Network Science and Engineering*, vol. 10, no. 3, pp. 1270–1282, 2023.

[23] A. Acar, H. Aksu, A. S. Uluagac, and M. Conti, "A survey on homomorphic encryption schemes: theory and implementation," *Association for Computing Machinery Computing Surveys*, vol. 51, no. 4, pp. 1–35, 2019.

[24] W. Lu, Y. Ding, Y. Gao et al., "Secure NOMA-based UAV-MEC network towards a flying eavesdropper," *IEEE Transactions on Communications*, vol. 70, no. 5, pp. 3364–3376, 2022.

[25] Y. Ding, Y. Feng, W. Lu et al., "Online edge learning offloading and resource management for UAV-assisted MEC secure communications," *IEEE Journal of Selected Topics in Signal Processing*, vol. 17, no. 1, pp. 54–65, 2023.

[26] M. Taha, A. Canovas, J. Lloret, and A. Ali, "A QoE adaptive management system for high definition video streaming over wireless networks," *Telecommunication Systems*, vol. 77, no. 1, pp. 63–81, 2021.

[27] P. C. Vien, D. Jasenka, C. Francisco, and J. Admela, "Enhancing REST HTTP with random linear network coding in dynamic edge computing environments," in *Proceedings of the 42nd International Convention on Information and Communication Technology, Electronics and Microelectronics MIPRO*, Taiwan China, May 2019.