Homomorphic MACs: MAC-based Integrity for Network Coding

Shweta Agrawal[1] and Dan Boneh[2⋆]

[1] U.T. Austin — shweta@cs.utexas.edu
[2] Stanford University — dabo@cs.stanford.edu

Abstract. Network coding has been shown to improve the capacity and robustness in networks. However, since intermediate nodes modify packets en-route, integrity of data cannot be checked using traditional MACs and checksums. In addition, network coded systems are vulnerable to pollution attacks where a single malicious node can flood the network with bad packets and prevent the receiver from decoding the packets correctly. Signature schemes have been proposed to thwart such attacks, but they tend to be too slow for online per-packet integrity. They also force network coding coefficients to be picked from a large field which causes the size of the network coding header to be large.
Here we propose a *homomorphic MAC* which allows checking the integrity of network coded data. Our homomorphic MAC is designed as a drop-in replacement for traditional MACs (such as HMAC) in systems using network coding.

1 Introduction

Network coding [1, 8] proposes to replace the traditional 'store and forward' paradigm in networks by more intelligent routing that allows intermediate nodes to transform the data in transit. Network coding has become popular due to its robustness and the improved throughput it offers.

When transmitting a message using linear network coding [11] the sender first breaks the message into a sequence of m vectors $\hat{\mathbf{v}}_1, \ldots, \hat{\mathbf{v}}_m$ in an n-dimensional linear space $\mathbb{F}_q^n$, where n, m and q are fixed ahead of time. Often $q = 2^8$ so that the entire transmitted message is $n \times m$ bytes. The sender transmits these message vectors to its neighboring nodes in the network. As the vectors traverse the network, moving from one node to the next on their way to the destination, the nodes randomly combine the vectors with each other. More precisely, each node in the network creates a random linear combination of the vectors it receives and transmits the resulting linear combination to its adjacent nodes. Intended recipients thus receive random linear combinations of the original message vectors. Recipients can recover the original message from any set of m random linear combinations that form a full rank matrix.

For this approach to work, every vector $\hat{\mathbf{v}}$ in the network must carry with it the coefficients $\alpha_1, \ldots, \alpha_m \in \mathbb{F}_q$ that produce $\hat{\mathbf{v}}$ as a linear combination of the original message vectors. To do so, prior to transmission, the source node augments every message vector $\hat{\mathbf{v}}_i$ with m additional components. The resulting vectors $\mathbf{v}_1, \ldots, \mathbf{v}_m$, called *augmented vectors*, are given by:

$$\mathbf{v}_i = (—\hat{\mathbf{v}}_i—,\ \overbrace{\underbrace{0, \ldots, 0, 1}_{i}, 0, \ldots, 0}^{m}) \quad \in \mathbb{F}_q^{n+m} \tag{1}$$

i.e., each original vector $\hat{\mathbf{v}}_i$ is appended with the vector of length m containing a single '1' in the ith position. These augmented vectors are then sent by the source as packets in the network.

⋆ Supported by DARPA IAMANET, NSF, and the Packard Foundation.

Observe that if $\mathbf{y} \in \mathbb{F}_q^{n+m}$ is a linear combination of $\mathbf{v}_1, \ldots, \mathbf{v}_m \in \mathbb{F}_q^{n+m}$ then the linear combination coefficients are contained in the last m coordinates of $\mathbf{y}$.

Pollution attacks. Our brief description of linear network coding assumes all nodes are honest. However, if some nodes are malicious and forward invalid linear combinations of received vectors, then recipients obtain multiple packets, only some of which are proper linear combinations of the original message vectors. In such a scenario, recipients have no way of telling which of their received vectors are corrupt and should be ignored during decoding.

Detailed discussion of pollution attacks can be found in [2, 13, 7]. Here we only note that pollution attacks cannot be mitigated by standard signatures or MACs. Clearly, signing the augmented message vectors is of no use since recipients do not have the original message vectors and therefore cannot verify the signature. Similarly, signing the entire message prior to transmission does not work. To see why, observe that recipients can obtain multiple vectors where, say, only half are proper linear combinations of the original message vectors and the other half are corrupt. Recipients would need to decode exponentially many m-subsets until they find a decoded message that is consistent with the signature (decoding produces the correct transmitted message only when all m vectors being decoded are a linear combination of the original message vectors). In summary, as explained in [2, 13, 7], new integrity mechanisms are needed to mitigate pollution attacks.

Previous solutions. Recently, several approaches have been proposed to thwart pollution attacks. Of these, some solutions are information theoretic while others are cryptographic. We refer to [2] for a survey of defenses. Here we restrict our attention to cryptographic solutions. Several authors [5, 9, 13, 2] devised digital signature schemes for signing a linear subspace. Let V be the linear space spanned by the augmented message vectors $\mathbf{v}_1, \ldots, \mathbf{v}_m \in \mathbb{F}_q^{n+m}$ that the sender transmits. These signature schemes produce a signature σ on V such that $\mathsf{Verify}(PK, \mathbf{v}, \sigma)$ holds for every $\mathbf{v} \in V$, but it is difficult to construct a vector $\mathbf{y} \notin V$ for which $\mathsf{Verify}(PK, \mathbf{y}, \sigma)$ holds. Recipients use these signatures to reject all received vectors that are not in the subspace V, mitigating the pollution problem.

The digital signature constructions in [5, 9, 13, 2] are very elegant and very appropriate for offline network coding systems, such as robust distributed file storage [6]. For online traffic, however, these systems are too slow to sign every packet. A different solution is needed if one is to defend against pollution attacks at line speeds.

Another difficulty with these methods is that they require the network coding coefficients to live in a field $\mathbb{F}_q$ where q is the order of a group where discrete-log is difficult, e.g. $q \approx 2^{160}$. Therefore transmitting each coefficient requires 20 bytes and hence the augmentation components add $20 \times m$ bytes to every packet. Recall that in typical linear network coding (without integrity) $q = 2^8$ so that the augmentation components add only m bytes to every packet. Again, a different solution is needed if one wishes to minimize the augmentation overhead.

Our contribution. We design a MAC scheme that can be used to mitigate pollution attacks. We construct the MAC in three steps.

First, we construct a homomorphic MAC. That is, given two (vector,tag) pairs $(\mathbf{v}_1, t_1)$ and $(\mathbf{v}_2, t_2)$ anyone can create a valid tag t for the vector $\mathbf{y} = \alpha_1\mathbf{v}_1 + \alpha_2\mathbf{v}_2$ for any $\alpha_1, \alpha_2 \in \mathbb{F}_q$. Roughly speaking, our security proof shows that, even under a chosen message attack, creating a valid tag for a vector outside the linear span of the original message vectors is difficult. We give the details in Section 2 and 3. Our MAC is related to the classic MAC of Carter and Wagman [4].

This MAC system can be used to mitigate pollution attacks when the source and recipient have a shared secret key. The source uses the secret key to compute a tag for each of the original message vectors. Intermediate nodes then use the homomorphic property to compute valid tags for random linear combinations they produce. The recipient verifies the tags of received vectors and drop all vectors with an invalid tag.

Network coding (for a single source) is most useful in broadcast settings where there are multiple recipients for every message. Using our basic homomorphic MAC the sender would need to have a shared secret key with each recipient. In addition, if we want intermediate nodes to verify tags before forwarding them on to other nodes, then the sender would need to have a shared secret key with each network node. Every packet would need to include a tag per network node, which is unacceptable.

Our second step converts the homomorphic MAC into a *broadcast* homomorphic MAC using a technique of Canetti et al. [3]. This enables any network node to validate vectors it receives. The limitation of this construction is that it is only c-collusion resistant for some pre-determined c. That is, when more than c recipients and intermediate nodes collude, the MAC becomes insecure. In some settings, it may be possible to apply TESLA-like methods [12] to convert our homomorphic MAC into a broadcast MAC. We do not explore this here since we want to enable intermediate network nodes to verify packet integrity before forwarding packets.

Our third step converts the broadcast homomorphic MAC into an integrity system where there are multiple senders and multiple verifiers. The result is a network coding MAC in networks where anyone can be either a sender, a recipient, or an intermediate node.

We experimented with our construction and give running time estimates in Section 7.

Notation: Throughout the paper we let $[m]$ denote the set of integers $\{1, \ldots, m\}$. For vectors $\mathbf{u}$ and $\mathbf{v}$ in $\mathbb{F}_q^n$ we let $\mathbf{u} \cdot \mathbf{v} \in \mathbb{F}_q$ denote the inner product of $\mathbf{u}$ and $\mathbf{v}$.

2 Homomorphic MACs: Definitions

We begin by defining homomorphic MACs and their security. A (q, n, m) *homomorphic MAC* is defined by three probabilistic, polynomial-time algorithms, (Sign, Verify, Combine). The Sign algorithm computes a tag for a vector space $V = \text{span}(\mathbf{v}_1, \ldots, \mathbf{v}_m)$ by computing a tag for one basis vector at a time. Combine implements the homomorphic property and Verify verifies vector-tag pairs. Each vector space V is identified by an identifier id which is chosen arbitrarily from a set $\mathcal{I}$. In more detail, these algorithms provide the following functionality:

- $\mathsf{Sign}(k, \mathsf{id}, \mathbf{v}, i)$: Input: a secret key k, a vector space identifier id, an augmented vector $\mathbf{v} \in \mathbb{F}_q^{n+m}$, and $i \in [m]$ indicating that $\mathbf{v}$ is the ith basis vector of the vector space identified by id. Output: tag t for $\mathbf{v}$.

  As explained above, the Sign algorithm signs a vector space $V \subseteq \mathbb{F}_q^{n+m}$ spanned by $\mathbf{v}_1, \ldots, \mathbf{v}_m$ by running $\mathsf{Sign}(k, \mathsf{id}, \mathbf{v}_i, i)$ for $i = 1, \ldots, m$. This produces a tag for each of the basis vectors. The identifier id identifies the vector space V. When transmitting a vector $\mathbf{v}_i$ into the network, the sender transmits $(\mathsf{id}, \mathbf{v}_i, t_i)$. Recipients collect all valid vectors with a given id and decode those as a group to obtain the original basis vectors encoding the transmitted message. We let $\mathcal{I}$ denote the set of identifiers and $\mathcal{K}$ denote the set of keys.

– $\mathsf{Combine}((\mathbf{v}_1, t_1, \alpha_1), \ldots, (\mathbf{v}_m, t_m, \alpha_m))$: Input: m vectors $\mathbf{v}_1, \ldots, \mathbf{v}_m \in \mathbb{F}_q^{n+m}$ and their tags $t_1, \ldots, t_m$ under key k plus m constants $\alpha_1, \ldots, \alpha_m \in \mathbb{F}_q$.
Output: a tag t on the vector $\mathbf{y} := \sum_{i=1}^{m} \alpha_i \mathbf{v}_i \ \in \mathbb{F}_q^{n+m}$.

– $\mathsf{Verify}(k, \mathsf{id}, \mathbf{y}, t)$: Input: a secret key k, an identifier id, a vector $\mathbf{y} \in \mathbb{F}_q^{n+m}$, and a tag t.
Output: 0 (reject) or 1 (accept).

We require that the scheme satisfy the following correctness property. Let V be an m-dimensional subspace of $\mathbb{F}_q^{n+m}$ with basis $\mathbf{v}_1, \ldots, \mathbf{v}_m$ and identifier id. Let $k \in \mathcal{K}$ and $t_i := \mathsf{Sign}(k, \mathsf{id}, \mathbf{v}_i, i)$ for $i = 1, \ldots, m$. Let $\alpha_1, \ldots, \alpha_m \in \mathbb{F}_q$. Then

$$\mathsf{Verify}\left(k,\ \mathsf{id},\ \sum_{i=1}^{m} \alpha_i \mathbf{v}_i,\ \mathsf{Combine}\big((\mathbf{v}_1, t_1, \alpha_1), \ldots, (\mathbf{v}_m, t_m, \alpha_m)\big)\right) = 1$$

Security. Next, we define security for homomorphic MACs. We allow the attacker to obtain the signature on arbitrary vector spaces of its choice (analogous to a chosen message attack on MACs). Each vector space V_i submitted by the attacker has an identifier id_i. The attacker should be unable to produce a valid triple $(\mathsf{id}, \mathbf{y}, t)$ where either id is new or $\mathsf{id} = \mathsf{id}_i$ but $\mathbf{y} \notin V_i$. More precisely, we define security using the following game.

Attack Game 1. let $\mathcal{T} = (\mathsf{Sign}, \mathsf{Combine}, \mathsf{Verify})$ be a (q, n, m) homomorphic MAC. We define security of $\mathcal{T}$ using the following game between a challenger C and an adversary $\mathcal{A}$.

Setup. The challenger generates a random key $k \xleftarrow{\text{R}} \mathcal{K}$.

Queries. $\mathcal{A}$ adaptively submits MAC queries where each query is of the form (V_i, id_i) where V_i is a linear subspace (represented by a basis of m vectors) and id_i is a space identifier. We require that all identifiers id_i submitted by $\mathcal{A}$ are distinct. To respond to a query for (V_i, id_i) the challenger does:

Let $\mathbf{v}_1, \ldots, \mathbf{v}_m \in \mathbb{F}_q^{n+m}$ be a basis for V_i
for $j = 1, \ldots, m$ let $t_j \xleftarrow{\text{R}} \mathsf{Sign}(k, \mathsf{id}_i, \mathbf{v}_j, j)$ // compute MAC for all basis vectors
send $(t_1, \ldots, t_m)$ to $\mathcal{A}$

Output. The adversary $\mathcal{A}$ outputs an identifier id^*, a tag t^*, and a vector $\mathbf{y}^* \in \mathbb{F}_p^{n+m}$.

The adversary *wins* the security game if $\mathsf{Verify}(k, \mathsf{id}^*, \mathbf{y}^*, t^*) = 1$, and either

1. $\mathsf{id}^* \neq \mathsf{id}_i$ for all i (a *type 1 forgery*), or
2. $\mathsf{id}^* = \mathsf{id}_i$ for some i and $\mathbf{y}^* \notin V_i$ (a *type 2 forgery*)

Moreover, let $\mathbf{y}^* = (y_1^*, \ldots, y_{n+m}^*)$. Then the augmentation $(y_{n+1}^*, \ldots, y_{n+m}^*)$ in $\mathbf{y}^*$ is not the all zero vector (which corresponds to a trivial forgery).

The *advantage* NC-Adv$[\mathcal{A}, \mathcal{T}]$ of $\mathcal{A}$ with respect to $\mathcal{T}$ is defined to be the probability that $\mathcal{A}$ wins the security game.

Definition 1. A (q, n, m) homomorphic MAC scheme $\mathcal{T}$ is secure if for all polynomial time adversaries $\mathcal{A}$ the quantity NC-Adv$[\mathcal{A}, \mathcal{T}]$ is negligible.

3 Construction 1: A Homomorphic MAC

In this section we describe a secure homomorphic MAC. Our construction is related to a classic MAC system due to Carter and Wagman [4].

The MAC scheme HomMac: To construct a (q, n, m) homomorphic MAC we use a Pseudo Random Generator $G : \mathcal{K}_{\mathrm{G}} \to \mathbb{F}_q^{n+m}$ and a Pseudo Random Function $F : \mathcal{K}_{\mathrm{F}} \times (\mathcal{I} \times [m]) \to \mathbb{F}_q$. Keys for our MAC consist of pairs (k_1, k_2) where $k_1 \in \mathcal{K}_{\mathrm{G}}$ and $k_2 \in \mathcal{K}_{\mathrm{F}}$.

– $\mathsf{Sign}(k, \mathsf{id}, \mathbf{v}, i)$: To generate a tag for an ith basis vector $\mathbf{v} \in \mathbb{F}_q^{n+m}$ using key $k = (k_1, k_2)$ do:
  (1) $\mathbf{u} \leftarrow G(k_1) \ \in \mathbb{F}_q^{n+m}$
  (2) $b \leftarrow F\big(k_2,\ (\mathsf{id}, i)\big) \in \mathbb{F}_q$
  (3) $t \leftarrow (\mathbf{u} \cdot \mathbf{v}) + b \quad \in \mathbb{F}_q$
  Output t. Note that the tag is a single element of $\mathbb{F}_q$.

– $\mathsf{Combine}((\mathbf{v}_1, t_1, \alpha_1), \ldots, (\mathbf{v}_m, t_m, \alpha_m))$: output $t \leftarrow \sum_{j=1}^{m} \alpha_j t_j \ \in \mathbb{F}_q$.

– $\mathsf{Verify}(k, \mathsf{id}, \mathbf{y}, t)$: Let $k = (k_1, k_2)$ be a secret key and let $\mathbf{y} = (y_1, \ldots, y_{n+m}) \in \mathbb{F}_q^{n+m}$.
  Do the following:
  $\mathbf{u} \leftarrow G(k_1) \ \in \mathbb{F}_q^{n+m}$ and $a \leftarrow (\mathbf{u} \cdot \mathbf{y}) \in \mathbb{F}_q$
  $b \leftarrow \sum_{i=1}^{m} \big[y_{n+i} \cdot F\big(k_2,\ (\mathsf{id}, i)\big)\big] \in \mathbb{F}_q$
  if $a + b = t$ output 1; otherwise output 0

This completes the description of HomMac. To verify correctness of the scheme, suppose $\mathbf{y} = \sum_{i=1}^{m} \alpha_i \mathbf{v}_i$ where $\mathbf{v}_1, \ldots, \mathbf{v}_m$ are the original augmented basis vectors and $t_1, \ldots, t_m$ are their tags. The coordinates $(y_{n+1}, \ldots, y_{n+m})$ of $\mathbf{y}$ are equal to the coefficients $(\alpha_1, \ldots, \alpha_m)$. Therefore, $a + b$ computed in Verify satisfies

$$a + b = \sum_{i=1}^{m} \alpha_i \cdot \big(\ (\mathbf{u} \cdot \mathbf{v}_i) + F(k_2,\ (\mathsf{id}, i))\ \big) = \sum_{i=1}^{m} \alpha_i \cdot t_i$$

which is precisely the output of $\mathsf{Combine}\big((\mathbf{v}_1, t_1, \alpha_1), \ldots, (\mathbf{v}_m, t_m, \alpha_m)\big)$, as required.

Security. We prove security assuming G is a secure PRG and F is a secure PRF. For a PRF adversary $\mathcal{B}_1$ we let PRF-Adv$[\mathcal{B}_1, F]$ denote $\mathcal{B}_1$'s advantage in winning the PRF security game with respect to F. Similarly, for a PRG adversary $\mathcal{B}_2$ we let PRG-Adv$[\mathcal{B}_2, G]$ be $\mathcal{B}_2$'s advantage in winning the PRG security game with respect to G.

Theorem 2. *For any fixed q, n, m, the MAC scheme HomMac is a secure (q, n, m) homomorphic MAC assuming the PRG G is a secure PRG and the PRF F is a secure PRF.*

In particular, for all homomorphic MAC adversaries $\mathcal{A}$ there is a PRF adversary $\mathcal{B}_1$ and a PRG adversary $\mathcal{B}_2$ (whose running times are about the same as that of $\mathcal{A}$) such that

$$\text{NC-Adv}[\mathcal{A}, \mathit{HomMac}] \leq \text{PRF-Adv}[\mathcal{B}_1, F] + \text{PRG-Adv}[\mathcal{B}_2, G] + (1/q)$$

Proof. We prove the theorem using a sequence of three games denoted Game 0,1,2. For $i = 0, 1, 2$ let W_i be the event that $\mathcal{A}$ wins the homomorphic MAC security game in Game i.

Game 0 is identical to Attack Game 1 applied to the scheme HomMac. Therefore

$$\Pr[W_0] = \text{NC-Adv}[\mathcal{A}, \mathsf{HomMac}] \tag{2}$$

In Game 1 we replace the output of the PRG used in HomMac with a truly random string. That is, Game 1 is identical to Game 0 except that to respond to MAC queries the challenger computes at initialization time $\mathbf{u} \xleftarrow{R} \mathbb{F}_q^{n+m}$ instead of $\mathbf{u} \leftarrow G(k_1)$ in step (1) of the Sign algorithm. Everything else remains the same. Then there is a PRG adversary $\mathcal{B}_2$ such that

$$\big|\Pr[W_0] - \Pr[W_1]\big| = \text{PRG-Adv}[\mathcal{B}_2, G] \tag{3}$$

In Game 2 we replace the PRF by a truly random function. That is, Game 2 is identical to Game 1 except that to respond to MAC queries the challenger computes $b \xleftarrow{R} \mathbb{F}_q$ instead of $b \leftarrow F\big(k_2,\ (\mathsf{id}_i, j)\big)$ in step (2) of the Sign algorithm. Everything else remains the same. Then there is a PRF adversary $\mathcal{B}_1$ such that

$$\big|\Pr[W_1] - \Pr[W_2]\big| = \text{PRF-Adv}[\mathcal{B}_1, F] \tag{4}$$

The complete challenger in Game 2 works as follows:

init: $\mathbf{u} \xleftarrow{R} \mathbb{F}_q^{n+m}$

The adversary submits MAC queries (V_i, id_i) where $V_i = \text{span}(\mathbf{v}_1, \ldots, \mathbf{v}_m) \subseteq \mathbb{F}_q^{n+m}$.
The challenger responds to query number i as follows:

for $j = 1, \ldots, m$ do:

$b_{i,j} \xleftarrow{R} \mathbb{F}_q$ and $t_{i,j} \leftarrow (\mathbf{u} \cdot \mathbf{v}_j) + b_{i,j} \quad \in \mathbb{F}_q$

send $(t_{i,1}, \ldots, t_{i,m})$ to $\mathcal{A}$

Eventually the adversary outputs $(\mathsf{id}^*, t^*, \mathbf{y}^*)$. To determine if the adversary wins the game we first compute:

if $\mathsf{id}^* = \mathsf{id}_i$ then set $(b_1^*, \ldots, b_m^*) \leftarrow (b_{i,1}, \ldots, b_{i,m})$ // (type 2 forgery)
else for $j = 1, \ldots, m$ set $b_j^* \xleftarrow{R} \mathbb{F}_q$ // (type 1 forgery)

Let $\mathbf{y}^* = (y_1^*, \ldots, y_{n+m}^*)$. The adversary wins (i.e. event W_2 happens) if

$$t^* = (\mathbf{u} \cdot \mathbf{y}^*) + \sum_{j=1}^{m} (y_{n+j}^* \cdot b_j^*) \tag{5}$$

and, for a type 2 forgery $\mathbf{y}^* \notin V_i$. Moreover the augmentation $(y_{n+1}^*, \ldots, y_{n+m}^*)$ in $\mathbf{y}^*$ is not all zero.

We now show that $\Pr[W_2] = 1/q$ in Game 2. This is the crux of the proof. Let T be the event that the adversary outputs a type 1 forgery.

Type 1 forgery (event T happens): We bound $\Pr[W_2 \wedge T]$. In a type 1 forgery the right hand side of (5) is a random value in $\mathbb{F}_q$ independent of the adversary's view. Therefore, when event T happens, the probability that (5) holds is exactly $1/q$. Hence, $\Pr[W_2 \wedge T] = (1/q) \cdot \Pr[T]$.

Type 2 forgery (event $\neg T$ happens): We bound $\Pr[W_2 \wedge \neg T]$. In a type 2 forgery $\mathcal{A}$ uses an id^* used in one of the MAC queries. Then $\mathsf{id}^* = \mathsf{id}_i$ for some i. Event W_2 happens if $\mathbf{y}^* \notin V_i$ and (5) holds.

Let $\{t'_1, \ldots, t'_m\}$ be the tags for the basis vectors $\{\mathbf{v}_1, \ldots, \mathbf{v}_m\}$ of the linear space V_i. Define

$$\mathbf{y}' := \sum_{j=1}^{m} y^*_{n+j} \cdot \mathbf{v}_j \in V_i \quad \text{and} \quad t' := \sum_{j=1}^{m} y^*_{n+j} \cdot t'_j \in \mathbb{F}_q$$

Then, t' is a valid tag for $\mathbf{y}'$. Hence, we now know that the following two relations hold:

$$(\mathbf{u} \cdot \mathbf{y}^*) + \sum_{j=1}^{m} y^*_{n+j} \cdot b_{i,j} = t$$
$$(\mathbf{u} \cdot \mathbf{y}') + \sum_{j=1}^{m} y'_{n+j} \cdot b_{i,j} = t'$$

Subtracting one from the other we obtain

$$\big(\mathbf{u} \cdot (\mathbf{y}^* - \mathbf{y}')\big) = t - t' \tag{6}$$

Hence, by producing a valid forgery, the adversary found a $\mathbf{y}^*$ and t that satisfy (6). Moreover, since $\mathbf{y}^* \notin V_i$ but $\mathbf{y}' \in \mathcal{V}_i$ we know that $\mathbf{y}^* \neq \mathbf{y}'$. But since in the adversary's view, $\mathbf{u}$ is indistinguishable from a random vector in $\mathbb{F}_q^{n+m}$, the probability that he can satisfy (6) is exactly $1/q$. Hence, when event $\neg T$ happens we have that $\Pr[W_2 \wedge \neg T] = (1/q) \cdot \Pr[\neg T]$.

Putting together our bounds for $\Pr[W_2 \wedge T]$ and $\Pr[W_2 \wedge \neg T]$ we obtain

$$\Pr[W_2] = \Pr[W_2 \wedge T] + \Pr[W_2 \wedge \neg T] = 1/q(\Pr[T] + \Pr[\neg T]) = 1/q \tag{7}$$

Putting together equations (2),(3),(4),(7) proves the theorem. □

Improved security. Since the tag on a vector $\mathbf{v}$ is a single element in $\mathbb{F}_q$, there is a homomorphic MAC adversary that can break the MAC (i.e. win the MAC security game) with probability $1/q$. When $q = 2^8$ the MAC can be broken with probability 1/256. Security can be improved by computing multiple MACs per vector. For example, with 8 tags per vector security becomes $1/q^8$. The resulting tag is 8 bytes long. The proof of Theorem 2 easily extends to prove these bounds for HomMac using multiple tags.

We note, however, that a homomorphic MAC with security 1/256 may be sufficient for the network coding application. The reason is that the homomorphic MAC is only used by recipients to drop malformed received vectors. The sender can, in addition, compute a regular MAC (such as HMAC) on the transmitted message prior to encoding it using network coding. Recipients, after decoding a matrix of vectors with valid homomorphic MACs, will further validate the HMAC on the decoded message and drop the message if its HMAC is invalid. Hence, success in defeating the homomorphic MAC does not mean that a rogue message is accepted by recipients. It only means that recipients may need to do a little more work to properly decode the message (by trying various m-subsets of the received vectors with a valid homomorphic MAC). As mentioned above, this issue can be avoided by increasing the security of the homomorphic MAC by computing multiple tags per vector.

4 Broadcast Homomorphic MACs: Definitions

We next convert the homomorphic MAC of the previous section to a broadcast homomorphic MAC. This will enable all nodes in the network (both recipients and routers) to verify tags in transmitted packets. We start by defining security for a broadcast homomorphic tag, which takes into account a set of nodes trying to fool some other node.

A broadcast homomorphic MAC is parametrized by a five tuple (q, n, m, μ, c) where (q, n, m) are as in the previous section, μ is the number of nodes in the system, and c is the collusion bound (the maximum number of nodes that can collude to fool another node).

A (q, n, m, μ, c) *broadcast homomorphic MAC* is defined by four probabilistic, polynomial-time algorithms, (Setup, Sign, Verify, Combine) that provide the following functionality:

- $\mathsf{Setup}(\lambda, \mu, c)$: Input: security parameter λ, number of users in the system μ, and desirable collusion resistance bound c. Output: A set of $\mu + 1$ keys $k, k_1, \ldots, k_\mu$. Here k is the sender's key and $k_1, \ldots, k_\mu$ are keys given to the μ verifiers.
- Algorithms Sign, Combine, Verify are as in Section 2, except that the Sign algorithm is given the key k and the Verify algorithm is given one of the keys k_i for some $i \in [\mu]$.

The system must satisfy a correctness requirement analogous to the one in Section 2.

Security: Next, we define security against c-collusions. The adversary $\mathcal{A}$ is given c verifier keys and its goal is to create a message-tag pair that will verify under some verifier's key not in the adversary's possession. More precisely, we define security using the following game.

Attack Game 2. Let $\mathcal{T} = (\mathsf{Setup}, \mathsf{Sign}, \mathsf{Combine}, \mathsf{Verify})$ be a (q, n, m, μ, c) broadcast homomorphic MAC. We define security of $\mathcal{T}$ using the following game between a challenger C and an adversary A (the security parameter λ is given as input to both the challenger and the adversary).

Setup. The adversary sends the challenger the indices of c users acting as verifiers $\{i_1, \ldots, i_c\}$. The challenger runs $\mathsf{Setup}(\lambda, c, \mu)$ to obtain keys $k, k_1, \ldots, k_\mu$ and sends the keys $\{k_{i_1}, \ldots, k_{i_c}\}$ to the adversary.

Queries. The adversary adaptively submits MAC queries as in Attack Game 1. The challenger responds as in that game using the sender's key k.

Output. The adversary $\mathcal{A}$ outputs an index $j^* \in [\mu]$, an identifier id^*, a tag t^*, and a vector $\mathbf{y}^* \in \mathbb{F}_p^{n+m}$.

The adversary *wins* the security game if $\mathsf{Verify}(k_{j^*}, \mathsf{id}^*, \mathbf{y}^*, t^*) = 1$, and the additional winning conditions of Attack Game 1 are satisfied.

The *advantage* BNC-Adv$[\mathcal{A}, \mathcal{T}]$ of $\mathcal{A}$ with respect to $\mathcal{T}$ is defined to be the probability that $\mathcal{A}$ wins this security game.

Definition 3. A (q, n, m, μ, c) broadcast homomorphic MAC scheme $\mathcal{T}$ is secure if for all polynomial time adversaries $\mathcal{A}$, the quantity BNC-Adv$[\mathcal{A}, \mathcal{T}]$ is negligible.

5 Construction 2: A Broadcast Homomorphic MAC

We convert our homomorphic MAC HomMac into a broadcast MAC using a technique of Canetti et al. [3] based on cover free set systems. Instead of computing one tag per vector, we compute several tags per vector using independent keys. We give each verifier a subset of all MAC keys. Thus, each verifier can validate a subset of the MACs on each packet. More importantly, when key assignment is done properly, no coalition of c verifiers can fool another verifier. We start by recalling a few definitions.

Definition 4. A *set system* is a pair $(\mathbb{X}, \mathbb{B})$ where $\mathbb{X}$ is a finite set of elements and $\mathbb{B} = (A_1, \ldots, A_\mu)$ is an ordered set of subsets of $\mathbb{X}$.

Definition 5. A set system $(\mathbb{X}, \mathbb{B})$ is called a (c, d)–*cover free family* if for all c distinct sets $A_1, \ldots, A_c \in \mathbb{B}$ and any other set $A \in \mathbb{B}$, we have $|A \setminus \cup_{j=1}^{c} A_j| > d$.

We construct a (q, n, m, μ, c) broadcast homomorphic MAC from HomMac and any (c, d) cover free family $(\mathbb{X}, \mathbb{B})$ where $|\mathbb{B}| = \mu$. The parameter d is important for security; the error term in the security proof is $(1/q)^d$. The system works as follows:

MAC scheme BrdctHomMac:

Setup(λ, c, μ): Pick a (c, d) cover free family $(\mathbb{X}, \mathbb{B})$,such that $|\mathbb{B}| = \mu$ and $\frac{1}{q^d} < \frac{1}{2^\lambda}$.
Let $\ell = |\mathbb{X}|$ and generate ℓ keys $\{K_1, \ldots, K_\ell\}$ for HomMac. We equate $\mathbb{X}$ with this set of keys, i.e. $\mathbb{X} := \{K_1, \ldots, K_\ell\}$.
The sender's key k consists of all ℓ keys in $\mathbb{X}$. We assign to verifier number i (where $i \in [\mu]$) the key $k_i := A_i \subseteq \mathbb{X}$ where A_i is subset number i in $\mathbb{B}$.

Sign$(\mathbb{X}, \mathsf{id}, \mathbf{v}, i)$. For $j = 1, \ldots, \ell$ compute $t_j \leftarrow$ HomMac-Sign$(K_j, \mathsf{id}, \mathbf{v}, i)$
and output $t := (t_1, \ldots, t_\ell)$.

Combine$((\mathbf{v}_1, t_1, \alpha_1), \ldots, (\mathbf{v}_m, t_m, \alpha_m))$: Apply HomMac-Combine to all ℓ tags in the m tuples.

Verify$(A_i, \mathsf{id}, \mathbf{y}, t)$. Here t is a tuple of ℓ tags. Output 1 if the HomMac tags in t verify for all keys in A_i. Output 0 otherwise.

Security. The following simple theorem states the security property of this construction. Recall that HomMac uses a PRF and a PRG.

Theorem 6. *For any fixed q, n, m, μ, c, the broadcast homomorphic MAC BrdctHomMac is a secure (q, n, m, μ, c) Broadcast Homomorphic MAC assuming the PRG G is a secure PRG and the PRF F is a secure PRF.*

In particular, for all broadcast homomorphic MAC adversaries $\mathcal{A}$ there is a PRF adversary $\mathcal{B}_1$ and a PRG adversary $\mathcal{B}_2$ (whose running times are about the same as that of $\mathcal{A}$) such that

$$\text{BNC-Adv}[\mathcal{A}, \mathit{BrdctHomMac}] \leq \text{PRF-Adv}[\mathcal{B}_1, F] + \text{PRG-Adv}[\mathcal{B}_2, G] + (1/q)^d \tag{8}$$

The proof is immediate from Theorem 2 and is omitted here. Since q is fairly small (e.g. $q = 256$) it is very important that the error term in (8) is $(1/q)^d$. In particular, one needs a (c, d) cover free set system where d makes $(1/q)^d$ negligible (or concretely $(1/q)^d < (1/2)^\lambda$). The $(1/q)^d$ error term is obtained thanks to properties of the HomMac homomorphic MAC.

6 Key Management for Multi-Sender Broadcast Homomorphic MACs

The key dissemination scheme described in the previous section supports a single sender. A real network consists of many nodes where each node can be a sender, a recipient, a router, or all three. To handle many senders with the system of the previous section one would need to set up a cover free family of keys for every sender. In this section, we describe a single cover free set system of keys that simultaneously supports all senders of the network.

We modify the key dissemination scheme as follows. Every node in the network is given two sets of keys, the first set corresponding to the node's role as a sender and the second set corresponding to its role as a router. Hence, the node uses its first set of keys to sign a message, and its second set of keys to verify the authenticity of a message it receives. Every node in the network is identified by a sender id that is unique. We denote the sender id of node i by sid_i. As before, the network is associated with a set of global keys, the set $\mathbb{X}$ of the cover free family. We denote the total number of keys in the network by ℓ, i.e. $|\mathbb{X}| = \ell$ and the keys in the set $\mathbb{X}$ by $x_1, x_2, \ldots, x_l$.

The first set of keys given to node i, called K_{1_i} is $\{F(x_1, \mathsf{sid}_i), F(x_2, \mathsf{sid}_i), \ldots, F(x_l, \mathsf{sid}_i)\}$. Here, F is a PRF. Note that $|K_{1_i}| = \ell$. The second set of keys given to sender i, called K_{2_i} is as before, a block in the cover free family, i.e. $K_{2_i} \in \mathbb{B}$. We denote the members of K_{2_i} by $\{x_{i_1}, x_{i_2}, \ldots, x_{i_b}\}$, where b is the block size.

This setup is sufficient for every node in the network to play its dual roles of sender and router. To sign a packet, node i simply uses its ℓ keys from set K_{1_i} to create ℓ tags for the packet. Verification of a received packet (say p),proceeds as follows:
Note that each packet carries with it the id of its sender. The verifying node reads the sender id of p, say sid_j. Then it computes the b keys it needs to verify p as $\{F(x_{i_1}, \mathsf{sid}_j), F(x_{i_2}, \mathsf{sid}_j), \ldots, F(x_{i_b}, \mathsf{sid}_j)\}$. Using these b keys, the node proceeds to verify p as before. Thus, using sender id of each received packet, a node *computes* 'on the fly' the keys it needs for verification.

The proof of security largely remains the same as in the previous section. As before, knowing a single block (or union of c blocks) is not enough to fool any other node in the system. Also, since the only information a node has about keys of other nodes is encoded by the PRF F, security is maintained.

7 Experimental results

We implemented the homomorphic broadcast MAC outlined in section 5 to measure its performance. In our implementation, we chose $q = 256$, i.e. we worked in the field $\mathbb{F}_{2^8}$. For brevity, we will denote this field by $\mathbb{F}$. Our messages were chosen as vectors of length 1024 over the field $\mathbb{F}$, and the network coding coefficients were picked randomly from $\mathbb{F}$. We ran our experiments using two cover free set systems constructed from polynomials [10].

- The first set system is a $(2, 1)$ cover-free set system where $|\mathbb{X}| = 49$ and each subset contains 7 keys. The number of verifiers is $\mu = 7^4 = 2401$.
- The second set system is a $(2, 5)$ cover-free set system where $|\mathbb{X}| = 121$ and each subset contains 11 keys. The number of verifiers is $\mu = 11^4 = 14641$.

In our implementation, we chose $m = 5$, so the sender sends 5 messages, each a 1 kilobyte vector as described above. Each message is signed with 49 (or 121) keys by the sender, hence a router receives 5 messages, with 49(or 121) tags each. The router then linearly combines the tags along

with the messages, to yield an aggregate message with an aggregate tag. We verify that the resultant aggregate tag is a valid MAC for the aggregate message thus constructed.

Since our homomorphic MAC requires fast multiplication in $\mathbb{F}$, we created a multiplication table offline which stores all 2^{16} products of pairs of elements of $\mathbb{F}$. This table speeds up product computation, which is now just a quick table lookup. The addition in the field is implemented as a simple XOR operation. We implemented the pseudorandom function F and the pseudorandom generator G using AES (from OpenSSL).
We timed the following three operations:

1. Signing: Source signs one message.
2. Combine and Verify: Router receives five (message,tag) pairs and computes a random linear combination of the five vectors and their corresponding tags. Then it verifies that the combined tag is valid for the combined message.

The results for both cover free set systems are shown in the following table. Timing units are in microseconds.

Operation timed	Sign	Combine & Verify	tag size(bytes)	Security
$p = 7$	430.3	88.5	49	$(1/2)^8$
$p = 11$	1329.3	161.5	121	$(1/2)^{40}$

These experiments were conducted on a GNU/Linux system with 4 Intel Xeon 3 Ghz processors with symmetric multiprocessing support.

8 Conclusions

We presented a homomorphic MAC suitable for networks using network coding. The homomorphic MAC can be converted to a broadcast homomorphic MAC using cover free set systems. The resulting broadcast MAC is collusion resistant up to a pre-determined collusion bound c. The tag size grows quadratically with c.

Our experimental results show that the MAC performs well as a point-to-point MAC. As a broadcast MAC it performs well for small values of c. It is an interesting question whether a TESLA-type mechanism [12] applied to our homomorphic MAC can be used to give the same functionality as our broadcast MAC, where every intermediate network router can verify the tag.

References

1. R. Ahlswede, N. Cai, S. Li, and R. Yeung. Network information flow. *IEEE Transactions on Information Theory*, 46(4):1204–1216, 2000.
2. D. Boneh, D. Freeman, J. Katz, and B. Waters. Signing a linear subspace: Signature schemes for network coding. In *Proc. of PKC 2009*, 2009.
3. R. Canetti, J. Garay, G. Itkis, D. Micciancio, M. Naor, and B. Pinkas. Multicast security: A taxonomy and some efficient constructions. In *Proc. of INFOCOM '99*, volume 2, pages 708–716, 1999.
4. L. Carter and M. Wegman. Universal classes of hash functions. *Journal of Computer and System Sciences*, 18(2):143–154, 1979.
5. Denis Charles, K Jain, and K Lauter. Signatures for network coding. In *CISS '06*, 2006. To appear in *International Journal of Information and Coding Theory*.

6. C. Gkantsidis and P. Rodriguez. Network coding for large scale content distribution. In *Proc. of IEEE INFOCOM 2005*, pages 2235–2245, 2005.
7. Keesook Han, Tracey Ho, Ralf Koetter, Muriel Medard, and Fang Zhao. On network coding for security. In *Military Communications Conference (Milcom)*, 2007.
8. Ralf Koetter. An algebraic approach to network coding. *IEEE/ACM Transactions on Networking*, 11:782–795, 2003.
9. M. Krohn, M. Freedman, and D. Mazieres. On the-fly verification of rateless erasure codes for efficient content distribution. In *Proc. of IEEE Symposium on Security and Privacy*, pages 226–240, 2004.
10. Ravi Kumar, Sridhar Rajagopalan, and Amit Sahai. Coding constructions for blacklisting problems without computational assumptions. In *Proc. of Crypto '99*, pages 609–623, 1999.
11. Shuo-Yen Robert Li, Raymond W. Yeung, and Ning Cai. Linear network coding. *IEEE Trans. Inform. Theory*, 49(2):371–381, 2003.
12. A. Perrig, R. Canetti, D. Tygar, and D. Song. Efficient authentication and signature of multicast streams over lossy channels. In *Proc. of 2000 IEEE Symposium on Security and Privacy*, 2000.
13. Fang Zhao, Ton Kalker, Muriel Médard, and Keesook Han. Signatures for content distribution with network coding. In *Proc. of International Symposium on Information Theory (ISIT)*, 2007.