An Efficient Edge-Based Authentication for Network Coding Against Entropy Attacks

Ryo Iguchi and Yoshifumi Manabe
Department of Computer Science, Faculty of Informatics
Kogakuin University
1-24-2, Nishi-Shinjuku, Shinjuku, Tokyo, 163-8677 Japan
Email: manabe@cc.kogakuin.ac.jp

***Abstract*—This paper proposes a new edge-based authentication scheme for network coding. Many authentication schemes for random linear network coding have been proposed against pollution attacks. However, random linear network coding is vulnerable to entropy attacks. An adversary can generate messages that are verified as correct messages by the authentication mechanism but obstruct the network coding. Random linear network coding is shown to be efficient in a random failure model, but not in an adversary model.**

This paper shows a simple solution to tolerate entropy attacks by changing random linear coding to deterministic message combining rule. For an example, this paper shows a modification of RIPPLE, an authentication scheme for random linear network coding. Lastly, we show that the total delay of modified RIPPLE can be reduced by an edge-based authentication. RIPPLE and many other authentication schemes are node-based, that is, verification keys and operations are defined for each node. We show that we can construct an edge-based scheme, that is, verification keys and operations are defined for each edge. We show that the edge-based authentication scheme is more efficient than the node-based schemes.

***Keywords*-network coding; random linear network coding; message authentication code;**

I. Introduction

Network coding was first introduced in [1] as a simple and elegant way to achieve the capacity of a network for multicast communications. Network coding allows intermediate nodes between the source node and the destination nodes not only to forward but also to encode the received packets before forwarding them. Random linear network coding [2]–[4] is one of the most commonly used technique to achieve high throughput.

Since the intermediate nodes encode packets, there is a threat that some intermediate nodes do not operate correctly. Malicious nodes may execute a pollution attack or entropy attack.

Pollution attack is injecting corrupted packets. Since correct intermediate nodes encode received packets, the corrupted packet might corrupt all packets in the network through encoding at each node. In order to detect pollution attacks, many authentication schemes have been proposed [5]–[20]. In these schemes, the intermediate nodes can verify received packets, thus the neighbor node of the malicious node can detect the corrupted packets and prevent the pollution by encoding the corrupted packets with correct packets. Several schemes have been shown to be insecure [21], [22]. Comparison of the schemes by transmission efficiency is shown [23].

Though pollution attack is widely considered, the following attack called entropy attack is not so much considered. A malicious node injects a *correct* packet and makes the final destination node impossible to recover the original messages because it cannot receive enough messages to recover. This type of attack is difficult to detect, since the injected packets are correct. Most random linear network coding schemes [5]–[20] do not consider entropy attacks. They are vulnerable to entropy attacks. Very few number of works [24]–[27] have been done for detecting entropy attacks for random linear network coding. Most of them collect coefficients of incoming packets and outgoing packets, and detect that the outgoing packet is not a random combination or is non-innovative. Thus, the computation cost of the detection is high. In addition, the detection is probabilistic, that is, some honest node really makes a random combination of incoming packets, but the output packet can be non-innovative.

Although the effectiveness of random linear network coding is shown in a random failure model [2]–[4], its effectiveness is not clear in an adversary model. We show that entropy adversaries do not need complicated computation to obtain coefficients of non-innovative packets to be injected. Actually, an adversary copies a correct packet to inject non-innovative packets in the connectivity attacks shown in this paper. Thus, randomness of the coefficients is not effective at all for entropy adversaries, because the attack succeeds no matter how the coefficients are set. Therefore, some new scheme for network coding is necessary for the adversary model, especially for the entropy adversary model.

To detect entropy attacks including connectivity attacks, we propose not to execute random linear combination of received packets. We had better go back to the old model that the sender decides an encoding rule at each intermediate node in advance and inform the rule to each node. Each intermediate node can immediately detect entropy attacks at its neighbor node. This paper shows a modification example

to RIPPLE [5], an authentication scheme for random linear network coding. This modification is effective to many other authentication schemes such as [6]–[20] in order to immediately detect entropy attacks. This paper considers RIPPLE because the attack and modification is easy to show since it has an explicit time schedule to send packets.

Above modification has a good side effect to improve performance of RIPPLE and other network coding schemes. The message sending in RIPPLE is node-based scheduling, that is, each node has a single time slot to encode received packets and send them to the neighbors. It is unnecessary to send packets to every neighbor at the same time slot, if some packets are ready to encode from already received packets. Thus, the schedule must be decided by edge-by-edge. This paper proposes a new edge-based authentication mechanism to increase efficiency of network coding. We show that the total delay is decreased by the edge-based authentication mechanism.

The rest of this paper is organized as follows. Section 2 shows an outline of RIPPLE, an authentication scheme for random linear network coding. In Section3, we show the connectivity attack, a simple entropy attack, and propose message combining rule for a solution. Section 4 shows an edge-based efficient authentication scheme.

II. Outline of RIPPLE

This section shows an outline of RIPPLE [5]. The network topology is modeled as a directed acyclic graph $G = (V, E)$. For node v, denote its indegree as $d^+(v)$ and outdegree as $d^-(v)$.

The source node S sends messages to the destination nodes, which is a subset of $V - \{S\}$. S separates a stream of messages into m messages of a fixed size $x_i (1 \leq i \leq m)$. Each message $x_i \in \mathbb{F}_q^n$ is a vector of n symbols, where each symbol is an element of the finite field $\mathbb{F}_q$. All arithmetic operations are executed over $\mathbb{F}_q$.

In order for each destination node to recover the original message from random linear combinations of messages, S generates message

$$M_i = (x_i, \overbrace{\underbrace{0, 0, \ldots, 0, 1}_{i}, 0, \ldots, 0}^{m}) \in \mathbb{F}_q^{n+m},$$

where a vector of size m is added to the payload. Initially, i-th element is 1 and the other elements are 0 for x_i. When internal node v receives message $M_i' (1 \leq i \leq d^+(v))$ from each incoming edge of v, v chooses coefficient $\alpha_{i,j} \in \mathbb{F}_q (1 \leq i \leq d^+(v), 1 \leq j \leq d^-(v))$ uniformly at random. v generates a coded message for j-th outgoing edge $(1 \leq j \leq d^-(v))$ as $M_j'' = \sum_{i=1}^{d^+(v)} \alpha_{i,j} M_i'$. The vector added in M_j'' carries the coefficients of the linear combination. A destination node can recover the original message from any m random linear combinations that form a full rank matrix, where the matrix can be obtained from the coefficients of each received message.

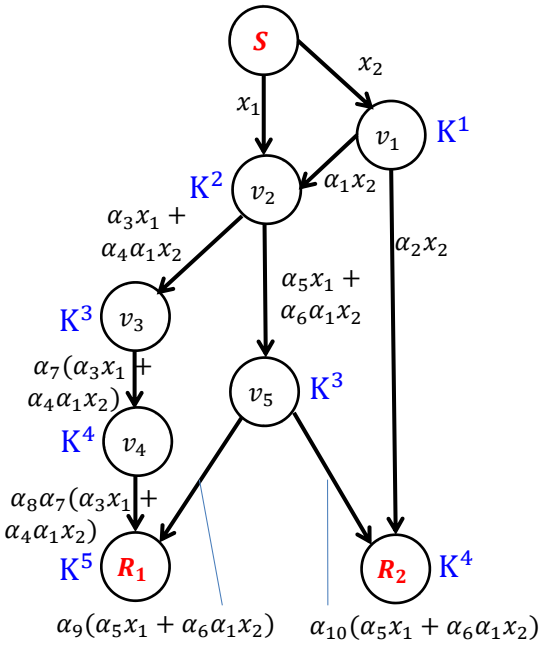

Figure 1. An example of message transmission by RIPPLE.

Figure 1 shows an example of message transmission of x_1 and x_2 from S to two destination R_1, R_2 using RIPPLE. v_2 receives message $M_1' = (x_1, 1, 0)$ from S (Note that the coefficient vector in each message is not written in Fig. 1 for simplicity) and $M_2' = (\alpha_1 x_2, 0, \alpha_1)$ from v_1. v_2 generates random number $\alpha_3, \alpha_4 \in \mathbb{F}_q$, calculates $M_1'' = \alpha_3 M_1' + \alpha_4 M_2' = (\alpha_3 x_1 + \alpha_4 \alpha_1 x_2, \alpha_3, \alpha_4 \alpha_1)$ and sends M_1'' to v_3. v_3 receives $(\alpha_3 x_1 + \alpha_4 \alpha_1 x_2, \alpha_3, \alpha_4 \alpha_1)$ from v_2, generates a random number α_7, calculates $(\alpha_7 \alpha_3 x_1 + \alpha_7 \alpha_4 \alpha_1 x_2, \alpha_7 \alpha_3, \alpha_7 \alpha_4 \alpha_1)$, and sends the message to v_4. The other message sending is similarly executed. R_1 receives message $M_1''' = (y_1, \alpha_8 \alpha_7 \alpha_3, \alpha_8 \alpha_7 \alpha_4 \alpha_1)$ from v_4 and $M_2''' = (y_2, \alpha_9 \alpha_5, \alpha_9 \alpha_6 \alpha_1)$ from v_5, where $y_1 = \alpha_8 \alpha_7 \alpha_3 x_1 + \alpha_8 \alpha_7 \alpha_4 \alpha_1 x_2$ and $y_2 = \alpha_9 \alpha_5 x_1 + \alpha_9 \alpha_6 \alpha_1 x_2$. Let matrix

$$A = \begin{pmatrix} \alpha_8 \alpha_7 \alpha_3 & \alpha_8 \alpha_7 \alpha_4 \alpha_1 \\ \alpha_9 \alpha_5 & \alpha_9 \alpha_6 \alpha_1 \end{pmatrix}.$$

If $Rank(A) = 2$, x_1 and x_2 are recovered by

$$\begin{pmatrix} x_1 \\ x_2 \end{pmatrix} = A^{-1} \begin{pmatrix} y_1 \\ y_2 \end{pmatrix}.$$

RIPPLE uses homomorphic MAC(Message Authentication Code) to detect pollution attacks by adversaries, that is, any modification of packets can be detected. Homomorphism allows combines of tagged messages. Suppose MAC tag t_1 for message M_1 and t_2 for M_2 can be verified by key K. Then, for any constant $\alpha_1, \alpha_2 \in \mathbb{F}_q$, $\alpha_1 t_1 + \alpha_2 t_2$ is

verified as a valid tag for message $\alpha_1 M_1 + \alpha_2 M_2$ by key K. Public-key cryptography or symmetric key cryptography can be used to detect pollution. Since public key cryptography needs high computation cost compared to symmetric key cryptography, RIPPLE uses symmetric key cryptography. Though symmetric key cryptography needs sharing keys in advance, RIPPLE avoids the procedure by a key disclosure mechanism during message transmission. The primitives used in the scheme is as follows.

For a given directed acyclic graph $G = (V, E)$, the message transmission schedule is defined as follows. For each node $v \in V - S$, level of node v, $Lv(v)$, is defined as the length of the maximum directed path from S to v. For the example in Fig. 1, $Lv(v_1) = 1$, $Lv(v_2) = 2$, $Lv(v_3) = Lv(v_5) = 3$, $Lv(v_4) = Lv(R_2) = 4$, and $Lv(R_1) = 5$. The network level L is defined as

$$L \stackrel{def}{=} \max_{v \in V - \{S\}} Lv(v).$$

$L = 5$ for the example in Fig. 1.

For edge $e = (v, v')$, define $T_e = Lv(v)$. T_e is the time slot that a packet is sent via edge e.

Four protocols, generate, MAC, verify, combine, are used in RIPPLE.

- Generate: S generates keys $K^1, K^2, \ldots, K^L$, where $K^j \stackrel{R}{\leftarrow} \mathbb{F}_q^{n+m+L-j}$. Key K^i is shared with node v whose level $Lv(v) = i$.
- MAC: Given message $M \in \mathbb{F}_q^{n+m}$ and keys $K^1, K^2, \ldots, K^L$, S generates the following L tags $t^L, L^{L-1}, \ldots, t^1$ as follows.
  $t^L = \langle M, K^L \rangle$,
  $t^j = \langle (M, t^L, t^{L-1}, \ldots, t^{j+1}), K^j \rangle (j = L-1, L-2, \ldots, 1)$,
  where $\langle X, Y \rangle$ is the inner product of X and Y. S sends packet $P = (M, t^L, t^{L-1}, \ldots, t^1)$ in order to send M.
- Verify: When node v whose level $Lv(v) = i$ receives packet $P = (M, t^L, t^{L-1}, \ldots, t^i)$, v verifies whether $t^i = \langle (M, t^L, t^{L-1}, \ldots, t^{i+1}), K^i \rangle$ using K^i. The tag t^i is removed from P after the verification.
- Combine: After node v whose level $Lv(v) = i$ verifies each received packet $P_j = (M_j, t_j^L, t_j^{L-1}, \ldots, t_j^i) (1 \le j \le d^+(v))$, v generates random coefficients $\alpha_{j,k} \in \mathbb{F}_q (1 \le j \le d^+(v), 1 \le k \le d^-(v))$, outputs combined packet $P'_k = \sum_{j=1}^{d^+(v)} \alpha_{j,k} P_j = (\sum_{j=1}^{d^+(v)} \alpha_{j,k} M_j, \sum_{j=1}^{d^+(v)} \alpha_{j,k} t_j^L, \ldots, \sum_{j=1}^{d^+(v)} \alpha_{j,k} t_j^{i+1})$, for $1 \le k \le d^-(v)$. v sends $P'_1, \ldots, P'_{d^-(v)}$ to each outgoing edge.

Note that if node v whose level $Lv(v) = i$ receives a packet from v' whose level $Lv(v')$ satisfies $Lv(v') < i - 1$, the tags $t^{Lv(v')+1}, \ldots, t^{i-1}$ are removed and then the packet is verified using t^i.

Symmetric keys for verification are defined for each level. Thus, multiple nodes with the same level share the same key. Though it seems that sharing the same key causes the following attack, it is avoided by a key release mechanism. v_1 and v_2 shares the same key K^i if $Lv(v) = Lv(v')$. v_1 can generate a false packet P which passes the verification by v_2 because v_1 knows v_2's verification key K^i. This type of attack is impossible in RIPPLE because K^i is received by the level i nodes at the time just when the packets to be verified arrives. Even if v_1 generates a false packet for v_2 when v_1 receives K^i, the false packets are received by v_2 after the scheduled time of the packet arrival. The false packets are ignored by v_2 because of the late arrival. The detail of the scheduling is written in [5].

III. Connectivity Attack to Random Linear Network Coding

Though RIPPLE is secure to pollution attacks, it is vulnerable to entropy attacks. Entropy attack is replacing a correct packet by another correct packet and making the receiver impossible to recover the original messages. Since each node sends a random linear combination of received packets to each outgoing edge, each node cannot predict the coefficients of the linear combination of each receiving packet. As written in the outline of the scheme, each node just verifies the tag using its symmetric key whatever its coefficients are. Thus, each node cannot distinguish the replaced packet from the original packet.

Thus, the following simple entropy attack, connectivity attack, is possible. A malicious node w selects an edge $e_0 \in E$ and some number of edges $E' \subset E - \{e_0\}$ such that

1) the number of edge-disjoint paths from S to some receiver node R is less than $m - 1$ in $G - \{e_0\} - E'$ and
2) $T_{e_0} < T_e$ for any edge $e \in E'$, that is, a packet is sent at e_0 earlier than at any edge $e \in E'$.

In the example of Fig. 1, the number of edge-disjoint paths from S to R_1 is 2. w selects $e_0 = (v_2, v_5)$ and $E' = \{(v_3, v_4)\}$. When these two edges are removed, there is no edge-disjoint path from S to R_1. $T_{e_0} = 3$ and $T_{e_1} = 4$, where $e_1 = (v_3, v_4)$. For the efficiency of the attack, w should select E' and e_0 such that the size of E' is as small as possible. w might select e_0 as one of the edges connected to w, but this condition is not always necessary.

w obtains a copy of the correct packet P sent via e_0 at time T_{e_0}, discards the correct packet P' sent via each edge $e \in E'$ at time T_e, and sends P instead.

For the packet P sent via e_0 at time T_{e_0}, $P = (M, t^L, t^{L-1}, \ldots, t^{T_{e_0}})$ for some message M ($M = (\sum_{k=1}^m \alpha_k x_k, \alpha_1, \ldots, \alpha_m)$ for some random coefficients $\alpha_k (1 \le k \le m)$). For edge $e = (v, v') \in E'$, $P' = (M, t^L, t^{L-1}, \ldots, t^{T_e})$ (removing the tags between T_{e_0} and $T_e - 1$ from P) is a valid packet for e. When v' receives P', v' verifies P' and decides it as a correct packet. P' is actually a valid packet. Since each intermediate node executes random linear combination, each intermediate

receiving node cannot predict coefficients, thus it cannot distinguish connectivity attack packets from correct packets. When the final destination node R receives all packets, R makes the matrix A that has the coefficients of each received packet, calculates its rank, and realizes that the rank of A is less than m. R cannot recover the original messages from the received packets. This attack succeeds because the number of edge-disjoint paths from S to R is less than $m-1$ in $G-\{e_0\}-E'$, the information sent from S to R via $G-\{e_0\}-E'$ is at most $m-2$ and the total information sent from S to R via $G-E'$ is at most $m-1$. No new information is sent via E', thus the total information sent from S to R by G is at most $m-1$.

A connectivity attack for Fig. 1 is as follows. w captures the packet P sent via (v_2, v_5) at time 3. P is $(\alpha_5 x_1 + \alpha_6\alpha_1 x_2, \alpha_5, \alpha_6\alpha_1)$. Note that w does not need to know the coefficients. w discards the packet $(\alpha_7(\alpha_3 x_1 + \alpha_4\alpha_1 x_2), \alpha_7\alpha_3, \alpha_7\alpha_4\alpha_1)$ sent via (v_3, v_4) at time 4. Instead, w sends P via (v_3, v_4) at time 4 as in Fig. 2.

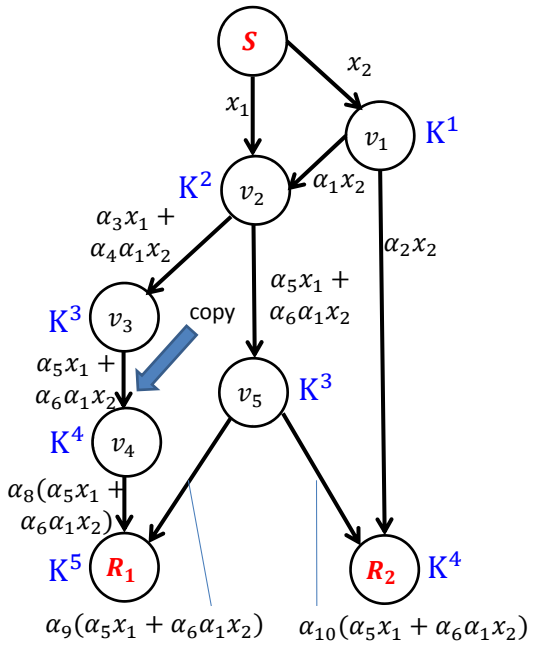

Figure 2. Connectivity attack to RIPPLE.

v_4 verifies this packet as a correct packet. In fact, if v_2 selects α_5/α and α_6/α as the random coefficients to send a packet for v_3 (v_2 selects some other random coefficients for the packet sent to v_5) and v_3 selects α as its random coefficient, P is received by v_4. Thus, v_4 cannot reject this packet.

v_4 generates a random number α_8, and sends $(\alpha_8(\alpha_5 x_1 + \alpha_6\alpha_1 x_2), \alpha_8\alpha_5, \alpha_8\alpha_6\alpha_1)$ to R_1. R receives this packet from v_4 and $(\alpha_9(\alpha_5 x_1 + \alpha_6\alpha_1 x_2), \alpha_9\alpha_5, \alpha_9\alpha_6\alpha_1)$ from v_5. These are the same information, and

$$rank(A) = rank \begin{pmatrix} \alpha_8\alpha_5 & \alpha_8\alpha_6\alpha_1 \\ \alpha_9\alpha_5 & \alpha_9\alpha_6\alpha_1 \end{pmatrix} = 1.$$

R_2 cannot recover the original message x_1 and x_2 from these received packets. Note that replacing packets at edge $e = (v, v')$ can be detected by signing to each sending packet at v and verifying the signature at e'. This simple signing mechanism does not work if v colludes. In order to detect an entropy attack (includes connectivity attack) when some node v might collude, the detection mechanism needs to collect all incoming packets and outgoing packets of v and then the relation must be verified. Thus, the detection needs complicated mechanism to collect packets.

Note that connectivity attack is possible for any other schemes that use random linear combination [6]–[20]. This type of attack can be detected only at the destination node by receiving insufficient information.

The connectivity attack is fatal for all these authentication schemes, because they were proposed to detect attacks at each intermediate node. Connectivity attacks cannot be detected at the intermediate nodes, thus it is unnecessary to verify at each intermediate node and just verify at the final receiver nodes. These authentication schemes lose their most important characteristics by the connectivity attack.

In order to achieve intermediate node detection, we had better go back to the old model that the all schedule is defined by the sender. Though the sender needs to know the current network topology, the modified scheme is tolerant to entropy attacks.

Sender S decides the messages at each intermediate node, that is, the coefficients of each outgoing message are defined in advance. Sender S decides the vector of coefficient $\beta(e)$ for each edge $e \in E$. $\beta(e) = (\alpha_1, \alpha_2, \ldots, \alpha_m)$ means that the message $(\sum_{i=1}^{m} \alpha_i x_i, \alpha_1, \alpha_2, \ldots, \alpha_m)$ must be sent via e. The function β that maps from E to the set of vectors of coefficients is called the message combining rule.

For the example in Fig. 1, S decides the message combining rule as in Fig. 3. In the example, v_2 must send $(x_1 + x_2, 1, 1)$ to v_5 and $(x_1, 1, 0)$ to v_3 (note again that the coefficient vectors are not written in Fig. 3 for simplicity).

The message combining rule is sent from S to each node in advance. For example in Fig. 3, S informs v_4 that the message with coefficients $(1, 0)$ will arrive from v_3. The message combining rule can be sent together with the symmetric key to verify MAC. The information is signed by S to detect forgery by some malicious node. This is not a great additional overhead because the symmetric key to verify MAC must also be signed by S to detect forgery in the original RIPPLE.

If an entropy attack, which includes a connectivity attack is done at some edge, the receiver node immediately detects the attack by checking whether the coefficients of the received message is the same as the message combining

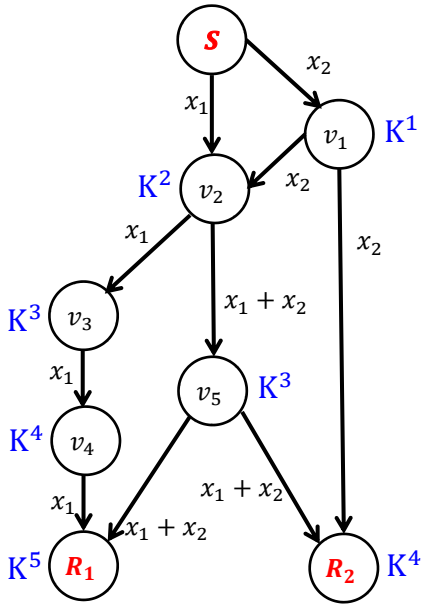

Figure 3. Message combining rule of modified RIPPLE.

rule. In the example of Fig. 3, suppose that a malicious node w selects $e_0 = (v_2, v_5)$ and $E' = \{(v_3, v_4)\}$, copies the packet P sent via e_0, discards the correct packet sent via E', and sends P via E'. $P = (x_1+x_2, 1, 1)$ and the message combining rule $\beta((v_3, v_4)) = (1, 0)$. Thus, the receiver node v_4 immediately detects that the packet is not correct. Entropy attacks can be immediately detected by each intermediate node.

The detail of the modified RIPPLE is as follows.

- Rule: S decides message combining rule β for given network $G = (V, E)$. When $e = (v, v')$, $\beta(e)$ is sent to v and v'.
- Generate, MAC: Same as the original scheme.
- Verify: Suppose that node v whose level $Lv(v) = i$ receives packet $P = (M, t^L, t^{L-1}, \ldots, t^i)$ from edge $e = (\cdot, v)$.
  v verifies whether $t^i = \langle (M, t^L, t^{L-1}, \ldots, t^{i+1}), K^i \rangle$.
  v then checks whether the coefficients of M, $(\alpha_1, \ldots, \alpha_m) = \beta(e)$. If either is not satisfied, reject the packet.
- Combine: After node v whose level $Lv(v) = i$ verifies $P_j = (M_j, t_j^L, t_j^{L-1}, \ldots, t_j^i)(j = 1, \ldots, d^+(v))$, v outputs combined packet to send via $e' = (v, \cdot)$, $P'_k = (\sum_{j=1}^{d^+(v)} \alpha_{j,k} M_j, \sum_{j=1}^{d^+(v)} \alpha_{j,k} t_j^L, \ldots, \sum_{j=1}^{d^+(v)} \alpha_{j,k} t_j^{i+1})$, for $k = 1, \ldots, d^-(v)$, where $\alpha_{j,k}$ is decided such that the coefficients of P'_k, $(\alpha_1^k, \ldots, \alpha_m^k)$ is equal to $\beta(e')$. v sends $P'_1, \ldots, P'_{d^-(v)}$ to each outgoing edge.

In the example of Fig. 3, suppose that v_2 receives packet $P_1 = (M_1, t_1^5, \ldots, t_1^1)$ from S and $P_2 = (M_2, t_2^5, \ldots, t_2^2)$ from v_1. v_2 verifies whether $t_1^2 = \langle (M_1, t_1^5, t_1^4, t_1^3), K^2 \rangle$ and $M_1 = (\cdot, 1, 0)$. If not, v_2 rejects P_1. v_2 then verifies whether $t_2^2 = \langle (M_2, t_2^5, t_2^4, t_2^3), K^2 \rangle$ and $M_2 = (\cdot, 0, 1)$. If not, v_2 rejects P_2. If both packets are correct, v_2 sends $P'_1 = (M_1, t_1^5, t_1^4, t_1^3)$ to v_3 and $P'_2 = (M_1 + M_2, t_1^5 + t_2^5, t_1^4 + t_2^4, t_1^3 + t_2^3)$ to v_5. The other nodes similarly execute the protocol.

IV. Edge-based Efficient Authentication

The modification in the previous section makes possible to reduce the total delay by introducing a novel edge-based authentication mechanism. All previous authentication schemes are node-based, that is, all operations are defined in a node-centric way. In RIPPLE and other many network coding authentication schemes, each node receives all packets from its incoming edges, verifies their tags, combines packets, and sends the combined packets to each outgoing edge. In RIPPLE, network level is defined for each node to decide the time to operate. However, it is unnecessary to wait for every packet to arrive in some cases.

In the example of Figure 3, at time 1, a packet that carries x_1 arrives at v_2 from S. It is unnecessary for v_2 to wait for the arrival of the packet that carries x_2 at time 2 from v_1, in order to send x_1 to v_3. v_2 can send a packet that carries x_1 to v_3 at time 2. At time 3, v_2 sends a packet that carries $x_1 + x_2$ to v_5 using the packet that arrives from v_1 at time 2 (and the packet that arrived from S at time 1). v_3 and v_4 send x_1 at time 3 and 4, respectively. v_5 sends $x_1 + x_2$ to R_1 at time 4. Thus R_1 receives all messages at time 4 as in Fig. 4. In the original RIPPLE, R_1 receives all messages at time 5. Thus the modified edge-based scheme is more efficient than the original RIPPLE.

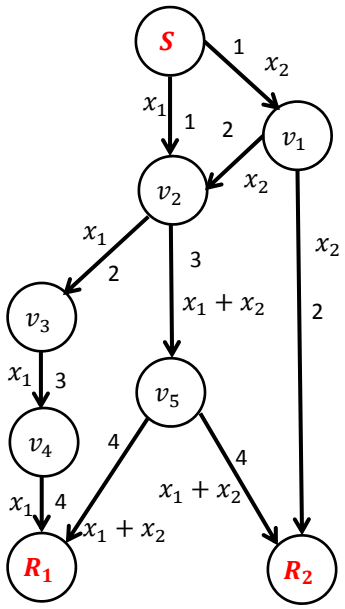

Figure 4. Edge level for the message combining rule β.

The edge-based authentication mechanism is formally defined as follows.

The edge level Ev is defined for each edge $e \in E$ by a given message combining rule β.

- For each edge $e = (S, \cdot) \in E$, $Ev(e, \beta) = 1$ ($Ev(e, \beta) = 1$ for edge whose source node is S).
- For each edge $e = (v, v') \in E$ whose source node is not S, let $c(e, \beta)$ be the set of edges $e' = (\cdot, v) \in E'$ such that the messages received from e' can derive a message whose coefficient is $\beta(e)$.
  $Ev(e, \beta) = 1 + max_{e' \in c(e,\beta)} Ev(e', \beta)$.
- The edge-based network level, $EL(G, \beta)$ is defined as

$$EL(G, \beta) = max_{e \in E} Ev(e, \beta).$$

Edge level means that v can send a packet P to send via $e = (v, \cdot)$ when v receives all packets that are enough to generate P. The edge-based network level is the total delay to send all messages to all final destinations.

For the graph G and message combining rule β in Fig. 4, $c((v_1, v_2), \beta) = \{(S, v_1)\}$, $c((v_2, v_3), \beta) = \{(S, v_2)\}$, $c((v_2, v_5), \beta) = \{(S, v_2), (v_1, v_2)\}$ and so on. Thus, $Ev((S, v_1), \beta) = Ev((S, v_2), \beta) = 1$, $Ev((v_2, v_3), \beta) = 2$, $Ev((v_2, v_5), \beta) = 3$, and so on. The edge-based network level $EL(G, \beta) = 4$.

Theorem 1: For any network $G = (V, E)$ and any message combining rule β, $EL(G, \beta) \leq L$.

Proof: For each edge $e = (v, v')$, $Ev(e, \beta) \leq Lv(v)$ holds because in order to send a message via e, all messages sent to v might not be necessary.

Since $EL(G, \beta) = max_{e \in E} Ev(e, \beta)$ and $L = max_{v \in V - S} Lv(v)$, $EL(G, \beta) \leq L$ holds. ■

The delay of the edge-based authentication is never worse than RIPPLE. As shown in Fig. 4, there are cases when the edge-based authentication is better than RIPPLE.

Node v must have multiple keys to verify packets. In the example of Fig. 4, v_2 receives $P_1 = ((x_1, 1, 0), t_1^4, t_1^3, t_1^2, t_1^1)$ from S and $P_2 = ((x_2, 0, 1), t_2^4, t_2^3, t_2^2)$ from v_1. Thus, v_2 needs K^1 to verify P_1 and K^2 to verify P_2. Though some nodes have multiple keys, the nodes cannot make forged packets using the keys. Because each node receives each key just at the verification time, the node cannot make a false packet that can be verified by other nodes using the key before the arrival time limit.

The detail of edge-based RIPPLE is as follows.

- Rule: S decides message combining rule β for given network $G = (V, E)$. When $e = (v, v')$, $\beta(e)$ is sent to v and v'.
- Generate: Sgenerates keys $K^1, K^2, \ldots, K^{EL(G,\beta)}$, where $K^j \xleftarrow{R} \mathbb{F}_q^{n+m+EL(G,\beta)-j}$.
  Key K^i is sent to node v that has edge $e = (\cdot, v)$ such that $Ev(e, \beta) = i$.
- MAC: Suppose that from β, M is forwarded at most $z (\leq EL(G, \beta))$ hops.
  Given message $M \in \mathbb{F}_q^{n+m}$, keys $K^1, K^2, \ldots, K^{EL(G,\beta)}$, and message sending schedule β, S generates the following z tags.
  $t^z = \langle M, K^z \rangle$.
  $t^j = \langle (M, t^z, t^{z-1}, \ldots, t^{j+1}), K^j \rangle (j = z - 1, z - 2, \ldots, 1)$.
  S sends packet $P = (M, t^z, t^{z-1}, \ldots, t^1)$ to each outgoing edge according to β.
- Verify: Suppose that node v receives packet $P = (M, t^z, t^{z-1}, \ldots, t^i)$ from edge $e = (\cdot, v)$ such that $Ev(e, \beta) = i$ at time i.
  v verifies whether $t^i = \langle (M, t^z, t^{z-1}, \ldots, t^{i+1}), K^i \rangle$.
  v then verifies whether the coefficients of M, $(\alpha_1, \ldots, \alpha_m) = \beta(e)$. If either is not satisfied, reject the packet.
  Note that P might also be used to combine a sending packet P' for edge $e' = (v, v'')$ such that $Ev(e') = j (j > i + 1)$. In the case, the tags $t^{i+1}, \ldots, t^{j-1}$ are removed from P at combining P'.
- Combine: At time j, node v combines sending packet P' to edge $e' = (v, v'')$ such that $Ev(e', \beta) = j$ from packets $P_k = (M_k, t_k^z, \ldots, t_k^j) (1 \leq k \leq l)$ received before time $j - 1$.
  $P' = (\sum_{i=1}^{l} \alpha_i M_i, \sum_{i=1}^{l} \alpha_i t_i^z, \ldots, \sum_{i=1}^{l} \alpha_i t_i^j)$, where $\alpha_i (1 \leq i \leq l)$ are decided such that the coefficients of $\sum_{i=1}^{l} \alpha_i M_i$, $(\gamma_1, \ldots, \gamma_m)$, is equal to $\beta(e')$.
  v sends P' via e'.

V. Conclusion

This paper showed a new type of entropy attack, connectivity attack, to network coding authentication schemes. We showed an improvement of RIPPLE to tolerate entropy attacks. Lastly, we showed the total delay can be reduced by edge-based authentication. This attack is effective for the other random linear network coding authentication schemes. The modification technique can also be applied to the other random linear network coding authentication schemes.

References

[1] R. Ahlswede, N. Cai, S. yen Robert Li, and R. W. Yeung, "Network information flow," *IEEE Transactions on Information Theory*, vol. 46, pp. 1204–1216, 2000.

[2] T. Ho, M. Mdard, R. Koetter, D. R. Karger, M. Effros, J. Shi, and B. Leong, "A Random Linear Network Coding Approach to Multicast," *IEEE Transactions on Information Theory*, vol. 52, pp. 4413–4430, 2006.

[3] T. Ho, R. Koettert, M. Medard, D. R. Karger, and M. Effrost, "The benefits of coding over routing in a randomized setting," in *IEEE International Symposium on Information Theory*, 2003.

[4] P. A. Chou, Y. Wu, and K. Jain, "Practical network coding," in *Allerton conference on communication control and computing*, 2003.

[5] Y. Li, H. Yao, M. Chen, S. Jaggi, and A. Rosen, "RIPPLE Authentication for Network Coding," in *IEEE INFOCOM*, 2010, pp. 2258–2266.

[6] X. Wu, Y. Xu, L. Xiang, and W. Xu, "A Hybrid Scheme against Pollution Attack to Network Coding," in *Workshop on Network Coding, Theory and Applications*, 2011.

[7] X. Wu, Y. Xu, C. Yuen, and L. Xiang, "A tag encoding scheme against pollution attack to linear network coding," *Parallel and Distributed Systems, IEEE Transactions on*, vol. 25, no. 1, pp. 33–42, Jan 2014.

[8] S. Agrawal and D. Boneh, "Homomorphic macs: Mac-based integrity for network coding," in *Applied Cryptography and Network Security*, ser. Lecture Notes in Computer Science, M. Abdalla, D. Pointcheval, P.-A. Fouque, and D. Vergnaud, Eds. Springer Berlin Heidelberg, 2009, vol. 5536, pp. 292–305.

[9] N. Attrapadung and B. Libert, "Homomorphic network coding signatures in the standard model," in *Public Key Cryptography PKC 2011*, ser. Lecture Notes in Computer Science, D. Catalano, N. Fazio, R. Gennaro, and A. Nicolosi, Eds. Springer Berlin Heidelberg, 2011, vol. 6571, pp. 17–34.

[10] S. Agrawal, D. Boneh, X. Boyen, and D. Freeman, "Preventing pollution attacks in multi-source network coding," in *Public Key Cryptography PKC 2010*, ser. Lecture Notes in Computer Science, P. Nguyen and D. Pointcheval, Eds. Springer Berlin Heidelberg, 2010, vol. 6056, pp. 161–176.

[11] A. Le and A. Markopoulou, "TESLA-Based Defense against Pollution Attacks in P2P Systems with Network Coding," in *Workshop on Network Coding, Theory and Applications*, 2011.

[12] R. Gennaro, J. Katz, H. Krawczyk, and T. Rabin, "Secure network coding over the integers," in *Public Key Cryptography PKC 2010*, ser. Lecture Notes in Computer Science, P. Nguyen and D. Pointcheval, Eds., vol. 6056. Springer Berlin Heidelberg, 2010, pp. 142–160.

[13] D. Catalano, D. Fiore, and B. Warinschi, "Efficient network coding signatures in the standard model," in *Public Key Cryptography - PKC 2012*, ser. Lecture Notes in Computer Science, M. Fischlin, J. Buchmann, and M. Manulis, Eds. Springer Berlin Heidelberg, 2012, vol. 7293, pp. 680–696.

[14] D. CHARLES, K. JAIN, and K. LAUTER, "Signatures for Network Coding," in *Conference on Information Sciences and Systems*, 2006.

[15] Y. Jiang, H. Zhu, M. Shi, and C. Lin, "An efficient dynamic-identity based signature scheme for secure network coding," *Computer Networks*, vol. 54, pp. 28–40, 2010.

[16] C. Cheng, T. Jiang, and Q. Zhang, "Tesla-based homomorphic mac for authentication in p2p system for live streaming with network coding," *Selected Areas in Communications, IEEE Journal on*, vol. 31, no. 9, pp. 291–298, September 2013.

[17] Z. Tang, "Homomorphic authentication codes for network coding," *Concurrency and Computation: Practice and Experience*, 2013.

[18] C. Cheng and T. Jiang, "A Novel Homomorphic MAC Scheme for Authentication in Network Coding," *IEEE Communications Letters*, vol. 15, pp. 1228–1230, 2011.

[19] P. Zhang, Y. Jiang, C. Lin, H. Yaot, A. Wasef, and X. Shenz, "Padding for orthogonality: Efficient subspace authentication for network coding," in *IEEE INFOCOM*, 2011, pp. 1026–1034.

[20] W. Yan, M. Yang, L. Li, and H. Fang, "Short Signatures for Multi-source Network Coding," in *International Conference on Multimedia Information Networking and Security*, 2009.

[21] Y. Wang, "Insecure "provably secure network coding" and homomorphic authentication schemes for network coding," *Cryptology ePrint Archive, Report 2010/060*, 2010, http://eprint.iacr.org/.

[22] J. Zhang, X. Li, and F.-W. Fu, "Security analysis on "an authentication code against pollution attacks in network coding"," *CoRR*, vol. abs/1303.0557, 2013.

[23] S. Pfennig and E. Franz, "Comparison of different secure network coding paradigms concerning transmission efficiency," in *Computer Aided Modeling and Design of Communication Links and Networks (CAMAD), 2013 IEEE 18th International Workshop on*, Sept 2013, pp. 18–22.

[24] C. Gkantsidis and P. R. Rodriguez, "Cooperative Security for Network Coding File Distribution," in *IEEE INFOCOM*, 2006, pp. 1–13.

[25] Y. Jiang, Y. Fan, and C. Lin, "A self-adaptive probabilistic packet filtering scheme against entropy attacks in network coding," *Computer Networks*, vol. 53, pp. 3089–3101, 2009.

[26] A. J. Newell, R. Curtmola, and C. Nita-Rotaru, "Entropy attacks and countermeasures in wireless network coding," in *Proceedings of the Fifth ACM Conference on Security and Privacy in Wireless and Mobile Networks*, ser. WISEC '12. New York, NY, USA: ACM, 2012, pp. 185–196.

[27] R. A. Popa, A. Chiesa, T. Badirkhanli, and M. Médard, "Going Beyond Pollution Attacks: Forcing Byzantine Clients to Code Correctly," *ArXiv e-prints*, Aug. 2011.