DA Draft

by **XingyuZhou** @ *TU Dresden* im *SS23*

Grundlagen

Rank and Linear Dependence

> also check Linear Algebra

The presence of any zero row or column will produce a **zero determinant**.

> this is easy to verify: just use the Laplace expansion along that zero row or column.

In the (2×2) case we obtained a zero determinant when one row was a multiple of the other (or is the same as the other).

- And this remains true of the (3×3) case. A more subtle case is when one row is a linear combination of the other two, for instance

$$\mathbf{A} = \begin{bmatrix} \mathbf{a_1} \\ \mathbf{a_2} \\ \mathbf{a_3} \end{bmatrix} = \begin{bmatrix} 1 & 2 & 3 \\ 4 & 2 & 4 \\ 3 & 3 & 5 \end{bmatrix}, \qquad \mathbf{a_3} = \mathbf{a_1} + \frac{1}{2}\mathbf{a_2} \rightsquigarrow \det(\mathbf{A}) = 0$$

  We say that the three row vectors that make up the matrix are “linearly dependent”. This “linear dependence” between the rows leads to a zero determinant because it means that row operations can be applied to ultimately create a zero row.
- A (3×3) matrix will have a non-zero determinant **only if** it has three linearly independent rows (or columns). More generally, We say that three (column or row) vectors $\mathbf{a_1}$, $\mathbf{a_2}$, and $\mathbf{a_3}$ are linearly dependent if there are constants c_1, c_2, and c_3, not all zero, such that

$$c_1\mathbf{a_1} + c_2\mathbf{a_2} + c_3\mathbf{a_3} = \mathbf{0}$$

  For now, define the **rank** of the matrix is the number of linearly independent row vectors that it contains.

> 可以这样想象线性不相关：在多维空间中，任意一个向量，都不在由其他向量组成的子空间中。如在三维空间中，第三个向量不在由前两个向量确定的平面中。

column rank is **always** equal to row rank.

Span of a Set of Vectors

Def.: An **augmented matrix** for a system of equations is a matrix of numbers in which each row represents the constants from one equation (both the coefficients and the constant on the other side of the equal sign) and each column represents all the coefficients for a single variable. E.g.:

$$\begin{aligned} x - 2y + 3z &= 7 \\ 2x + y + z &= 4 \\ -3x + 2y - 2z &= -10 \end{aligned} \xrightarrow{\text{augmented matrix}} \left[\begin{array}{ccc|c} 1 & -2 & 3 & 7 \\ 2 & 1 & 1 & 4 \\ -3 & 2 & -2 & 10 \end{array}\right]$$

- we can use augmented matrices and row operations to solve the systems of equations:
  1. write down the augmented matrix for this system
  2. use elementary row operations to convert it into the following form of augmented matrix:

$$\left[\begin{array}{ccc|c} 1 & 0 & 0 & p \\ 0 & 1 & 0 & q \\ 0 & 0 & 1 & r \end{array}\right] \implies \begin{aligned} x &= p \\ y &= q \\ z &= r \end{aligned}$$

Def.: The **span** of a set $\mathcal{S}$ of vectors, denoted $\text{span}(\mathcal{S})$ is the set of all linear combinations of those vectors.

- we can say that the span of the two vectors is the xy-plane (for example), but we also say that the two vectors span the xy-plane. That is, the word span is used as either a noun or a verb, depending on how it is used.

Bsp.: is $\mathbf{v} = [3, -2, -4, 1]^\intercal$ in the span of $\mathcal{S} = \{\, [1, 2, 3, 4]^\intercal, [1, -1, 1, -1]^\intercal, [2, 0, -3, 1]^\intercal \,\}$?

- The question is, “can we find scalars c_1, c_2 and c_3 such that

$$c_1\begin{bmatrix}1\\2\\3\\4\end{bmatrix}+c_2\begin{bmatrix}1\\-1\\1\\-1\end{bmatrix}+c_3\begin{bmatrix}2\\0\\-3\\1\end{bmatrix}\overset{?}{=}\begin{bmatrix}3\\-2\\-4\\1\end{bmatrix}$$

$$\implies \begin{pmatrix}c_1 & c_2 & c_3\end{pmatrix}\cdot\begin{bmatrix}1&2&3&4\\1&-1&1&-1\\2&0&-3&1\end{bmatrix}\overset{?}{=}\begin{pmatrix}3&-2&-4&1\end{pmatrix}$$

- augmented matrix tells us that the system above has no solution, so there are no scalars c_1, c_2 and c_3 for which equation holds. Thus $\mathbf{v}$ is not in the span of $\mathcal{S}$.

Orthogonal vectors and subspaces

Orthogonal (正交) is just another word for *perpendicular* (垂直). Two vectors are orthogonal if the angle between them is 90 degrees. Thus, we can use the Pythagorean theorem to prove that:

$$\mathbf{x}\cdot\mathbf{y}^\intercal=\mathbf{x}^\intercal\cdot\mathbf{y}=0 \quad\iff\quad \mathbf{x}\perp\mathbf{y}$$

Subspace S is orthogonal to subspace T means: every vector in S is orthogonal to every vector in T:

$$S\perp T \quad\iff\quad \forall\mathbf{s}\in S\perp\forall\mathbf{t}\in T$$

- In the plane, the space containing only the zero vector and any line through the origin are orthogonal subspaces. A line through the origin and the whole plane are never orthogonal subspaces. Two lines through the origin are orthogonal subspaces if they meet at right angles.

Permutation and Combination

$$\begin{aligned}P_n^m&=\frac{n!}{(n-m)!}=A_n^m\\C_n^m&=\frac{n!}{m!(n-m)!}\end{aligned}\qquad\text{z.B.}\quad\begin{aligned}P_3^2&=3\times2=6\\C_3^2&=\frac{3\times2}{2\times1}=3\end{aligned}$$

Complexity

Time Complexity: The time complexity of an algorithm quantifies the amount of time taken by an algorithm to run as a function of the length of the input. Note that the time to run is a function of the **length of the input** and not the actual execution time of the machine on which the algorithm is running on.

In order to calculate time complexity on an algorithm, it is assumed that a **constant time** c is taken to execute one operation, and then the total operations for an input length on N are calculated.

Big-O Analysis of Algorithms - Def.: Let g and f be functions from the set of natural numbers to itself. The function f is said to be $\mathrm{O}(g)$ (read big-oh of g), if there is a constant $c>0$ and a natural number n_0 such that $f(n)\le c\cdot g(n)$ for all $n\ge n_0$.

- $\mathrm{O}(g)$ is a set!
- The Big-O Asymptotic Notation gives us the *Upper Bound* Idea, mathematically described: $f(n)=\mathrm{O}(g(n))$ if there exists a positive integer n_0 and a positive constant c, such that $\forall n\ge n_0:\ f(n)\le c\cdot g(n)$
  - $f=\mathrm{O}(g)$ does not mean $f\in\mathrm{O}(g)$

The general step wise procedure for Big-O runtime analysis is as follows:

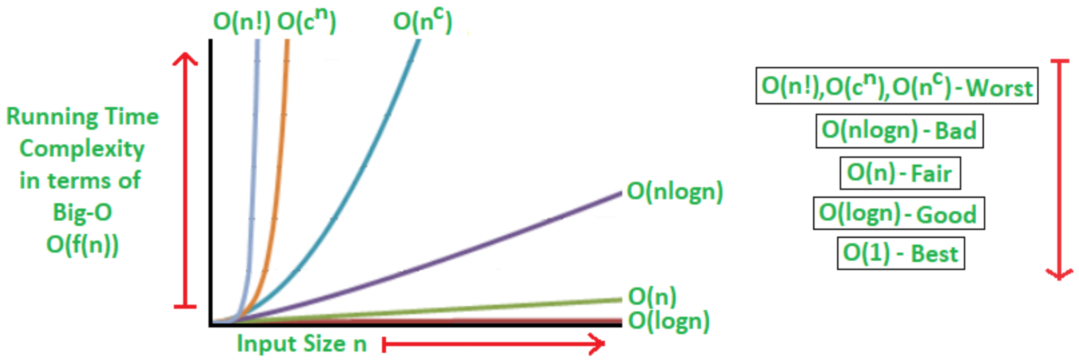
1. Figure out what the input is and what n represents.
2. Express the maximum number of operations, the algorithm performs in terms of n.
3. Eliminate all excluding the highest order terms.
4. Remove all the constant factors.

Some of the useful properties of Big-O notation analysis:

1. Constant Multiplication:
   If $f(n)=c\cdot g(n)$, then $\mathrm{O}(f(n))=\mathrm{O}(g(n))$; where c is a nonzero constant.
2. Polynomial Function:
   If $f(n)=a_0+a_1\cdot n+a_2\cdot n^2+\ldots+a_m\cdot n^m$, then $\mathrm{O}(f(n))=\mathrm{O}(n^m)$.
3. Summation Function:
   If $f(n)=f_1(n)+f_2(n)+\ldots+f_m(n)$ and $f_i(n)\le f_{i+1}(n)\ \forall i=1,2\ldots,m$, then $\mathrm{O}(f(n))=\mathrm{O}(\max(f_1(n),f_2(n),\ldots,f_m(n)))$.
4. Logarithmic Function:
   If $f(n)=\log_a n$ and $g(n)=\log_b n$, then $\mathrm{O}(f(n))=\mathrm{O}(g(n))$; all log functions grow in the same manner in terms of Big-O.

In general cases, we mainly used to measure and compare the worst-case theoretical running time complexities of algorithms for the performance analysis:

- The fastest possible running time for any algorithm is $\mathrm{O}(1)$, commonly referred to as **Constant Running Time**. In this case, the algorithm always takes the same amount of time to execute, regardless of the input size. This is the ideal runtime for an algorithm, but it's rarely achievable.
- The algorithms can be classified as follows from the best-to-worst performance (Running Time Complexity):



1. A logarithmic algorithm – $\mathrm{O}(\log n)$
   Runtime grows logarithmically in proportion to n. z.B. binary search
2. A linear algorithm – $\mathrm{O}(n)$
   Runtime grows directly in proportion to n. z.B. linear search
3. A superlinear algorithm – $\mathrm{O}(n \log n)$
   Runtime grows in proportion to n. z.B. heap sort, merge sort
4. A polynomial algorithm – $\mathrm{O}(n^c)$
   Runtime grows quicker than previous all based on n. z.B. bubble sort, insertion sort, selection sort, bucket sort
5. A exponential algorithm – $\mathrm{O}(c^n)$
   Runtime grows even faster than polynomial algorithm based on n. z.B. tower of Hanoi
6. A factorial algorithm – $\mathrm{O}(n!)$
   Runtime grows the fastest and becomes quickly unusable for even small values of n. z.B. traveling salesman problem

Exp.:

```
def findPair(a, n, z) :
    for i in range(n) :
        for j in range(n) :
            if (i != j and a[i] + a[j] == z) :
                return True
    return False

a = [ 1, -2, 1, 0, 5 ]
z = 0
n = len(a)

if (findPair(a, n, z)) :
    print("True")
else :
    print("False")
```

- In the worst case, $n \cdot c$ operations are required for input. The outer loop `i` loop runs n times. For each `i`, the inner loop `j` loop runs n times. So total execution time is $n \cdot c + n \cdot n \cdot c + c$.
- Now ignore the lower order terms since the lower order terms are relatively insignificant for large input, therefore only the highest order term is taken (without constant) which is $n \cdot n$ in this case.
- Different notations are used to describe the limiting behavior of a function, but since the worst case is taken so big-O notation will be used to represent the time complexity. Hence, the time complexity is $\mathrm{O}(n^2)$ for the above algorithm.

1999 - Multicast Security: A Taxonomy and Some Efficient Constructions

> by Ran Canetti

- a serious bottleneck in multicast security: **source** and **message** authentication

- other works attempts: sharing a **single key** among the multicast group members
  - adequate for encrypting messages so that only group members can decrypt.
  - inadequate for source authentication, since a key shared among all members cannot be used to differentiate among senders in the group.
  - In fact, the only known solutions for multicast authentication involve heavy use of public key signatures

- We present solutions to the source authentication problem based on shared key mechanisms (MACs)
  - Our main savings are in the time to generate signatures

⭘ we say that a MAC scheme is **q-per-message unforgeable** if no (probabilistic polynomial-time) adversary has a positive expected payoff in the following guessing game:

$$\begin{tikzcd} \boxed{\text{adversary}} \arrow[d, "\text{if gamble}"] \arrow[r, shift left, "\text{query for } m_1,\dots,m_k"] & \boxed{\text{challenger}} \arrow[l, shift left, "\text{MACs}"] \\ \text{guess MACs for a } m \notin \{ m_1, \dots, m_k \} & \end{tikzcd}$$

$$\left.\begin{array}{ll} \text{guess correct} & \text{get } \$(1-q) \\ \text{otherwise} & \text{pay } \$q \end{array}\right\} \rightsquigarrow \begin{aligned} \Pr[\text{payoff} > 0] &= (1-q)\cdot\Pr[\text{win}] - q\cdot\Pr[\text{lose}] \le 0 \\ &= (1-q)\cdot\Pr[\text{win}] - q\cdot(1-\Pr[\text{win}]) \le 0 \\ \implies \Pr[\text{win}] \le q \end{aligned}$$

⭘ The Basic Authentication Scheme for a Single Source

- Setup:

$$\begin{tikzcd} \mathcal{R} = \{ K_1, \dots, K_\ell \} \arrow[d] & \mathcal{R}_u \subset \mathcal{R} \arrow[d] \\ \boxed{\text{source } S} & \boxed{\text{recipient } u} \end{tikzcd}$$

$$\ell := e(w+1)\ln\frac{1}{q}$$

$$\forall i = 1, \dots, \ell: \quad \Pr[K_i \in \mathcal{R}_u] := \frac{1}{w+1}$$

$$\implies \qquad |\mathcal{R}_u| \overset{\text{expect}}{=} \ell \cdot \Pr[K_i \in \mathcal{R}_u] = e \ln\frac{1}{q}$$

- MAC:

$$\forall i = 1, \dots, \ell: \quad \text{mac}(K_i, M)$$

⭘ Theorem 1: Assume that the probability of computing the output of a MAC without knowing the key is at most q'. Let u be a user. Then the probability that a coalition of w corrupt users can authenticate a message M to u is at most $q + q'$.

- A similar result holds with respect to per-message unforgability. That is, if the MAC is q'-per-message unforgeable then for any user u and coalition of other w corrupt users, it holds with probability $1-q$ that the resulting scheme is q'-per-message unforgeable with respect to the coalition and the user.
- Proof:

$$\begin{aligned} \Pr[\text{a key is good}] &= g = \frac{1}{w+1}\cdot(1-\frac{1}{w+1})^w > \frac{1}{e(w+1)} \\ \Pr[\text{all keys are bad}] &= (1-g)^\ell < (1-\frac{1}{e(w+1)})^\ell = (1-\frac{1}{e(w+1)})^{e(w+1)\ln\frac{1}{q}} \overset{\clubsuit}{<} e^{-\ln\frac{1}{q}} = q \\ \Pr[\text{at least one key is good}] &= 1 - \Pr[\text{all keys are bad}] > 1-q \end{aligned}$$

$$\begin{aligned} \forall x \in \mathbb{R}: &\quad (1+x) \le e^x \\ \forall x \in (-1,0) \implies &\quad (1+x)^{-\frac{1}{x}} \le (e^x)^{-\frac{1}{x}} \\ \text{let } x = -\tfrac{1}{e(w+1)} \rightsquigarrow &\quad (1-\frac{1}{e(w+1)})^{e(w+1)} < e^{-1} \\ \ln\frac{1}{q} \in (1,\infty) \implies &\quad (1-\frac{1}{e(w+1)})^{e(w+1)\cdot\ln\frac{1}{q}} < e^{-\ln\frac{1}{q}} \end{aligned} \tag{$\clubsuit$}$$

  - a specific key is **good**: contained in a user's subset, but not in the subset of any of the w members of the coalition.
  - By *union bound*

$$\Pr[\text{authenticate } M \text{ to } u] = \Pr[\underbrace{\mathcal{R}_u \text{ is not covered}}_{q'} \vee \underbrace{\mathcal{R}_u \text{ is covered}}_{q}] \le q' + q$$

- The security is against an arbitrary, but fixed, coalition of up to w corrupt recipients. Notice that it is possible to construct schemes which are secure against any coalition of size w by letting

$$q = \frac{1}{n\cdot\binom{n}{w}}$$

- 1 over the number of possible combinations of coalitions and users
- By a probabilistic argument, there exists a system for n recipients in which the subset of **no** user is covered by the union of the subsets of a coalition of size w. The system has a total of less than $e(w+1)^2 \ln n$ keys, and each recipient has a subset of expected size less than $e(w+1) \ln n$.

> The **union bound** is often used to bound the probability that at least one of several events will occur. It provides an upper bound on the probability of the union of multiple events, which can be expressed as:
>
> $$\Pr[A \vee B] \leq \Pr[A] + \Pr[B]$$

2009 - Homomorphic MACs: MAC-based Integrity for Network Coding

> by Shweta Agrawal

◢ augmented vectors $\rightsquigarrow$ the linear combination coefficients are contained in the last m coordinates of $\mathbf{y}$:

$$\mathbf{y} = (\ldots, \underbrace{y_{n+1}, \ldots, y_{n+m}}_{=(\alpha_1, \ldots, \alpha_m)}) = \sum_{i=1}^{m} \alpha_i \mathbf{v}_i$$

◢

$$\operatorname{span}(\{\mathbf{v}_1, \ldots, \mathbf{v}_m\}) = V$$
$$\mathbf{v}' \notin V \rightsquigarrow \mathbf{v}' \text{ is invalid}$$

◢ Signature on V:

$$\sigma = (t_1, \ldots, t_m) \overset{\text{decides}}{\rightsquigarrow} \mathbf{v} \overset{?}{\in} V$$

> 🤔 actually it only decides correctly with a probability of $1 - \frac{1}{q}$ (type 2 forgery) in this paper.

◢ homomorphic MAC:

$$\left.\begin{matrix}(\mathbf{v}_1, t_1) \\ (\mathbf{v}_2, t_2)\end{matrix}\right\} \rightsquigarrow \mathbf{y} = \alpha_1 \mathbf{v}_1 + \alpha_2 \mathbf{v}_2 \xrightarrow[\text{property}]{\text{homomorphic}} t \text{ for } \mathbf{y} \rightsquigarrow (\mathbf{y}, t)$$

◢ **HomMac** Scheme:

- Setup:

$$\begin{aligned}
&\text{Secret Keys as pair } \mathbf{k} = (k_1, k_2): && k_1 \in \mathcal{K}_G,\ k_2 \in \mathcal{K}_F \\
&\text{Pseudo Random Generator } G: && \mathcal{K}_G \xrightarrow{G} \mathbb{F}_q^{n+m} \\
&\text{Pseudo Random Function } F: && \mathcal{K}_F \times (\mathcal{I} \times [m]) \xrightarrow{F} \mathbb{F}_q
\end{aligned}$$

$$\begin{tikzcd}
 & \mathbf{k} = (k_1, k_2) \arrow[dl] \arrow[dr] & \\
\boxed{\text{Source}} & & \boxed{\text{Verifier}}
\end{tikzcd}$$

- Sign:

$$\left.\begin{matrix}\mathbf{v} \in \mathbb{F}_q^{n+m} \\ \mathbf{k} = (k_1, k_2) \in \mathcal{K} \\ \text{id} \in \mathcal{I} \\ i \in [m]\end{matrix}\right\} \to \boxed{\text{Sign}} \to \begin{cases}\mathbf{u} = G(k_1) & \in \mathbb{F}_q^{n+m} \\ b = F(k_2, (\text{id}, i)) & \in \mathbb{F}_q \\ t = (\mathbf{u} \cdot \mathbf{v}) + b & \in \mathbb{F}_q\end{cases}$$

$$(\text{id}, \mathbf{v}_i, t_i) \to \boxed{\text{Network}}$$

- Combine:

$$\left.\begin{matrix}(\mathbf{v}_1, t_1, \alpha_1) \\ \vdots \\ (\mathbf{v}_m, t_m, \alpha_m)\end{matrix}\right\} \to \boxed{\text{Combine}} \to \begin{cases}\mathbf{y} = \sum_{i=1}^{m} \alpha_i \cdot \mathbf{v}_i & \in \mathbb{F}_q^{n+m} \\ t = \sum_{i=1}^{m} \alpha_i \cdot t_i & \in \mathbb{F}_q\end{cases}$$

- Verify:

$$\left.\begin{array}{l}\mathbf{k}=(k_1,k_2)\\ \mathbf{y}=(y_1,\ldots,y_{n+m})\in\mathbb{F}_q^{n+m}\\ \text{id}\\ t\\ i\in[m]\end{array}\right\}\rightarrow\boxed{\text{Verify}}\rightarrow\begin{cases}\mathbf{u}=G(k_1) & \in\mathbb{F}_q^{n+m}\\ a=\mathbf{u}\cdot\mathbf{y} & \in\mathbb{F}_q\\ b=\sum_{i=1}^{m}\Big(y_{n+i}\cdot F(k_2,(\text{id},i))\Big) & \in\mathbb{F}_q\\ t\overset{?}{=}a+b & \overset{\text{yes}}{\rightsquigarrow}\text{ pass}\end{cases}$$

$$\begin{aligned}a+b&=\mathbf{u}\cdot\mathbf{y}+\sum_{i=1}^{m}\Big(y_{n+i}\cdot F(k_2,(\text{id},i))\Big)\\&=\mathbf{u}\cdot\mathbf{y}+\sum_{i=1}^{m}\Big(\alpha_i\cdot F(k_2,(\text{id},i))\Big)\\&=\mathbf{u}\cdot\sum_{i=1}^{m}\alpha_i\mathbf{v}_i+\sum_{i=1}^{m}\Big(\alpha_i\cdot F(k_2,(\text{id},i))\Big)\\&=\sum_{i=1}^{m}\Big(\mathbf{u}\cdot\alpha_i\mathbf{v}_i+\alpha_i\cdot F(k_2,(\text{id},i))\Big)\\&=\sum_{i=1}^{m}\alpha_i\cdot\Big(\mathbf{u}\cdot\mathbf{v}_i+F(k_2,(\text{id},i))\Big)\\&=\sum_{i=1}^{m}\alpha_i\cdot t_i\end{aligned}$$

◢ Attack Game 1:

$$\begin{tikzcd}
 & & \mathbf{k}\overset{R}{\leftarrow}\mathcal{K} \arrow[d] \\
\boxed{\text{adversary }\mathcal{A}} \arrow[d] & \begin{array}{c}\xrightarrow{\text{query for }(V_i,\text{id}_i)}\\ \xleftarrow{(t_1,\ldots,t_m)}\\ \vdots\\ \overset{\text{other queries}}{\longleftrightarrow}\end{array} & \boxed{\text{challenger }C} \\
\text{id}^*,\mathbf{y}^*,t^* & &
\end{tikzcd}$$

$$\text{adversary wins}:\quad \text{Verify}(\mathbf{k},\text{id}^*,\mathbf{y}^*,t^*)=1\ \wedge\ \begin{cases}\forall i:\text{id}^*\neq\text{id}_i & \text{type 1 forgery}\\ \exists i:\text{id}^*=\text{id}_i\ \wedge\ \mathbf{y}^*\notin V_i & \text{type 2 forgery}\end{cases}$$

- 🤔 The adversary doesn't know the $\mathbf{k}$.
- 🤔 these queries are for different generations.
  - in other works, most of them only consider type 2 forgery (in only one generation).

> ? what does id do? And why we divide these 2 types of forgery?

◢ **Theorem 2.**: For any fixed q,n,m, the HomMac is a secure 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

$$\text{NC-Adv}[\mathcal{A},\text{HomMac}]\leq\underbrace{\text{PRF-Adv}[\mathcal{B}_1,F]+\text{PRG-Adv}[\mathcal{B}_2,G]}_{\text{considered safe}}+\frac{1}{q}$$

- Proof:
  - Game 0 is identical to *Attack Game 1*. Therefore

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

  - Game 1 is identical to Game 0 except that we replace the output of the PRG with a *truly random* string. Then there is a PRG adversary $\mathcal{B}_2$ such that

$$\mathbf{u}\overset{R}{\leftarrow}\mathbb{F}_q^{n+m}$$
$$|\Pr[W_0]-\Pr[W_1]|=\text{PRG-Adv}[\mathcal{B}_2,G]\tag{3}$$

  - Game 2 is identical to Game 1 except that we replace the PRF by a *truly random function*. Such that

$$\forall j = 1, \dots, m : \quad b_j \xleftarrow{R} \mathbb{F}_q$$

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

- Let T be the event that the adversary outputs a type 1 forgery. Adversary wins Game 2 if

$$t^* = (\mathbf{u} \cdot \mathbf{y}^*) + \sum_{j=1}^{m} (y^*_{n+j} \cdot b^*_j) \ \wedge \ \begin{cases} T \\ \neg T \end{cases} \tag{5}$$

1. 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. Hence

$$\Pr[W_2 \wedge T] = \frac{1}{q} \cdot \Pr[T]$$

2. In a type 2 forgery $\mathcal{A}$ uses an id^* used in one of the MAC queries. Then $\text{id}^* = \text{id}_i$ for some i. Event W_2 happens if $\mathbf{y}^* \notin V_i$ and (5) holds. Let $\{ t'_1, \dots, t'_m \}$ be the valid tags for the basis vectors $\{ \mathbf{v}_1, \dots, \mathbf{v}_m \}$ of the linear space V_i. Define

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

Then, t' should be a valid tag for $\mathbf{y'}$. Hence

$$t' = (\mathbf{u} \cdot \mathbf{y'}) + \sum_{j=1}^{m} (y'_{n+j} \cdot b^*_j) \tag{★}$$

we now know that the two relations $(5), (\bigstar)$ both hold, then

$$\overset{(5)-(\bigstar)}{\Longrightarrow} \mathbf{u} \cdot (\mathbf{y}^* - \mathbf{y'}) = t^* - t' \tag{6}$$

since $\mathbf{y}^* \notin V_i$ but $\mathbf{y}' \in V_i$ we know that $\mathbf{y}^* \neq \mathbf{y}'$. $\mathbf{u}$ is completely random, so the result of the left hand side of (6) is a random value in $\mathbb{F}_q$. Hence

$$\Pr[W_2 \wedge \neg T] = \frac{1}{q} \cdot \Pr[\neg T]$$

Put them together:

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

- the Theorem 2 is proved by combining $(2), (3), (4), (7)$. □

◢ Improved security:

$$\text{1 tag per vector (1 byte)} \ \rightsquigarrow \ \text{NC-Adv}[\mathcal{A}, \text{HomMac}] \leq \frac{1}{q} = \frac{1}{2^8} = \frac{1}{256}$$

$$\text{8 tag per vector (8 byte)} \ \rightsquigarrow \ \text{NC-Adv}[\mathcal{A}, \text{HomMac}] \leq \frac{1}{q^8}$$

◢ HomMac $\longrightarrow$ broadcast homomorphic MAC (BrdctHomMac):

- since we don't want source node to share a secret key with each non-source nodes, who want to do the verification.
- 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.
- We construct a BrdctHomMac from HomMac and any (c, d)**-cover free family** $(\mathbb{X}, \mathbb{B})$.
  - using a technique of Canetti et al. [3] $\rightsquigarrow$ when key assignment is done properly, **no** coalition of c verifiers can fool another verifier.
  - Def. 4.: A set system is a pair $(\mathbb{X}, \mathbb{B})$ where $\mathbb{X}$ is a finite set of elements and $\mathbb{B} = (A_1, \dots, A_\mu)$ is an *ordered set* of subsets of $\mathbb{X}$.
  - Def. 5.: A set system $(\mathbb{X}, \mathbb{B})$ is called a (c, d)–cover free family if for all c *distinct* sets $A_1, \dots, A_c \in \mathbb{B}$ and any other set $A \in \mathbb{B}$, we have $|A \setminus \bigcup_{j=1}^{c} A_j| > d$.

> we use $()$ to represent a ordered set, and $\{ \}$ to represent a unordered set.

◢ **BrdctHomMac** Scheme:

- Setup:

$$\text{pick a } (\mathbb{X}, \mathbb{B}) \rightsquigarrow \begin{cases} |\mathbb{X}| = \ell \\ |\mathbb{B}| = \mu \\ \dfrac{1}{q^d} < \dfrac{1}{2^\lambda} \end{cases}$$

$$\left.\begin{matrix}\lambda\\ \mu\\ c\end{matrix}\right\} \to \boxed{\text{Setup}} \to \begin{cases} \mathbb{X} \leftarrow \{\, K_1,\dots,K_\ell \,\} \\ \mathbb{B} = (\mathcal{A}_1,\dots,\mathcal{A}_\mu) & \mathcal{A}_i \subseteq \mathbb{X} \\ \mathbf{k} := \mathbb{X} \\ \mathbf{k}_i := \mathcal{A}_i & i \in [\mu] \end{cases}$$

$$\begin{tikzcd} \mathbf{k} \arrow[d] & \mathbf{k}_i \arrow[d] \\ \boxed{\text{Source}} & \boxed{\text{Verifier } i} \end{tikzcd}$$

- security parameter λ

> $K_1 = (k_1, k_2)$ in HomMac

- Sign:

$$\left.\begin{matrix}\mathbf{k}\\ \text{id}\\ \mathbf{v}\\ i \in [m]\end{matrix}\right\} \to \boxed{\text{Sign}} \xrightarrow{\forall j=1,\dots,\ell} \boxed{\text{HomMac-Sign}} \xrightarrow{t_j} \mathbf{t} := (t_1,\dots,t_\ell)$$

- Combine: Apply HomMac-Combine to all ℓ tags in the m tuples.
- Verify:

$$\left.\begin{matrix}\mathbf{k}_i\\ \text{id}\\ \mathbf{y}\\ \mathbf{t}\\ i \in [m]\end{matrix}\right\} \to \boxed{\text{Verify}} \xrightarrow{\forall \text{key} \in \mathbf{k}_i} \boxed{\text{HomMac-Verify}} \to \text{pass if all } \ell \text{ tags are valid}$$

◢ Attack Game 2:

$$\begin{tikzcd} \lambda \arrow[d] & & \mathbf{k}, \mathbf{k}_1, \dots, \mathbf{k}_\mu, \lambda \arrow[d] \\ \boxed{\text{adversary } \mathcal{A}} \arrow[d] \arrow[rr, shift left=4, "\text{indices of } c \text{ users}\{\, i_1,\,\dots,\, i_c \,\}"] \arrow[rr, leftarrow, shift left=2, "(\mathbf{k}_{i_1},\,\dots,\,\mathbf{k}_{i_c})"] \arrow[rr, "\text{query for } (V_i,\text{id}_i)"] \arrow[rr, leftarrow, shift right=2, "(t_1,\,\dots,\,t_m)"] \arrow[rr, leftrightarrow, shift right=4, "\vdots \; \text{other queries}"'] & & \boxed{\text{challenger } C} \\ j^* \in [\mu],\ \text{id}^*,\ \mathbf{y}^*,\ t^* & & \end{tikzcd}$$

$$\text{adversary wins :} \quad \text{Verify}(\mathbf{k}_{j^*}, \text{id}^*, \mathbf{y}^*, t^*) = 1 \ \wedge \ \text{the additional winning conditions of Attack Game 1}$$

◢ **Theorem 6.**: For any fixed q, n, m, μ, c, the BrdctHomMac is a secure 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 A) such that

$$\text{BNC-Adv}[\mathcal{A}, \text{BrdctHomMac}] \le \underbrace{\text{PRF-Adv}[\mathcal{B}_1, F] + \text{PRG-Adv}[\mathcal{B}_2, G]}_{\text{considered safe}} + \frac{1}{q^d}$$

- The proof is immediate from Theorem 2 and is omitted here.
  - From Theorem 2 we know that the adversary can forge one tag to pass a verifier by the probability of $\frac{1}{q}$. And the adversary doesn't know the key the verifier owned.
  - here, each good user has at least d more keys than the coalition. The situation for these d tags are the same as Theorem 2. So the adversary can forge the whole tag tuple to pass a good verifier by the probability of $\frac{1}{q^d}$.

◢ Key Management:

$$\begin{array}{ll}\mathcal{K}_{1,i} = \{\, F(x_1, \mathrm{sid}_i), \ldots, F(x_\ell, \mathrm{sid}_i) \,\} & \text{for the role as a sender} \\ \mathcal{K}_{2,i} = \{\, x_{i,1}, \ldots, x_{i,b} \,\} \in \mathbb{B} & \text{for the role as a router}\end{array}$$

$$\downarrow$$

$$\boxed{\text{node } i}$$

- The key dissemination scheme described previously only supports a **single** sender.
- 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.
- We modify the key dissemination scheme as follows:
  - ...(omitted)

◢ Conclusions:

- only data attack is considered, no tag attack
- sign and verify using the same secret key (symmetric authentication)
- every nodes can be sender or router (based on cover free set system)

◢ Performance Evaluation:

- computation overhead:

$$\underline{\text{Verify}}$$

$$\left.\begin{array}{ll}(n+m)+m & \text{times/tag} \\ \ell & \text{tags}\end{array}\right\} \rightsquigarrow \ell(2m+n) \text{ multiplications/packet}$$

> [❓ 10.30] it doesn't tell how to choose ℓ

2011 - Padding for Orthogonality: Efficient Subspace Authentication for Network Coding

> by Peng Zhang

▣ disadvantages of HomMac:

1. larger bandwidth overhead
2. must carefully manage the keys

▣ basic idea of **homomorphic subspace authentication**:

- **the invariance of linear subspace**: although packets undergo rounds of coding processes at forwarders, the linear subspace V spanned by them stays constant.
- We can check the integrity of a packet $\mathbf{w}$ by verifying whether $\mathbf{w} \in V$.
- we can characterize V using a vector $\mathbf{v}$ randomly chosen from its *orthogonal subspace*, and let forwarders check whether $\mathbf{w} \cdot \mathbf{v}^\intercal = 0$.
- two practical problems unsolved:
  1. For different generations, the source should calculate different $\mathbf{v}$'s, and must distribute them prior to transmission
  2. These $\mathbf{v}$'s should also be authenticated

▣ overcome the 2 problems above $\rightsquigarrow$ **padding for orthogonality**:

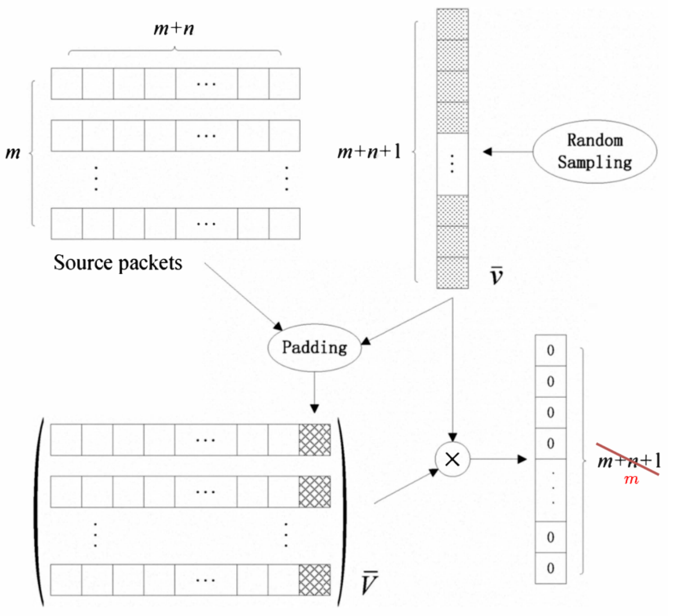


Fig. 1: The idea of "padding for orthogonality"

$$\overline{\mathbf{v}} \xleftarrow{R} \mathbb{F}_q^{m+n+1}$$
$$\overline{V} \perp \overline{\mathbf{v}}$$
$$\mathbf{w} \cdot \overline{\mathbf{v}}^\intercal = \mathbf{0} \implies \mathbf{w} \in \overline{V}$$

- the source pads each packet with an extra symbol, so that the subspace spanned by these padded packets is orthogonal to a specific vector. Forwarders check the integrity of a received packet by verifying whether it maps this vector to zero.
- only one $\overline{\mathbf{v}}$ for all generations need to be distributed using a secure channel.
- pad one more tag cause no startup latency.

▣ make this approach work in presence of Byzantine adversaries (forge tags) $\rightsquigarrow$ 2 solutions:

1. use $\overline{\mathbf{v}}$ as private key and based on it generate a public key which can be used by relay nodes to verify.
   $\rightsquigarrow$ **Homomorphic Subspace Signature** (HSS)
2. let the source keep a pool $\mathcal{P}$ of $\mathbf{v}$'s, and pad each packet with multiple tags generated according to these $\mathbf{v}$'s; each relay node is assigned with a subset of $\mathcal{P}$, and can only verify a packet against part of its tags. If these $\mathbf{v}$'s are distributed properly, a corrupted packet generated by a malicious node will fail the verifications of other nodes with high probability.
   $\rightsquigarrow$ **Homomorphic Subspace MAC** (HSM)

▣ HSS Scheme Construction:

- Setup:

$$\left.\begin{matrix} 1^k \\ N \end{matrix}\right\} \to \boxed{\text{Setup}} \to \begin{cases} q > 2^k & \text{a prime number, the base of } \mathbb{F} \\ \mathbb{G} \text{ with generator } g & |\mathbb{G}| = q \\ \boldsymbol{\beta} \xleftarrow{R} \mathbb{F}_q^N \mathbb{F}_q^* & K_s = \boldsymbol{\beta} \\ \mathbf{h} = g^{\boldsymbol{\beta}} \triangleq (g^{\beta_1}, \ \ldots, \ g^{\beta_{N+1}}) & K_p = \mathbf{h} \end{cases}$$

- Sign:

$$\left.\begin{matrix} \mathbf{x} \in \mathbb{F}_q^N \\ K_s \end{matrix}\right\} \to \boxed{\text{Sign}} \to \begin{cases} \overline{\mathbf{x}} = (\mathbf{x}, \sigma) \\ \sigma = -\dfrac{\sum_{i=1}^N \beta_i \cdot x_i}{\beta_{N+1}} \in \mathbb{F}_q \end{cases} \implies (\mathbf{x} \quad \sigma) \perp \boldsymbol{\beta}$$

- Combine:

$$\left.\begin{matrix} \overline{\mathbf{x}}_1, \ldots, \overline{\mathbf{x}}_\ell \in \mathbb{F}_q^{N+1} \\ \alpha_1, \ldots, \alpha_\ell \in \mathbb{F}_q \end{matrix}\right\} \to \boxed{\text{Combine}} \to \overline{\mathbf{x}} = \sum_{i=1}^{\ell} \alpha_i \overline{\mathbf{x}}_i$$

- Verify:

$$\left.\begin{matrix}\overline{\mathbf{x}} \in \mathbb{F}_q^{N+1} \\ K_p\end{matrix}\right\} \to \boxed{\text{Verify}} \to \delta = K_p^{\overline{\mathbf{x}}} \triangleq \prod_{i=1}^{N+1} h_i^{\overline{x}_i} \overset{?}{=} 1, \quad \overset{\text{yes}}{\rightsquigarrow} \overline{\mathbf{x}} \text{ pass}$$

▣ HSS-GAME:

$$\begin{tikzcd} K_p \arrow[r] & \boxed{\text{adversary } A} \arrow[d] \arrow[rr, "\text{specifiy } k{,}N"] \arrow[rr, "\text{query for } \mathbf{x}_1, \dots, \mathbf{x}_m"'] & & \boxed{\text{challenger } C} \arrow[ll, bend left, "\overline{x}_1, \dots, \overline{x}_m"] \\ & \begin{matrix}\overline{\mathbf{y}} = (\mathbf{y} \in \mathbb{F}_q^N \quad \sigma \in \mathbb{F}_q) \\ \text{with } \mathbf{y} \notin \text{span}(\mathbf{x}_1, \dots, \mathbf{x}_m)\end{matrix} & & \end{tikzcd}$$

$$\text{adversary wins :} \quad \text{Verify}(\overline{\mathbf{y}}, K_p) = 1$$

- An HSS is said to be **secure** if for any PPT adversary A, the probability that A wins the security game HSS-GAME is negligible in the **security parameter** k.
- **Theorem 2.**: Our construction of HSS is secure.

$$\Pr[K_p^{\overline{\mathbf{y}}} = 1] \overset{[21]}{\rightsquigarrow} \Pr[\text{solve the DL-problem over } \mathbb{G}] \geq 1 - \frac{1}{q}$$

$$\Pr[\text{solve the DL-problem over } \mathbb{G}] \overset{[22]}{=} \text{negligible in } k$$

  - equivalent to randomly pick a σ from $\mathbb{F}_q$.

▣ HSM Scheme Construction:

- Setup:

$$\left.\begin{matrix} 1^k \\ N \end{matrix}\right\} \to \boxed{\text{Setup}} \to \begin{cases} q \\ \mathcal{K} = \{\boldsymbol{\gamma}_1, \dots, \boldsymbol{\gamma}_r\} \\ \boldsymbol{\gamma}_i = (\gamma_{i,1}, \dots, \gamma_{i,N+1}) \overset{R}{\leftarrow} \mathbb{F}_q^N \mathbb{F}_q^* \end{cases}$$

- MAC:

$$\left.\begin{matrix} \mathbf{x} \in \mathbb{F}_q^N \\ \mathcal{K} \end{matrix}\right\} \to \boxed{\text{MAC}} \to \begin{cases} \overline{\mathbf{x}} = (\mathbf{x}, T) \\ T = (t_1, \dots, t_r) \in \mathbb{F}_q^r \\ t_i = -\frac{\sum_{j=1}^N \gamma_{i,j} x_j}{\gamma_{i,N+1}} \end{cases} \implies (\mathbf{x} \quad t_i) \perp \boldsymbol{\gamma}_i$$

- Combine:

$$\left.\begin{matrix} \overline{\mathbf{x}}_1, \dots, \overline{\mathbf{x}}_\ell \in \mathbb{F}_q^{N+r} \\ \alpha_1, \dots, \alpha_\ell \in \mathbb{F}_q \end{matrix}\right\} \to \boxed{\text{Combine}} \to \overline{\mathbf{x}} = \sum_{i=1}^{\ell} \alpha_i \overline{\mathbf{x}}_i$$

- Verify:

$$\left.\begin{matrix} \overline{\mathbf{x}} \in \mathbb{F}_q^{N+r} \\ \mathcal{K}' \subset \mathcal{K} \end{matrix}\right\} \to \boxed{\text{Verify}} \to \left\{ \begin{aligned} \forall \boldsymbol{\gamma}_i \in \mathcal{K}' : \qquad \xi_i &= \sum_{j=1}^N \gamma_{i,j} \overline{x}_j + \gamma_{i,N+1} \overline{x}_{N+i} \\ &= (\mathbf{x} \quad \overline{x}_{N+i}) \cdot \boldsymbol{\gamma}_i^\mathsf{T} \\ &= (\mathbf{x} \quad t_i) \cdot \boldsymbol{\gamma}_i^\mathsf{T} \\ &\overset{?}{=} 0, \quad \overset{\text{all yes}}{\rightsquigarrow} \overline{\mathbf{x}} \text{ pass} \end{aligned} \right.$$

> 🤔 how verifier know which $\boldsymbol{\gamma}_i$ to use? $\rightsquigarrow$ E.g.: $\mathcal{K}' = \{\boldsymbol{\gamma}_1, \boldsymbol{\gamma}_4, \boldsymbol{\gamma}_5\}$.

▣ HSM-GAME:

$$\begin{tikzcd}
 & & & \begin{matrix} \mathcal{K}' \overset{R}{\leftarrow} \mathcal{K},\ \mathcal{K}' \subset \mathcal{K} \\ \mathcal{K}'' \overset{R}{\leftarrow} \mathcal{K},\ \mathcal{K}'' \subset \mathcal{K} \end{matrix} \wedge\ |\mathcal{K}'' \setminus \mathcal{K}'| = d \arrow[d] \\
\mathcal{K}' \arrow[r] & \boxed{\text{adversary } A} \arrow[d] \arrow[rr, shift left=4, "\text{specify } k{,}N"] \arrow[rr, "\text{query for } \mathbf{x}_1,\ldots,\mathbf{x}_m"] & & \boxed{\text{challenger } C} \arrow[ll, shift left=4, "{T_1,\ldots,T_m,\ T_i=(t_{i,1},\ldots,t_{i,r})}"] \\
 & \mathbf{y}, T \arrow[d] & & \\
\mathcal{K}'' \arrow[r] & \boxed{\text{Relay}} & &
\end{tikzcd}$$

$$A \text{ wins} := \begin{cases} \mathbf{y} \notin \operatorname{span}(\{\ \mathbf{x}_1,\ \ldots,\ \mathbf{x}_m\ \}) \\ \operatorname{Verify}((\mathbf{y}, T), \mathcal{K}'') = 1 \end{cases}$$

- An HSM is said to be **secure** if for any PPT adversary A, the probability that A wins the security game *HSM-GAME* is $\leq \frac{1}{q^d}$.
- **Theorem 4.**: Our construction of HSM is secure.
  - we consider 3 cases:

$$\forall i = 1, \ldots, r: \qquad \begin{cases} \gamma_i \in \mathcal{K}' & \text{accurately calculates } T & \text{case 1} \\ \gamma_i \notin \mathcal{K}' \ \wedge \ \gamma_i \notin \mathcal{K}'' & \text{any } T \text{ is valid} & \text{case 2} \\ \gamma_i \notin \mathcal{K}' \ \wedge \ \gamma_i \in \mathcal{K}'' & \text{randomly forges } T & \text{case 3} \end{cases}$$

  - adversary can always win in case 1 and case 2

    > 🤔 the key distribution will prevent these 2 cases.

  - in case 3:

$$\overset{\text{after query}}{\Longrightarrow} \begin{bmatrix} \mathbf{x}_1 & t_{1,i} \\ \vdots & \\ \mathbf{x}_m & t_{m,i} \end{bmatrix} \cdot \underbrace{\begin{bmatrix} \gamma_{i,1} \\ \vdots \\ \gamma_{i,N+1} \end{bmatrix}}_{\gamma_i} = \mathbf{0} \tag{4}$$

$$\overset{\text{insert forged one}}{\Longrightarrow} \begin{bmatrix} \mathbf{x}_1 & t_{1,i} \\ \vdots & \\ \mathbf{x}_m & t_{m,i} \\ \mathbf{y} & t_i \end{bmatrix} \cdot \begin{bmatrix} \gamma_{i,1} \\ \vdots \\ \gamma_{i,N+1} \end{bmatrix} = \mathbf{0} \tag{5}$$

    - let R be the *row rank of the coefficient matrix* of (4). And there are $N+1$ unknowns ($\gamma_{i,1}, \ldots, \gamma_{i,N+1}$) $\rightsquigarrow$ there are $N+1-R$ free variables. Each variable is chosen from $\mathbb{F}_q$ uniformly at random, so there are q^{N+1-R} possible solutions.
    - after insertation, $\mathbf{y} \notin \operatorname{span}(\{\ \mathbf{x}_1, \ldots, \mathbf{x}_m\ \})$ $\rightsquigarrow$ row rank of the coefficient matrix will increase by 1. Thus, there are q^{N-R} possible solutions.
    - therefore, $\frac{q^{N-R}}{q^{N+1-R}} = \frac{1}{q}$ solutions from (4) can satisfy (5). Which means, the adversary can randomly forge a t_i to pass the verification with probability $\frac{1}{q}$.
    - since the verifier has at least d safe keys in case 3, the adversary has to forge d tags to pass the verification with probability $\frac{1}{q^d}$.

▣ MAC Key Distribution for HSM

- let SAFE denote the event that there is **at least one safe key** for a randomly chosen w_i and A with $|A| = c$.
- We say the key distribution scheme is ***c*-secure** if for any $w_i \in \Omega$ and $A \subset \Omega$ with $|A| \leq c$, there is at least one safe key.
- probabilistic key distribution mechanism introduced in Canetti et al. [19]:

$$\left.\begin{aligned} |\mathcal{K}| &= e(c+1) \ln \frac{1}{\epsilon} \\ P_a &= \frac{1}{c+1} \end{aligned}\right\} \quad \rightsquigarrow \quad \Pr[\mathrm{SAFE}] \geq 1 - \epsilon$$

  - to achieve this for any w_i and A with $|A| = c \rightsquigarrow |\mathcal{K}| \geq e(c+1)^2 \ln N$
  - Problem: each packet should carry $e(c+1)^2 \ln N$ MACs, which does not scale when the network size N is large.
- **Double-Random Key Distribution** - MAC keys are distributed via two random procedures:
  1. the first procedure assigns each node with a random set of keys, just like in [19]

2. the second one randomly selects ℓ keys from K to be used for MAC calculations $\rightsquigarrow$ rationale! - introduce randomness when generating MACs at the source.

▣ **Theorem 5.**:

$$\left.\begin{aligned} |\mathcal{K}| &:= e(c+1)m = e(c+1)^2 \ln N \cdot \frac{2}{\delta^2}(\gamma+1) \\ m &:= \frac{2}{\delta^2}(\gamma+1)(c+1)\ln N, \qquad\qquad \gamma > 0,\ 0<\delta<1 \\ \ell &:= \frac{1}{1-\delta} e(c+1) \ln\frac{1}{\epsilon} \\ P_a &:= \frac{1}{c+1} \end{aligned}\right\} \rightsquigarrow \Pr[\text{SAFE}] \geq 1-\epsilon,\ \text{when } \gamma\to\infty$$

- Proof:
  For **each** node w_i and **any** set $\mathcal{A}$ of c nodes

$$\Pr[\text{a specific key } k \text{ is safe} \mid \forall w_i, \mathcal{A}] = (1-P_a)^c \cdot P_a > \frac{1}{e(c+1)} \tag{13}$$

Then the *expected number of safe keys* can be derived by

$$\text{E}(\#\text{ of safe keys} \mid \forall w_i, \mathcal{A}) = |\mathcal{K}| \cdot \Pr[\text{a specific key } k \text{ is safe} \mid \forall w_i, \mathcal{A}] > \frac{|\mathcal{K}|}{e(c+1)} = m \tag{14}$$

use *Chernoff bound*

$$\Pr[\#\text{ of safe keys} < (1-\delta)m \mid \forall w_i, \mathcal{A}] < e^{-\frac{\delta^2}{2}m} \tag{15}$$

then

$$\begin{aligned} \Pr[\#\text{ of safe keys} < (1-\delta)m \mid \exists w_i, \mathcal{A}] &= \Pr[\bigcup_{w_i,\mathcal{A}} \#\text{ of safe keys} < (1-\delta)m \mid \forall w_i, \mathcal{A}] \\ &< \sum_{w_i,\mathcal{A}} \Pr[\#\text{ of safe keys} < (1-\delta)m \mid \forall w_i, \mathcal{A}] \\ &< N \cdot \binom{N}{c} \cdot e^{-\frac{\delta^2}{2}m} \\ &< \underbrace{N^{c+1}}_{=e^{\ln N^{c+1}}} \cdot e^{-\frac{\delta^2}{2}m} \\ &= e^{(c+1)\ln N} \cdot e^{-\frac{\delta^2}{2}m} \\ &= e^{(c+1)\ln N - \frac{\delta^2}{2}m} \\ &= e^{(c+1)\ln N - \frac{\delta^2}{2}\cdot\frac{2}{\delta^2}(\gamma+1)(c+1)\ln N} \\ &= e^{(c+1)\ln N - (\gamma+1)(c+1)\ln N} \\ &= e^{-\gamma(c+1)\ln N} \\ &= \frac{1}{N^{\gamma(c+1)}} \qquad\qquad \overset{\gamma\to\infty}{\to} 0 \end{aligned}$$

then

$$\Pr[k \overset{R}{\leftarrow} \mathcal{K} \text{ is safe} \mid \forall w_i, \mathcal{A}] > \frac{(1-\delta)m}{|\mathcal{K}|} = \frac{(1-\delta)m}{e(c+1)m} = \frac{1-\delta}{e(c+1)} \tag{16}$$

the source will randmonly choose ℓ keys from $\mathcal{K}$

$$\Pr[\text{all } \ell \text{ keys are not safe}] = \left(1-\frac{1-\delta}{e(c+1)}\right)^{\ell} = \left(1-\frac{1-\delta}{e(c+1)}\right)^{\frac{e(c+1)}{1-\delta}\ln\frac{1}{\epsilon}} < e^{-\ln\frac{1}{\epsilon}} = \epsilon \tag{17}$$

Thus, the double-random key distribution scheme is c-secure with probability no less than $1-\epsilon$. □

- Theorem 5 shows that the number of MACs per packet (ℓ) has no relation with the N.
- 🤔 [19] use all keys in $\mathcal{K}$ to genearate MACs, but here only use ℓ keys in $\mathcal{K}$ to generate MACs.

> The basic form of the **Chernoff bound** states that for a collection of independent random variables $X_1, \ldots, X_n$, each taking values in $[0,1]$, the probability that the sum of the variables $S = X_1 + \ldots + X_n$ deviates from its expected value $\mathrm{E}(S)$ by a certain amount can be bounded as follows:
>
> $$\begin{aligned} \Pr[S \geq (1+\delta)\,\mathrm{E}(S)] &\leq e^{-\frac{\delta^2}{2}\,\mathrm{E}(S)} \\ \Pr[S \leq (1-\delta)\,\mathrm{E}(S)] &\leq e^{-\frac{\delta^2}{2}\,\mathrm{E}(S)} \end{aligned} \qquad \text{for } 0 < \delta < 1$$

▣ **MacSig** Scheme:

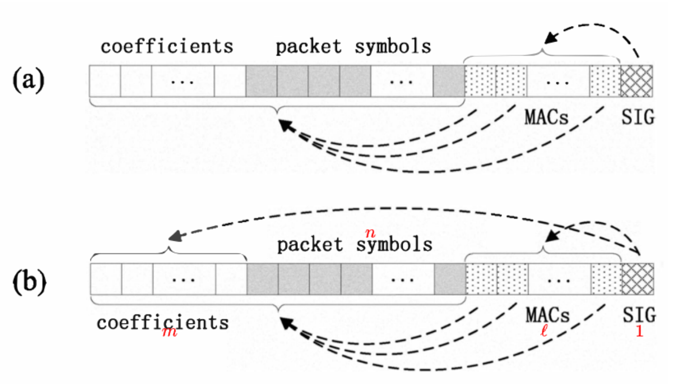


Fig. 3: Basic idea of the MacSig authentication scheme.

- real implementation of MacSig → 3(b)
- Setup
  - HSS Setup + HSM Setup
  - for each MAC key, assign it to each node with an equal probability P_a.
- MAC+Sign:

$$\left.\begin{aligned} \mathbf{x}_i &= (x_{i,1}, \ldots, x_{i,m+n}) \in \mathbb{F}_p^{m+n} \\ \mathcal{K} &= \{\,\boldsymbol{\gamma}_i\,\}_{i=1}^{\ell} \\ \boldsymbol{\gamma}_i &= (\gamma_{i,1}, \ldots, \gamma_{i,m+n+1}) \xleftarrow{R} \mathbb{F}_p^{m+n}\mathbb{F}_p^{*} \\ K_s &= \boldsymbol{\beta} = (\beta_1, \ldots, \beta_{m+\ell+1}) \xleftarrow{R} \mathbb{F}_p^{m+\ell}\mathbb{F}_p^{*} \end{aligned}\right\} \rightarrow \boxed{\text{MAC+Sign}} \rightarrow \left\{\begin{aligned} \overline{\mathbf{x}}_i &= (\mathbf{x}_i, T_i, \sigma_i) \in \mathbb{F}_p^{m+n}\mathbb{F}_p^{\ell}\mathbb{F}_p^{*} \\ T_i &= (t_{i,1}, \ldots, t_{i,\ell}) \in \mathbb{F}_p^{\ell} \\ t_{i,j} &= -\frac{\sum_{r=1}^{m+n}\gamma_{j,r}x_{i,r}}{\gamma_{j,m+n+1}} && (6) \\ \sigma_i &= \frac{-\sum_{j=1}^{m}\beta_j x_{i,j} + \sum_{j=1}^{\ell}\beta_{m+j}t_{i,j}}{\beta_{m+\ell+1}} && (7) \end{aligned}\right.$$

- Combine
- Verify:

$$\left.\begin{aligned} \overline{\mathbf{y}} &= (\overline{y}_1,\ \ldots,\ \overline{y}_{m+n+\ell+1}) \in \mathbb{F}_p^{m+n+\ell+1} \\ K_p &= \mathbf{h} = (h_1,\ \ldots,\ h_{m+\ell+1}) \\ \{\,\overline{\boldsymbol{\gamma}}_1,\ &\overline{\boldsymbol{\gamma}}_2,\ \ldots\} \subset \mathcal{K} \end{aligned}\right\} \boxed{\text{Verify}} \left\{\begin{aligned} \delta &= \prod_{i=1}^{m} h_i^{\overline{y}_i} \cdot \prod_{i=1}^{\ell+1} h_{m+i}^{\overline{y}_{m+n+i}} \overset{?}{=} 1, && \overset{\text{no}}{\rightsquigarrow} \text{reject} && (8) \\ \xi_i &= \sum_{r=1}^{m+n}\gamma_{i,r}\overline{y}_r + \gamma_{i,m+n+1}\overline{y}_{m+n+1}, && \exists \xi_i \neq 1 \rightsquigarrow \text{reject} && (9) \end{aligned}\right.$$

> [❓ 10.20] the verification order here is correct, e.g., verify Sign first then MAC. ⇝ the order does not matter.

▣ **Theorem 6.**: The MacSig authentication scheme is secure against *tag pollution*.

- notations:

$$\begin{aligned} \overline{\mathbf{x}} &= (\mathbf{x} \quad \mathbf{t}),\ \begin{cases} \mathbf{x} = (x_1 \quad \cdots \quad x_m) \\ \mathbf{t} = (t_1 \quad \cdots \quad t_\ell \quad \sigma) \end{cases} \\ \overline{V} &= \operatorname{span}(\{\,\overline{\mathbf{x}}_1, \ldots, \overline{\mathbf{x}}_m\,\}) \\ T &= \operatorname{span}(\{\,\mathbf{t}_1, \ldots, \mathbf{t}_m\,\}) \\ T' &= \operatorname{span}(\{\,(\mathbf{c}_1 \quad \mathbf{t}_1), \ldots, (\mathbf{c}_m \quad \mathbf{t}_m)\,\}) \\ \mathcal{K}_I &= \{\,\boldsymbol{\gamma}_1, \ldots, \boldsymbol{\gamma}_b\},\quad b < m \text{ (assumed)} \end{aligned}$$

$$\begin{tikzcd} \overline{\mathbf{y}} = (\mathbf{y}, \mathbf{t}) \in \overline{V} \arrow[d] \\ \boxed{\text{adversary } A} \arrow[d] \\ \overline{\mathbf{y}}' = (\mathbf{y}, \mathbf{t}' \neq \mathbf{t}) \arrow[d] \\ \boxed{\text{innocent Node } I} \arrow[d] \\ \text{pass the verification } (\overline{\mathbf{y}}' \in \overline{V}) \end{tikzcd}$$

- 3(a) is employed:
  - $\mathbf{t}'$ should satisfy the following two conditions:
    1. $\mathbf{t}' \in T$ (by Theorem 2)
    2. $\forall i = 1, \dots, b: \ t'_i = t_i$ (since message symbols are not modified).
  - let $\overline{\mathbf{y}}_1, \dots, \overline{\mathbf{y}}_m$ be a basis of $\overline{V}$. Then any $\mathbf{t}'$ that satisfy the above two conditions can be represented as

$$\mathbf{t}' = \boldsymbol{\alpha}^{1\times m} \cdot \begin{pmatrix} \mathbf{t}_1^\intercal & \cdots & \mathbf{t}_m^\intercal \end{pmatrix}^\intercal \tag{10}$$

> [❓ 10.20] not very clear why $\mathbf{t}' \in T$ can lead to (10)? $\rightsquigarrow$ check span.

with $\boldsymbol{\alpha}$ subjected to

$$\boldsymbol{\alpha}^{1\times m} \cdot \begin{bmatrix} t_{1,1} & \cdots & t_{1,b} \\ \vdots & \ddots & \vdots \\ t_{m,1} & \cdots & t_{m,b} \end{bmatrix} = \begin{pmatrix} t_1 & \cdots & t_b \end{pmatrix} \tag{11}$$

  - let λ be the column rank of the coefficient matrix $\rightsquigarrow$ there are λ linear independent columns in $[t_{m,b}]$ matrix $\rightsquigarrow$ there are $m-\lambda$ elements in $\boldsymbol{\alpha}$ that can be chosen arbitrarily $\rightsquigarrow$ each element can be chosen from $\mathbb{F}_p$ uniformly at random $\rightsquigarrow$ there will be $p^{m-\lambda}$ possible $\boldsymbol{\alpha}$.
  - each possible $\boldsymbol{\alpha}$ can construct $\mathbf{t}'$ to pass the verification. And one of them will equal to $\mathbf{t}$.
  - so the $\Pr[\text{ tag pollution succeed}] = 1 - \dfrac{1}{p^{m-\lambda}}$ $\rightsquigarrow$ 3(a) is not secure.
- (3b) is employed:
  - conditions:
    1. $\begin{pmatrix} y_1 & \dots & y_m & \mathbf{t}' \end{pmatrix} \in T'$
    2. remains the same
  - As a result, the equations that $\boldsymbol{\alpha}$ is subjected to becomes

$$\boldsymbol{\alpha}^{1\times m} \cdot \begin{bmatrix} y_{1,1} & \cdots & y_{1,m} & t_{1,1} & \cdots & t_{1,b} \\ \vdots & \ddots & \vdots & \vdots & \ddots & \vdots \\ y_{m,1} & \cdots & y_{m,m} & t_{m,1} & \cdots & t_{m,b} \end{bmatrix} = \begin{pmatrix} y_1 & \cdots & y_m & t_1 & \cdots & t_b \end{pmatrix} \tag{12}$$

  - $\overline{\mathbf{y}}_1, \dots, \overline{\mathbf{y}}_m$ are linearly independent $\rightsquigarrow$ $\lambda = m$ $\rightsquigarrow$ there is only one solution for $\boldsymbol{\alpha}$ $\rightsquigarrow$ $\mathbf{t}'$ constructed by using this $\boldsymbol{\alpha}$ is the same as $\mathbf{t}$ $\rightsquigarrow$
  - So $\Pr[\text{ tag pollution succeed}] = 0$ $\rightsquigarrow$ MacSig (3b) is secure against tag pollution.

▣ Performance of MacSig:

- Bandwidth Overhead:
  1. consumed by the key distribution $\rightsquigarrow$ neglect, cause it can be done offline
  2. online bandwidth overhead per packet:

$$\left.\begin{array}{lll} \ell \text{ MACs} & |p| = \lceil \log_2 p \rceil & \text{bits/MAC} \\ \ell \text{ MAC key indices} & \lambda = \lceil \log_2 |\mathcal{K}| \rceil & \text{bits/index} \\ 1 \text{ signature} & |p| & \text{bits/signature} \end{array}\right\} \rightsquigarrow O_b = \frac{\ell+1}{m+n} + \frac{32\ell}{|p|(m+n)}$$

    - $|\mathcal{K}|, \ell$ are used in double-random key distribution.
    - We choose $\lambda = 32$ (byte alignment), $\delta = 0.1, n = 20m, |p| = 128$

> ❓ what is the use of MAC key indexes?

- Computation Overhead
  1. offline computation overhead is also neglected
  2. online computation overhead per packet:

$$\underline{\text{MAC+Sign}}$$

$$\left.\begin{array}{lll} \ell \text{ MACs} & (m+n+1) & \text{times/MAC} \\ 1 \text{ signature} & (m+\ell+1) & \text{times/Sign} \end{array}\right\} \rightsquigarrow (m+n+1)\ell + (m+\ell+1) \text{ times}$$

$$\underline{\text{Combine}}$$

$$w(\ell+1) \text{ times}$$

$$\underline{\text{Verify}}$$

$$\left.\begin{array}{ll} m+(\ell+1) & \text{exponentions/packet} \\ (m+n)+1 & \text{times/key} \end{array}\right\} \rightsquigarrow \frac{3}{2}|p|(m+\ell+1) + (m+n+1)\ell \text{ times/packet}$$

$$\underline{\text{Benchmark}}$$

$$\text{running time of times over } \mathbb{F}_p \overset{\text{Montgomery multiplication algorithm [23]}}{\rightsquigarrow} 2.5 \times 10^5 \text{ times/s}$$

- $1 \text{ exponention over } \mathbb{F}_p \overset{\text{square and multiple}}{\equiv} \frac{3}{2}|p| \text{ muliplications}$

> [❓ 10.20] actually it should not be $(m+n+1)\ell$ times/packet, since a verifier only uses the keys it owns, which is less than ℓ. $\rightsquigarrow$ we consider the "worst case".

▣ three authentication schemes (HSS, HSM, and MacSig) introduced above assume **single generation**.

- If the transmission consists of multiple generations, the adversary can launch *repetitive attack*: collect legitimate packets of previous generations and use them to fake packets for subsequent ones.

2014 - A Tag Encoding Scheme against Pollution Attack to Linear Network Coding

> by Xiaohu Wu

⊙ Introduction:

- The schemes in HomMac [13] are based on key predistribution. The computational complexities are low, but they experience tag pollution attack that leads to numerous correct data packets being discarded.
- The MagSig [15] requires a field of size 128 bits with very high computational complexity. In addition, even if the [13], [15] tolerate less nodes to be compromised, very large redundancy is needed to append to each data packet. For example, if the [15] allows 15 nodes to be compromised with a probability 99.5 percent, the size of redundancy appended to an IP data packet of 1,400 bytes will be 5,144 bytes

⊙ propose a key predistribution-based tag encoding (KEPTE) scheme:

- some insights on mathematical relations in linear network coding
- basic idea:

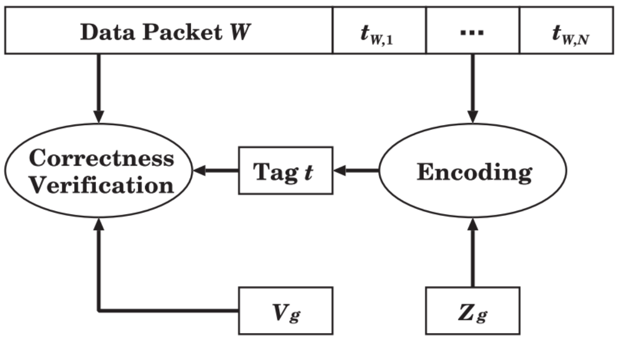


Fig. 3. Verification at a node g.

- advantages:
  1. does not require a large field size
  2. any non-source node can verify (both data and tag pollution)
  3. tolerate more compromised nodes

> data pollution: make nodes unable to detect error data packets
> tag pollution: get correct data packets be judged as wrong and be discarded

⊙ **KEPTE** Scheme:

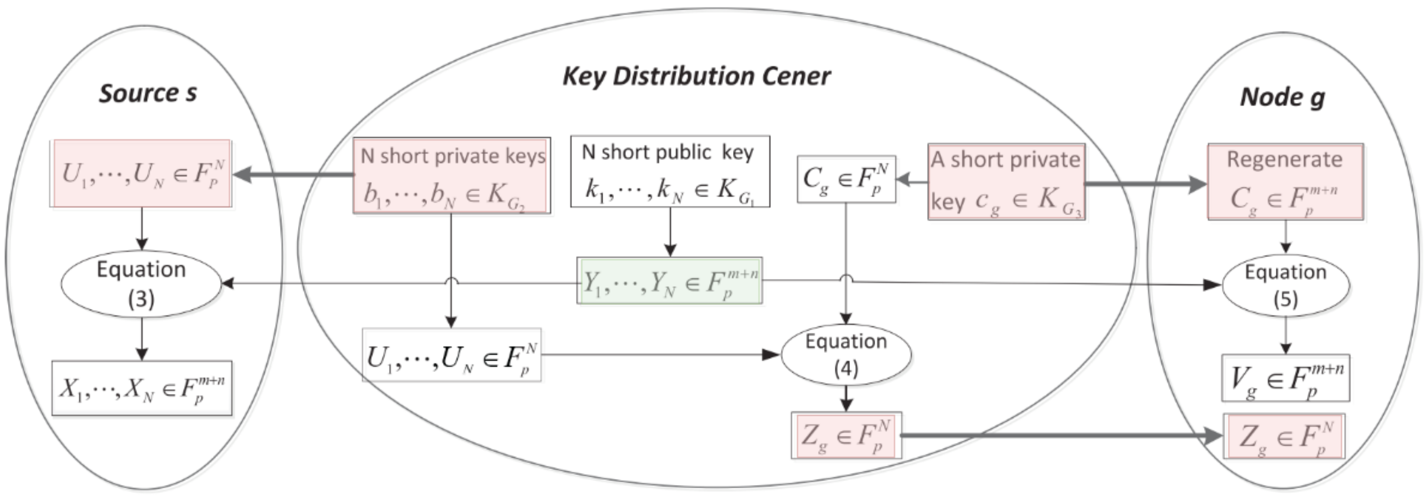
- Sign:

$$\left.\begin{array}{l} X_1,\dots,X_N \in \mathbb{F}_p^{m+n} \\ P_i \in \mathbb{F}_p^{m+n} \end{array}\right\} \rightarrow \boxed{\text{Sign}} \rightarrow \begin{cases} t_{P_i,1},\dots,t_{P_i,N} \in \mathbb{F}_p \\ t_{P_i,j} = P_i \cdot X_j^\intercal \end{cases}$$

  - ⚠️ n: generation size, m: number of message symbols.
- Combine
- Verify

$$\left.\begin{array}{l} Z_g \in \mathbb{F}_p^N \\ V_g \in \mathbb{F}_p^{m+n} \\ W \in \mathbb{F}_p^{m+n} \\ t_{W,1},\dots,t_{W,N} \in \mathbb{F}_p \end{array}\right\} \rightarrow \boxed{\text{Verify}} \rightarrow Z_g \cdot (t_{W,1},\dots,t_{W,N})^\intercal \overset{?}{=} W \cdot V_g^\intercal, \quad \overset{\text{yes}}{\rightsquigarrow} \text{pass}$$

⊙ KEPTE key distribution:



Source s
Key Distribution Cener
Node g
$U_1,\cdots,U_N \in F_p^N$
N short private keys $b_1,\cdots,b_N \in K_{G_2}$
N short public key $k_1,\cdots,k_N \in K_{G_1}$
$C_g \in F_p^N$
A short private key $c_g \in K_{G_3}$
Regenerate $C_g \in F_p^{m+n}$
Equation (3)
$Y_1,\cdots,Y_N \in F_p^{m+n}$
Equation (5)
$U_1,\cdots,U_N \in F_p^N$
Equation (4)
$X_1,\cdots,X_N \in F_p^{m+n}$
$V_g \in F_p^{m+n}$
$Z_g \in F_p^N$
$Z_g \in F_p^N$

$$
\begin{aligned}
G_1 : \mathcal{K}_{G_1} \to \mathbb{F}_p^{m+n} \quad & Y_i = G_1(k_i), \quad k_1,\ldots,k_N \in \mathcal{K}_{G_1} \\
G_2 : \mathcal{K}_{G_2} \to \mathbb{F}_p^{N} \quad & U_i = G_2(b_i), \quad b_1,\ldots,b_N \in \mathcal{K}_{G_2} \\
G_3 : \mathcal{K}_{G_3} \to \mathbb{F}_p^{N} \quad & C_g = G_3(c_g), \quad c_g \in \mathcal{K}_{G_3}
\end{aligned}
$$

$$V_g = Z_g \cdot \begin{bmatrix} X_1 \\ \vdots \\ X_N \end{bmatrix} \tag{2}$$

$$\begin{bmatrix} U_1 \\ \vdots \\ U_N \end{bmatrix} \cdot \begin{bmatrix} X_1 \\ \vdots \\ X_N \end{bmatrix} = \begin{bmatrix} Y_1 \\ \vdots \\ Y_N \end{bmatrix} \implies \begin{bmatrix} X_1 \\ \vdots \\ X_N \end{bmatrix} \tag{3}$$

$$Z_g = C_g \cdot \begin{bmatrix} U_1 \\ \vdots \\ U_N \end{bmatrix} \tag{4}$$

$$V_g = C_g \cdot \begin{bmatrix} Y_1 \\ \vdots \\ Y_N \end{bmatrix} \tag{5}$$

⊙ Security analyse of data pollution attack:

- Useful information to launch attacks:
  - known to the adversary:
    1. $P_1,\ldots,P_n$
    2. $C_{g_i}, Z_{g_i}, V_{g_i},\ 1 \le i \le r,\ r < N$
  - According to the algorithm Sign, the adversary can get (9) as an estimation of $X_1,\ldots,X_N$.

$$\forall j = 1,\ldots,N : \quad \begin{bmatrix} P_1 \\ \vdots \\ P_n \end{bmatrix} \cdot X_j^\intercal = \begin{bmatrix} t_{P_1,j} \\ \vdots \\ t_{P_n,j} \end{bmatrix} \tag{9}$$

  - According to the key distribution, the adversary can get (10), (11) as an estimation of $X_1,\ldots,X_N$ or $U_1,\ldots,U_N$.

$$\overset{(2)}{\Longrightarrow} \begin{bmatrix} Z_{g_1} \\ \vdots \\ Z_{g_r} \end{bmatrix} \cdot \begin{bmatrix} X_1 \\ \vdots \\ X_N \end{bmatrix} = \begin{bmatrix} z_{g_1,1} & \cdots & z_{g_1,N} \\ \vdots & \ddots & \vdots \\ z_{g_r,1} & \cdots & z_{g_r,N} \end{bmatrix} \cdot \begin{bmatrix} x_{1,1} & \cdots & x_{1,m+n} \\ \vdots & \ddots & \vdots \\ x_{N,1} & \cdots & x_{N,m+n} \end{bmatrix} = \begin{bmatrix} v_{g_1,1} & \cdots & v_{g_1,m+n} \\ \vdots & \ddots & \vdots \\ v_{g_r,1} & \cdots & v_{g_r,m+n} \end{bmatrix} \tag{10}$$

$$\overset{(4)}{\Longrightarrow} \begin{bmatrix} C_{g_1} \\ \vdots \\ C_{g_r} \end{bmatrix} \cdot \begin{bmatrix} U_1 \\ \vdots \\ U_N \end{bmatrix} = \begin{bmatrix} c_{g_1,1} & \cdots & c_{g_1,N} \\ \vdots & \ddots & \vdots \\ c_{g_r,1} & \cdots & c_{g_r,N} \end{bmatrix} \cdot \begin{bmatrix} u_{1,1} & \cdots & u_{1,N} \\ \vdots & \ddots & \vdots \\ u_{N,1} & \cdots & u_{N,N} \end{bmatrix} = \begin{bmatrix} Z_{g_1} \\ \vdots \\ Z_{g_r} \end{bmatrix} \tag{11}$$

1. The First Pollution Attack Behavior - Searching for $X_1,\ldots,X_N$:
   - The First Way - $X \overset{(3)}{\leftarrow} U$

$$\overset{(11)}{\Longrightarrow} \quad \underbrace{\forall i = 1, \ldots, N}_{N \text{ systems of linear equations}} \quad : \quad \begin{bmatrix} C_{g_1} \\ \vdots \\ C_{g_r} \end{bmatrix} \cdot \underbrace{\begin{bmatrix} u_{1,i} \\ \vdots \\ u_{N,i} \end{bmatrix}}_{N \text{ unknows}} = \begin{bmatrix} z_{g_1,i} \\ \vdots \\ z_{g_r,i} \end{bmatrix} \tag{13}$$

- **Theorem 1.**: let $\mathcal{U}$ be the set of all the solutions of $U_1, \ldots, U_N$ in (11). If the adversary randomly selects an element from $\mathcal{U}$, the probability that the element is exactly the N secret vectors $U_1, \ldots, U_N$ held by s is $\le \frac{1}{p^{N(N-r)}}$, where p is the size of field $\mathbb{F}_p$.
  - Proof:
    - the rank of $\left[C_{g_i}\right]$ in (13) is $\le r \rightsquigarrow$ at least p^{N-r} possible solutions.
    - N systems of linear equations $\rightsquigarrow |\mathcal{U}| \ge p^{N(N-r)}$
    - Hence, if the adversary randomly selects an element from $\mathcal{U}$, the probability that the element is exactly the N secret vectors $U_1, \ldots, U_N$ is no greater than $\frac{1}{p^{N(N-r)}}$. $\square$
- The Second Way - $X \leftarrow P, t$
  - **Theorem 2.**: let $\mathcal{X}$ be the set of all the solutions of $X_1, \ldots, X_N$ that satisfy both (9) and (10). If the adversary randomly selects an element from $\mathcal{X}$, the probability that the element is exactly the N secret vectors $X_1, \ldots, X_N$ held by s is $\le \frac{1}{p^{m(N-r)}}$.
    - Proof:

    > ❓ not clear

2. The Second Pollution Attack Behavior - Finding a *Deceptive Data Packet*:
   - According to the algorithm Verify, the following is useful information for the adversary to find out a deceptive data packet $W' \notin \operatorname{span}(\{ P_1, \ldots, P_n \})$:

$$\begin{bmatrix} Z_{g_1} \\ \vdots \\ Z_{g_r} \end{bmatrix} \cdot \begin{bmatrix} t_{W',1} \\ \vdots \\ t_{W',N} \end{bmatrix} = \begin{bmatrix} V_{g_1} \\ \vdots \\ V_{g_r} \end{bmatrix} \cdot W'^{\intercal} \tag{12}$$

   - The First Way: The adversary randomly selects $t_{W',1}, \ldots, t_{W',N} \in \mathbb{F}_p$. The adversary randomly selects an element from $\mathcal{W}_1$ and hopes that the element belongs to $\mathcal{W}_2$.
     - let $\mathcal{W}_2$ be the set of all the deceptive data packets with $t_{W',1}, \ldots, t_{W',N}$ as their tags.
     - Let $\mathcal{W}_1$ be the set of all W' s which not only are solutions of (12) but also satisfy $W' \notin \operatorname{span}(\{ P_1, \ldots, P_n \})$, but W' may not pass the correctness check at some node(s).
     - $\mathcal{W}_2 \subseteq \mathcal{W}_1$

     > 🤔 $W' \in \mathcal{W}_1$ is only guaranteed to pass the verification of the comprised nodes. $W' \in \mathcal{W}_2$ can pass verification of all nodes.

   - **Theorem 3.**: the probability of W' belonging to $\mathcal{W}_2$ is $\le \frac{1}{p^{N-r}}$.
     - Proof:
       - A deceptive data packet W' with given N tags satisfies

$$C_g \cdot \begin{bmatrix} U_1 \\ \vdots \\ U_N \end{bmatrix} \cdot \begin{bmatrix} t_{W',1} \\ \vdots \\ t_{W',N} \end{bmatrix} = C_g \cdot \begin{bmatrix} Y_1 \\ \vdots \\ Y_N \end{bmatrix} \cdot W'^{\intercal}$$
$$\Longrightarrow \begin{bmatrix} t_{W',1} \\ \vdots \\ t_{W',N} \end{bmatrix} = \begin{bmatrix} X_1 \\ \vdots \\ X_N \end{bmatrix} \cdot W'^{\intercal} \tag{14}$$

       - W' has solutions of cardinality p^{m+n}
       - (14) reduces the cardinality of solutions to p^{m+n-N}
       - $|\operatorname{span}(\{ P_1, \ldots, P_n \})| = p^n \rightsquigarrow |\mathcal{W}_2| = p^{m+n-N} - p^n$
       - the rank of $\left[Z_{g_i}\right]$ in (12) is $\le r \rightsquigarrow |\mathcal{W}_1| = p^{m+n-r} - p^n$
       - $\Pr[W' \overset{R}{\leftarrow} \mathcal{W}_1 \in \mathcal{W}_2] = \frac{|\mathcal{W}_1|}{|\mathcal{W}_2|} = \frac{p^{m-N}-1}{p^{m-r}-1} \le \frac{1}{p^{N-r}}$ $\square$
   - The Second Way: The adversary fixes W' first, and try to find $t_{W',1}, \ldots, t_{W',N} \in \mathbb{F}_p$ to make W' with $t_{W',1}, \ldots, t_{W',N}$ be a deceptive data packet.
     - let $\mathcal{W}_3$ be the set of all the solutions $t_{W',1}, \ldots, t_{W',N}$ in (11) with the fixed W'.
     - the adversary randomly selects an element $t_{W',1}, \ldots, t_{W',N}$ for W' from $\mathcal{W}_3$ and hopes they togother is a deceptive data packet.
   - **Theorem 4.**: the probability of this formed packet is a deceptive data packet is $\le \frac{1}{p^{N-r}}$.
     - Proof:
       - the rank of $\left[V_{g_i}\right]$ in (12) is $\le r \rightsquigarrow |\mathcal{W}_3| = p^{m+n-r} - p^n$
       - $\Pr[W' \overset{R}{\leftarrow} \mathcal{W}_1 \in \mathcal{W}_2] = \frac{|\mathcal{W}_1|}{|\mathcal{W}_3|} = \frac{p^{m-N}-1}{p^{m-r}-1} \le \frac{1}{p^{N-r}}$ $\square$

⊙ Security analyse of tag pollution attack:

- **Theorem 5.**: If the adversary modifies d tags of a data packet, the probability that the d modified tags do not be checked out by a node during the verification process is $\frac{1}{p^d}$.
  - i-th coordinate of the vector Z_g is zero $\rightsquigarrow$ the i-th tag of W will not be checked.
  - the i-th coordinate of Z_g is randomly selected from $\mathbb{F}_p$ $\rightsquigarrow$ the probability of it being zero is $\frac{1}{p}$ $\rightsquigarrow$ the probability that all d modified tags are not checked out is $\frac{1}{p^d}$.

> ⚠️ d here is not the number of safe keys!

⊙ Performance of KEPTE:

- Computational Complexity
  1. the key and tag generation at the source s

$$\begin{gathered}\underline{X_1,\ldots,X_N \leftarrow (3)}\\ \rightsquigarrow \mathrm{O}(N^3+N^2(m+n))\end{gathered}$$

$$\underline{\text{Sign}}$$

$$\left.\begin{array}{l} N \text{ tags} \\ (m+n) \text{ times}/ \end{array}\right\} \rightsquigarrow \mathrm{O}(N(m+n))$$

  2. the key generation and the correctness verification of packets at each node g

$$\begin{gathered}\underline{V_g \leftarrow (5)}\\ \rightsquigarrow N(m+n) \text{ times}\end{gathered}$$

$$\begin{gathered}\underline{\text{Verify}}\\ (N+m+n) \text{ times} \rightsquigarrow \mathrm{O}(m+n+N)\end{gathered}$$

> [❓ 11.02] As I understand it, the number of multiplication operations required in the verification step is $N+m+n$, but DMAC claims that it requires $N(m+n)$.

- Communication Overhead
  1. tags appended to each data packet

$$\underline{\text{Tags}}$$

$$\left.N \text{ tags} \quad \lceil\log_2 p\rceil \text{ bits/tag}\right\} \rightsquigarrow O_b = \frac{\text{tag bits}}{\text{packet bits}} = \frac{N}{m+n}$$

  2. keys distributed

$$\begin{array}{ll} \text{to source} & \left\{ N\ b_i\text{-keys} \quad \text{len}(b_i) \text{ bits/key} \right. \\ \text{to nodes} & \left\{ \begin{array}{ll} 1\ c_g\text{-key} & N\lceil\log_2 p\rceil \text{ bits/key} \\ 1\ Z_g\text{-key} & \text{len}(Z_g) \text{ bits/key} \end{array} \right. \end{array}$$

- Storage Overhead

$$\begin{array}{rl} \text{at source} & N(m+n)\lceil\log_2 p\rceil \text{ bits} \\ \text{at nodes} & 2\times(N+m+n)\lceil\log_2 p\rceil \text{ bits} \end{array}$$

2015 - A Null Space-based MAC Scheme against Pollution Attacks to Random Linear Network Coding

> by Alireza Esfahani

◎ Introductions:

- MacSig [18] is based on homomorphic MACs and homomorphic signatures. Their scheme can resist against data and tag pollution. However, the verification phase increases the computational complexity and delay of the scheme.
- to detect both data pollution and tag pollution attacks in an efficient way $\rightsquigarrow$ an efficient null space-based homomorphic MAC scheme

◎ Null Space Properties:

- Null space (**null key**) is the set of solutions to the equation:

$$\underbrace{\mathbf{A}^{m\times n}}_{\text{a linear map}} \cdot \underbrace{\mathbf{x}}_{\text{null key}} = \mathbf{0}$$

- randomization and the **Subspace Properties** of random network coding:
  in RLNC, the source native packets form a subspace and any linear combination of these native packets belongs to that same subspace.
- these null keys are not randomly generated but calculated at the source node , to ensure their orthogonality to the subspace spanned by the data vectors in a generation ($\text{span}(\{\mathbf{u}_1, \dots, \mathbf{u}_m\})$).
- Similar to the source native packets, the null keys go through random linear combinations, which makes it hard for a malicious node to identify them at its neighbors.

> ❓ how null keys go through random linear combinations?

◎ **D-MAC** Scheme:

- key distribution:

$$\begin{tikzcd} & \boxed{\text{KDC}} & \\ \mathcal{K}_S = \{\mathbf{k}_1, \dots, \mathbf{k}_\ell\},\ \mathbf{k}_v \in \mathbb{F}_p^{m+n+1} \arrow[d] & & \mathcal{K}_{n_i} = \{\mathbf{k}_1, \dots, \mathbf{k}_R\},\ \mathbf{k}_r \in \mathbb{F}_p^{m+n+1} \arrow[d] \\ \boxed{\text{Source}} & & \boxed{\text{Node } n_i} \end{tikzcd}$$

$$\mathcal{K}_{n_i} \subset \mathcal{K}_S$$
$$\mathbf{k}_\ell = \mathbf{k}_r \rightsquigarrow \ell = r$$

  - PKD method [24] is used $\rightsquigarrow \ell < \ell_{\text{MacSig}}$.

> [❓ 11.01] Even though the MAC length in this scheme (ℓ) is less than the MAC length in MacSig (L), it is not sufficient to obtain $\ell + \ell' << L$ as claimed in the paper?

- tag generation:

$$\text{for } \ell \text{ MACs}: \quad \forall v = 1, \dots, \ell: \quad t_{\mathbf{u}_i, v} = -\frac{\sum_{j=1}^{m+n} u_{i,j} \cdot k_{v,j}}{k_{v,m+n+1}} \tag{5}$$

$$\overset{(5)}{\Longrightarrow} \quad \forall v = 1, \dots, \ell: \quad \begin{bmatrix} u_{i,1} & \cdots & u_{i,m+n} & t_v \end{bmatrix} \cdot \begin{bmatrix} k_{v,1} \\ \vdots \\ k_{v,m+n} \\ k_{v,m+n+1} \end{bmatrix} = 0$$

$$\Longrightarrow \quad (\mathbf{u}_i \quad t_v) \perp \mathbf{k}_v$$

$$\text{for } \ell' \text{ D-MACs}: \begin{bmatrix} k_{1,1} & \cdots & k_{1,\ell} & k_{1,\ell+1} & \cdots & k_{1,\ell+\ell'} \\ \vdots & \ddots & \vdots & \vdots & \ddots & \vdots \\ k_{\ell',1} & \cdots & k_{\ell',\ell} & k_{\ell',\ell+1} & \cdots & k_{\ell',\ell+\ell'} \end{bmatrix} \cdot \underbrace{\begin{bmatrix} t_1 \\ \vdots \\ t_\ell \\ t'_1 \\ \vdots \\ t'_{\ell'} \end{bmatrix}}_{\text{"null key"}} = \mathbf{0}^{\ell' \times 1} \tag{6}$$

$$\overset{(6)}{\Longrightarrow} \quad \forall v = 1, \dots, \ell': \quad \underbrace{(t_1 \quad \cdots \quad t_\ell \quad t'_1 \quad \cdots \quad t'_{\ell'}) \perp \mathbf{k}_v}_{\text{orthogonality property}}$$

  - we use ℓ' of ℓ keys which is used for the MAC generation to generate D-MACs.
  - the D-MACs are created in a way that $\mathbf{k}_v$ is orthogonal to the concatenation of all MACs and D-MACs. This relationship is true for all packets in the same generation.

> 🤔 with assumption: $\ell + \ell' \leq m + n + 1$ and $\ell' \leq \ell$

- verification:

$$(\mathbf{y} \in \mathbb{F}_p^{m+n}, \mathbf{t_y}, \mathbf{t'_y})\ \forall \mathbf{k}_r \in \{\mathbf{k}_1, \dots, \mathbf{k}_R\}:$$

$$\begin{aligned}\delta_r &= \Big(\sum_{j=1}^{m+n} y_j \cdot k_{r,j}\Big) + t_{\mathbf{y},r} \cdot k_{r,m+n+1} \\ &= \begin{pmatrix}\mathbf{y} & t_{\mathbf{y},r}\end{pmatrix} \cdot \mathbf{k}_r^\intercal \\ &\overset{?}{=} 0 \quad \overset{\text{yes}}{\rightsquigarrow} \text{pass data check, output } 1\end{aligned} \tag{7}$$

$$\begin{aligned}\delta'_r &= \Big(\sum_{j=1}^{\ell} t_{\mathbf{y},j} \cdot k_{r,j}\Big) + \Big(\sum_{j=L+1}^{\ell'} t'_{\mathbf{y},j} \cdot k_{r,j}\Big) \\ &= \begin{pmatrix}t_{\mathbf{y},1} & \cdots & t_{\mathbf{y},\ell} & t'_{\mathbf{y},1} & \cdots & t'_{\mathbf{y},\ell'}\end{pmatrix} \cdot \mathbf{k}_r^\intercal \\ &\overset{?}{=} 0 \quad \overset{\text{yes}}{\rightsquigarrow} \text{pass tag check, output } 1\end{aligned} \tag{8}$$

◎ Security Analysis:

1. Data attack
   - same as Dual MAC.
2. MAC/D-MAC tag attack
   - assume d tags (MACs or D-MACs) are modified and each node is assigned **only one** key, then the polluted tags pass the verification with probability $\frac{1}{p^d}$.

> [❓ 11.01] 🤔 Proof:
> The more tags an attacker forges, the more likely it is that it will fail authentication, so we only consider the scenario that is most likely to pass authentication, i.e., forging as few tags as possible.
>
> $$\begin{aligned}\begin{pmatrix}t_1 & \cdots & t_\ell & t'_1 & \cdots & t'_{\ell'}\end{pmatrix} \cdot \mathbf{k}^\intercal = 0 \\ \Longrightarrow t_1 \cdot k_1 + \cdots + t_\ell \cdot k_\ell + t'_1 \cdot k_{\ell+1} + \cdots + t'_{\ell'} \cdot k_{\ell+\ell'} = 0\end{aligned}$$
>
> It can be seen that it is not possible to pass verification by modifying only one tag, so at least two need to be modified. We first forge a tag t_1^* and then compute another tag t_2^* in order to pass (8).
>
> $$\begin{aligned}\cdots + t_1^* \cdot k_{t_1^*} + \cdots + t_2^* \cdot k_{t_2^*} + \cdots \overset{!}{=} 0 \\ \Longrightarrow t_1^* = -\frac{\cdots + t_2^* \cdot k_{t_2^*} + \cdots}{k_{t_1^*}}\end{aligned}$$
>
> So we could find a pair of t_1^* and t_2^* that satisfy the equation above. If the attacker does not know the key used by the verifier, then t_2^* cannot be computed and can only be chosen randomly from $\mathbb{F}_p$ with a probability of passing the verification of $\frac{1}{p}$. (But I don't know how the conclusion of $\frac{1}{p^d}$ in the paper was reached.)

> [❓ 11.01] The logic of Algorithm 2 seems to contradict Equation 8. According to the Algorithm 2, the packet is considered passed as long as the verifier uses **one of** the multiple keys held and passes $(\text{count} > 0)$. In contrast, according to Equation 8, the verifier is required to verify using **all keys held** and **pass them all** $(\text{count} == |\mathcal{K}_{n_i}|)$.
> If my proof is correct, then the success rate of tag pollution will still be related to the number of "safe" keys (i.e. keys held by the verifier, but not by the attacker) held by the verifier. And the logic of Algorithm 2 is just wrong. Because if the logic of Algorithm 2 is correct, then if the verifier uses an "unsafe" key to check a tag-polluted packet and passes, then the verifier will also accept the packet.

   - in fact, we assign more than one keys to each node ⇝ the probability of passing the verification is very small.
3. D-MAC tag attack
   - D-MACs are based on Null space properties and calculated by source node ⇝ no possibility to alter these tags.

> [❓ 11.01] In fact, the probability should not be 0. According to Theory 2, D-MAC attacks and MAC attacks can be analysed equally in this scheme. So it's also $\frac{1}{p}$.

◎ Performance Analysis:

- Communication Overhead

$$\left.\begin{matrix}\ell \text{ MACs} \\ \ell' \text{ D-MACs}\end{matrix}\right\} \rightsquigarrow \ell + \ell' \ll L$$

> L is used in MacSig [18]

- Computational Complexity
    - at source node (tags generation):

$$\left.\begin{array}{l} m+n+1 \text{ times}/\text{MAC} \\ \ell'+1 \text{ times}/\text{D-MAC} \end{array}\right\}$$

    - at non-source node (verification):

$$\left.\begin{array}{l} m+n+1 \text{ times}/\text{MAC} \\ \ell+\ell' \text{ times for all D-MACs} \end{array}\right\}$$

> 🤔 seems that it assums that each non-source node has only one key.

2015 - Dual-Homomorphic Message Authentication Code Scheme for Network Coding-Enabled Wireless Sensor Networks

> by Alireza Esfahani

▶ Related Work:

- As a lot of null keys are used in each generation, a high bandwidthoverheadcan be incurred (like D-MAC [2015]).
- the signature used in MacSig [9] is a time-consuming process. Double-random key distribution in MacSig needs to attach the indexes of the keys to each packet, which causes a high overhead.
- Scheme proposed here (Dual-Homomorphic MAC, *Dual-HMAC*) can achieve resistance for the half of the tags against tag pollution attack with significant low computational complexity compared to the MacSig scheme.

▶ **Dual-HMAC** Scheme:

- setup:

$$\boxed{\text{KDC}}$$

$$\begin{array}{cc}
\begin{array}{l}
\mathcal{K}_S = \{\mathbf{k}_1,\dots,\mathbf{k}_L\}, \quad \mathbf{k}_\ell \in \mathbb{F}_p^{m+n+1} \\
\mathcal{K}'_S = \{\mathbf{k}'_1,\dots,\mathbf{k}'_{L'}\}, \quad \mathbf{k}'_{\ell'} \in \mathbb{F}_p^{L+1}
\end{array}
&
\begin{array}{l}
\mathcal{K}_{n_i} = \{\mathbf{k}_1,\dots,\mathbf{k}_R\}, \quad \mathbf{k}_r \in \mathbb{F}_p^{m+n+1} \\
\mathcal{K}'_{n_i} = \{\mathbf{k}'_1,\dots,\mathbf{k}'_{R'}\}, \quad \mathbf{k}'_{r'} \in \mathbb{F}_p^{L+1}
\end{array} \\
\downarrow & \downarrow \\
\boxed{\text{Source}} & \boxed{\text{node } n_i}
\end{array}$$

$$\begin{array}{lrcl}
\mathcal{K}_S \cap \mathcal{K}'_S = \emptyset & i \neq j & \rightsquigarrow & \mathcal{K}_{n_i} \neq \mathcal{K}_{n_j} \ \wedge \ \mathcal{K'}_{n_i} \neq \mathcal{K'}_{n_j} \\
\mathcal{K}_{n_i} \subset \mathcal{K}_S & \mathbf{k}_\ell = \mathbf{k}_r & \rightsquigarrow & \ell = r \\
\mathcal{K'}_{n_i} \subset \mathcal{K'}_S & \mathbf{k}'_{\ell'} = \mathbf{k}'_{r'} & \rightsquigarrow & \ell' = r'
\end{array}$$

    - two sets of symmetric keys (secret) are distributed to all participant nodes in a secure and authenticated manner through a key distribution scheme.
- Tag generation:

$$\left.\begin{array}{l} \mathcal{K}_S \\ \mathbf{u}_i \in \mathbb{F}_p^{m+n} \end{array}\right\} \rightarrow \boxed{\text{MAC}} \rightarrow \left\{\begin{array}{l} t_{\mathbf{u}_i,1},\dots,t_{\mathbf{u}_i,L} \in \mathbb{F}_p \\ t_{\mathbf{u}_i,\ell} = -\dfrac{\sum_{j=1}^{m+n} u_{i,j}\cdot k_{\ell,j}}{k_{\ell,m+n+1}} \end{array}\right. \implies (\mathbf{u}_i \quad t_{\mathbf{u}_i,\ell}) \perp \mathbf{k}_\ell \tag{3}$$

    - the MAC tags are created in a way that $\mathbf{k}_\ell$ is orthogonal to the concatenation of $\mathbf{u}_i$ and its ℓ tags.
    - This relationship is true for all packets in the same generation, which represent as $\mathbf{k}_\ell \perp \operatorname{span}(\{\,(\mathbf{u}_i \quad t_{\mathbf{u}_i,\nu})_{i=1}^m\,\})$

$$\left.\begin{array}{l} \mathcal{K}'_S \\ t_{\mathbf{u}_i,\ell},\dots,t_{\mathbf{u}_i,L} \in \mathbb{F}_p \end{array}\right\} \boxed{\text{D-MAC}} \left\{\begin{array}{l} t'_{\mathbf{u}_i,1},\dots,t'_{\mathbf{u}_i,L'} \in \mathbb{F}_p \\ t'_{\mathbf{u}_i,\ell'} = -\dfrac{\sum_{j=1}^{L} t_{\mathbf{u}_i,j}\cdot k'_{\ell',j}}{k'_{\ell',L+1}} \end{array}\right. \implies (t_{\mathbf{u}_i,1} \quad \cdots \quad t_{\mathbf{u}_i,L} \quad t'_{\mathbf{u}_i,\ell'}) \perp \mathbf{k}'_{\ell'} \tag{4}$$

    - the generation for D-MAC differs from D-MAC
- Combine:

$$\left.\begin{array}{l} (\mathbf{u}_i, t_{\mathbf{u}_i,1},\dots,t_{\mathbf{u}_i,L},t'_{\mathbf{u}_i,1},\dots,t'_{\mathbf{u}_i,L'})_{i=1}^h \in \mathbb{F}_p^{m+n+L+L'} \\ c_1,\dots,c_h \in \mathbb{F}_p \end{array}\right\} \rightarrow \boxed{\text{Combine}} \rightarrow \left\{\begin{array}{l} (\mathbf{y}, t_{\mathbf{y},1},\dots,t_{\mathbf{y},L},t'_{\mathbf{y},1},\dots,t'_{\mathbf{y},L'}) \\ = \sum_{i=1}^{h} c_i \cdot (\mathbf{u}_i, t_{\mathbf{u}_i,1},\dots,t_{\mathbf{u}_i,L},t'_{\mathbf{u}_i,1},\dots,t'_{\mathbf{u}_i,L'}) \end{array}\right. \tag{5}$$

- Verify:

$$\left.\begin{array}{l}\mathcal{K}_{n_i}\\ \mathcal{K}'_{n_i}\\ \mathbf{y}\in\mathbb{F}_p^{m+n}\\ t_{\mathbf{y},1},\dots,t_{\mathbf{y},L}\in\mathbb{F}_p\\ t'_{\mathbf{y},1},\dots,t'_{\mathbf{y},L'}\in\mathbb{F}_p\end{array}\right\}\to\boxed{\text{Verify}}\to\begin{cases} \forall \mathbf{k}_r\in\mathcal{K}_{n_i}: \quad \delta_r = \left(\sum_{j=1}^{m+n} y_j\cdot k_{r,j}\right)+t_{\mathbf{y},r}\cdot k_{r,m+n+1} & (6)\\ \qquad = \left(\mathbf{y}\quad t_{\mathbf{y},r}\right)\cdot \mathbf{k}_r^\intercal & \\ \qquad \overset{?}{=} 0 \quad \overset{\text{yes}}{\rightsquigarrow}\text{pass} & \\ \forall \mathbf{k}'_{r'}\in\mathcal{K}'_{n_i}: \quad \delta'_{r'} = \left(\sum_{j=1}^{L} t_{\mathbf{y},j}\cdot k'_{r',j}\right)+t'_{\mathbf{y},r'}\cdot k'_{r',L+1}\overset{?}{=}0 & (7)\\ \qquad = \left(t_{\mathbf{y},1}\quad\cdots\quad t_{\mathbf{y},L}\quad t'_{\mathbf{y},r'}\right)\cdot \mathbf{k'}_{r'}^\intercal & \\ \qquad \overset{?}{=} 0 \quad \overset{\text{yes}}{\rightsquigarrow}\text{pass} & \end{cases}$$

▶ Key Predistribution Model:

- It is shown in [25] that, in order to resist against less than c compromised nodes, the cardinality of the key size at the source must be $(c+1)$ times larger than those key sizes at the relay/sink nodes:

$$L \geq (c+1)R$$
$$L' \geq (c+1)R'$$

- More specifically, we claim

$$\begin{array}{lll} & L \coloneqq e(c+1)\ln\frac{1}{q} & \overset{\text{for simplicity}}{=} L' \\ \begin{array}{r}\forall r = 1,\dots,R\\ \forall r' = 1,\dots,R'\end{array}: & \Pr[\mathbf{k}_r\in\mathcal{K}_{n_i}] \coloneqq \frac{1}{2(c+1)} & = \Pr[\mathbf{k}'_{r'}\in\mathcal{K}'_{n_i}]\ , \qquad q=\frac{1}{N\cdot\binom{N}{c}} \\ & R \coloneqq L\cdot\Pr[\mathbf{k}_r\in\mathcal{K}_{n_i}] = e\ln\frac{1}{\sqrt{q}} & \overset{\text{for simplicity}}{=} R' \end{array}$$

  - usually $q = 10^{-3}$ would be sufficient.

▶ Security Analysis:

- Data pollution
  - This polluted message can be detected immediately by the next hop, or in a few hops later (maximum $c-1$ hops later).
  - 🤔 MacSig Theorem 4. has proved that the data attack can succeed with probability $\frac{1}{p^d}$ (d is the number of safe MAC keys).
- MAC Tag pollution:
  - very similar to the Data Pollution, e.g. $\frac{1}{p^{d'}}$ (d' is the number of safe DMAC keys).
- DMAC Tag pollution
  - This attack can success in the next hop by the probability of $1-\frac{1}{2(c+1)}$.
    - 🤔 In fact, there is **no** mechanism to protect the DMAC from attack. An attacker randomly chooses only **one** DMAC to tamper with (the more modifications, the higher the probability that the attack will fail) and hopes that the next verifier does not have that corresponding key.

$$\boxed{\text{adversary}}\xrightarrow{N-i\text{ hops}}\boxed{\text{sink}}$$

  - the polluted DMAC can travel some hops, before it is detected, by the probability of $(1-\frac{1}{2(c+1)})^{N-i}$
  - **Theorem 3.**: The probability that a downstream user receives the same key as key_x of user x is $\geq (1-\frac{1}{2(c+1)})^{N-i}$.

> [❓ 10.30] If DMAC cannot be protected, then this proposed scheme cannot strictly be considered resistant to tag pollution. Because the attacker also knows which bits are DMAC and can target those bits.

▶ Performance Evaluation:

- Communication Overhead

$$\left.\begin{array}{ll} L\ \text{MACs} & \lceil\log_2 p\rceil\ \text{bits/MAC}\\ L'\ \text{D-MACs} & \lceil\log_2 p\rceil\ \text{bits/D-MAC}\end{array}\right\}\rightsquigarrow O_b=\frac{(L+L')\log_2 p}{(m+n)\log_2 p}\overset{L+L'=\ell}{=}\frac{\ell}{m+n}$$

> [❓ 10.30] This paper makes $L+L'=\ell$ (ℓ is the length of MACs used in HomMac), but according to the KDC presented in this paper, it should be $L=L'=\ell$. However, if we consider the number of keys each node has, then there are indeed $R+R'=r$ (r is the number of

> keys hold by each node who use HomMac).

- Computation Overhead

$$\underline{\text{MAC}}$$
$$L(m+n+1) \text{ times}$$
$$\underline{\text{D-MAC}}$$
$$L'(L+1) \text{ times}$$
$$\underline{\text{Verify}}$$
$$\left.\begin{array}{ll} L(m+n+1) \text{ times} & \text{for MACs} \\ L'(L+1) \text{ times} & \text{for D-MACs} \end{array}\right\}$$

2016 - Two Improved Homomorphic MAC Schemes in Network Coding

> by Xinran Li

◊ related work:

- an adversary can break the MAC in HomMac [16], MacSig [18] with probability $\frac{1}{p}$ $\rightsquigarrow$ we want to improve security level by increase the size of the tag space.

◊ Attack game 1:

$$\begin{tikzcd} & & \mathbf{k} \overset{R}{\leftarrow} \mathcal{K} \arrow[d] \\ \boxed{\text{adversary } A} \arrow[rr, shift left, "\text{query for } \mathbf{y}_1,\ldots,\mathbf{y}_m"] \arrow[d] & & \boxed{\text{challenger } C} \arrow[ll, shift left, "\mathbf{t}_1,\ldots,\mathbf{t}_m"] \\ \mathbf{y}^*, \mathbf{t}^* & & \end{tikzcd}$$

$$A \text{ wins} := \begin{cases} \mathbf{y}^* \notin \text{span}(\{\mathbf{y}_1, \ldots, \mathbf{y}_m\}) \\ \text{verify}(\mathbf{k}, \mathbf{y}^*, \mathbf{t}^*) = 1 \end{cases}$$

◊ Construction 1 (**DIP MAC** scheme):

- Setup:

$$\begin{array}{ll} \text{Pseudo-Random Generator } G: & \mathcal{K}_G \to \mathbb{F}_q^{n+m} \\ K \in \mathcal{K}_G & \mathbf{k} \leftarrow G(K) \in \mathbb{F}_q^{n+m} \end{array}$$

$$\begin{tikzcd} & K \arrow[dl] \arrow[dr] & \\ \boxed{\text{Source}} & & \boxed{\text{Verifier}} \end{tikzcd}$$

$$n+m = \ell s \rightsquigarrow \begin{array}{ll} \mathbf{v} = (v_1, \ldots, v_{n+m}) & \overset{\text{divide}}{=} (\underbrace{v_1, \ldots, v_s}_{1 \text{ set}}, \ldots, \underbrace{v_{(\ell-1)s+1}, \ldots, v_{\ell s}}_{\ell \text{ set}}) \\ \mathbf{k} = (k_1, \ldots, k_{n+m}) & \overset{\text{divide}}{=} (\underbrace{k_1, \ldots, k_s}_{1 \text{ set}}, \ldots, \underbrace{k_{(\ell-1)s+1}, \ldots, k_{\ell s}}_{\ell \text{ set}}) \end{array}$$

- MAC:

$$\left.\begin{array}{l} \mathbf{v} \in \mathbb{F}_q^{n+m} \\ K \in \mathcal{K}_G \end{array}\right\} \to \boxed{\text{MAC}} \to \begin{cases} \forall j = 1, \ldots, \ell: \quad t_j = \sum_{i=1}^{s} v_{(j-1)s+i} \cdot k_{(j-1)s+i} \\ \mathbf{t} = (t_1, \ldots, t_\ell) \in \mathbb{F}_q^\ell \end{cases} \quad (1)$$

- Combine: omit
- Verify:

$$\left.\begin{matrix}(\mathbf{v},\mathbf{t})\\ K\end{matrix}\right\} \rightarrow \boxed{\text{Verify}} \rightarrow \begin{cases}\forall j=1,\dots,\ell: \quad t'_j = \sum_{i=1}^{s} v_{(j-1)s+i}\cdot k_{(j-1)s+i}\\ \qquad\qquad\qquad \mathbf{t}' = (t'_1,\dots,t'_\ell)\in\mathbb{F}_q^\ell\\ \qquad\qquad\qquad \mathbf{t}\overset{?}{=}\mathbf{t}' \overset{\text{yes}}{\rightsquigarrow} \text{pass}\end{cases}$$

◊ Attack game 2.:

$$\begin{tikzcd}
 & & \mathbf{k}\overset{R}{\leftarrow}\mathcal{K} \arrow[d] \\
\boxed{\text{adversary } A} \arrow[rr, "\text{query for } \mathbf{y}_1,\dots,\mathbf{y}_m", shift left] \arrow[d] & & \boxed{\text{challenger } C} \arrow[ll, "\mathbf{t}_1,\dots,\mathbf{t}_m", shift left] \\
\mathbf{y}^* = (\mathbf{y}^{*(1)},\dots,\mathbf{y}^{*(\ell)}),\mathbf{t}^* & &
\end{tikzcd}$$

$$A \text{ wins} := \begin{cases}\mathbf{y}^{*(j)}\notin \text{span}(\{\mathbf{y}_1^{(j)},\dots,\mathbf{y}_m^{(j)}\}) \quad \forall j=1,\dots,\ell\\ \text{verify}(\mathbf{k},\mathbf{y}^*,\mathbf{t}^*)=1\end{cases}$$

- $\mathbf{y}^{*(j)}\notin \text{span}(\{\mathbf{y}_1^{(j)},\dots,\mathbf{y}_m^{(j)}\}),\ \forall j=1,\dots,\ell \implies \mathbf{y}^*\notin\text{span}(\{\mathbf{y}_1,\dots,\mathbf{y}_m\}) \rightsquigarrow$ if an adversary A wins the attack game 2, he can also wins the attack game 1, but **not** vice versa.

◊ Security Analysis of DIP MAC:

- **Theorem 2.**: For any fixed q,n,m, the DIP MAC is a secure MAC with respect to *attack game 2* assuming the PRG G is a secure PRG. Specially, for all homomorphic MAC adversaries A, there is a PRG adversary B who has similar running time to A, such that

$$\text{Adv}[A,\text{DIP MAC}] \le \text{PRG-Adv}[B,G] + \frac{1}{q^\ell} \tag{2}$$

  - Proof similar to HomMac.

$$\overset{(5)}{\Longrightarrow} \begin{bmatrix} t_{1,1}\\ \vdots\\ t_{1,\ell}\\ t_{2,1}\\ \vdots\\ t_{2,\ell}\\ \vdots\\ t_{m,1}\\ \vdots\\ t_{m,\ell}\end{bmatrix} = \begin{bmatrix} y_{1,1} & \cdots & y_{1,s} & \ddots & & & \\ & & & & y_{1,(\ell-1)s+1} & \cdots & y_{1,\ell s}\\ y_{2,1} & \cdots & y_{2,s} & \ddots & & & \\ & & & & y_{2,(\ell-1)s+1} & \cdots & y_{2,\ell s}\\ & & & \vdots & & & \\ y_{m,1} & \cdots & y_{m,s} & \ddots & & & \\ & & & & y_{m,(\ell-1)s+1} & \cdots & y_{m,\ell s}\end{bmatrix} \cdot \begin{bmatrix} k_1\\ \vdots\\ k_s\\ k_{s+1}\\ \vdots\\ k_{2s}\\ \vdots\\ k_{(\ell-1)s+1}\\ \vdots\\ k_{\ell s}\end{bmatrix} \tag{6}$$

$$\overset{(6)}{\Longrightarrow} \begin{bmatrix} t_{1,1}\\ t_{2,1}\\ \vdots\\ t_{m,1}\end{bmatrix} = \begin{bmatrix} y_{1,1} & \cdots & y_{1,s}\\ y_{2,1} & \cdots & y_{2,s}\\ \vdots & & \vdots\\ y_{m,1} & \cdots & y_{m,s}\end{bmatrix} \cdot \begin{bmatrix} k_1\\ \vdots\\ k_s\end{bmatrix} \tag{7}$$

$$\begin{bmatrix} t_{1,2}\\ t_{2,2}\\ \vdots\\ t_{m,2}\end{bmatrix} = \begin{bmatrix} y_{1,s+1} & \cdots & y_{1,2s}\\ y_{2,s+1} & \cdots & y_{2,2s}\\ \vdots & & \vdots\\ y_{m,s+1} & \cdots & y_{m,2s}\end{bmatrix} \cdot \begin{bmatrix} k_{s+1}\\ \vdots\\ k_{2s}\end{bmatrix} \tag{8}$$

$$\vdots$$

$$\begin{bmatrix} t_{1,\ell}\\ t_{2,\ell}\\ \vdots\\ t_{m,\ell}\end{bmatrix} = \begin{bmatrix} y_{1,(\ell-1)s+1} & \cdots & y_{1,\ell s}\\ y_{2,(\ell-1)s+1} & \cdots & y_{2,\ell s}\\ \vdots & & \vdots\\ y_{m,(\ell-1)s+1} & \cdots & y_{m,\ell s}\end{bmatrix} \cdot \begin{bmatrix} k_{(\ell-1)s+1}\\ \vdots\\ k_{\ell s}\end{bmatrix} \tag{9}$$

$$\overset{\text{insert forged one}}{\Longrightarrow} \begin{bmatrix} t_{1,1}\\ t_{2,1}\\ \vdots\\ t_{m,1}\\ t_1^*\end{bmatrix} = \begin{bmatrix} y_{1,1} & \cdots & y_{1,s}\\ y_{2,1} & \cdots & y_{2,s}\\ \vdots & & \vdots\\ y_{m,1} & \cdots & y_{m,s}\\ y_1^* & \cdots & y_s^*\end{bmatrix} \cdot \begin{bmatrix} k_1\\ \vdots\\ k_s\end{bmatrix} \tag{7*}$$

- Let R be the rank of the system (7) consists of m equations and s unknowns.
- $R \leq m \wedge s > m \implies R < s \rightsquigarrow (7)$ has solutions of cardinality q^{s-R}.
- $\mathbf{y}^{*(j)} \notin \operatorname{span}(\{\mathbf{y}_1^{(j)}, \ldots, \mathbf{y}_m^{(j)}\}) \rightsquigarrow$ the rank of system (7^*) increases by 1 $\rightsquigarrow$ (7^*) has solutions of cardinality q^{s-R-1}.
- therefore

$$\begin{aligned} \Pr[W_{1,1}] = \frac{q^{s-R-1}}{q^{s-R}} &= \frac{1}{q} \\ \Pr[W_1] = \prod_{j=1}^{\ell} \Pr[W_{1,j}] &= \frac{1}{q^\ell} \end{aligned} \tag{10}$$

⋄ Construction 2:

- Setup:

$$\mathbf{k} \xleftarrow{R} \mathbb{F}_q^{n+m+\ell} \quad \wedge \quad \begin{cases} (k_1, \ldots, k_{n+m}) \neq \mathbf{0} \\ k_{n+m+j} \neq 0 & \forall j = 1, \ldots, \ell \end{cases}$$

- MAC:

$$\left.\begin{matrix} \mathbf{k} \in \mathbb{F}_q^{n+m+\ell} \\ \mathbf{v} \in \mathbb{F}_q^{n+m} \end{matrix}\right\} \to \boxed{\text{MAC}} \to \begin{cases} \forall j = 1, \ldots, \ell: \quad t_j = -\dfrac{\sum_{h=1}^{n+m} v_{f_i(h)} k_h}{k_{n+m+j}} \\ \\ \qquad\qquad \mathbf{t} = (t_1, \ldots, t_\ell) \in \mathbb{F}_q^\ell \end{cases} \tag{11}$$

- Verify:

$$\left.\begin{matrix} \mathbf{v} \\ \mathbf{k} \\ \mathbf{t} \end{matrix}\right\} \to \boxed{\text{Verify}} \to \begin{cases} \forall j = 1, \ldots, \ell: \quad t'_j = -\dfrac{\sum_{h=1}^{n+m} v_{f_i(h)} k_h}{k_{n+m+j}} \\ \\ \qquad\qquad \mathbf{t}' = (t'_1, \ldots, t'_\ell) \in \mathbb{F}_q^\ell \\ \qquad\qquad \mathbf{t} \overset{?}{=} \mathbf{t}' \quad \overset{\text{yes}}{\rightsquigarrow} \text{pass} \end{cases}$$

⋄ **Theorem 4.**: If $m\ell < n + m + \ell$, then the construction 2 is a secure homomorphic MAC with respect to *attack game 1*.

- Proof.

$$\begin{bmatrix} y_{1,f_1(1)} & \cdots & y_{1,f_1(n+m)} & t_{1,1} & \cdots & 0 \\ \vdots & & \vdots & \vdots & & \vdots \\ y_{1,f_\ell(1)} & \cdots & y_{1,f_\ell(n+m)} & 0 & \cdots & t_{1,\ell} \\ \vdots & & \vdots & \vdots & & \vdots \\ y_{m,f_1(1)} & \cdots & y_{m,f_1(n+m)} & t_{m,1} & \cdots & 0 \\ \vdots & & \vdots & \vdots & & \vdots \\ y_{m,f_\ell(1)} & \cdots & y_{m,f_\ell(n+m)} & 0 & \cdots & t_{m,\ell} \end{bmatrix} \cdot \begin{bmatrix} k_1 \\ \vdots \\ k_{n+m} \\ k_{n+m+1} \\ \vdots \\ k_{n+m+\ell} \end{bmatrix} = \mathbf{0} \tag{13}$$

$$\xrightarrow{\text{rewrite (13) and insert forged one}} \begin{bmatrix} \mathbf{y}_{1,f_1} & t_{1,1} & 0 & \cdots & 0 \\ \mathbf{y}_{2,f_1} & t_{2,1} & 0 & \cdots & 0 \\ \vdots & \vdots & \vdots & & \vdots \\ \mathbf{y}_{m,f_1} & t_{m,1} & 0 & \cdots & 0 \\ \mathbf{y}^*_{f_1} & t^*_1 & 0 & \cdots & 0 \\ \mathbf{y}_{1,f_2} & 0 & t_{1,2} & \cdots & 0 \\ \mathbf{y}_{2,f_2} & 0 & t_{2,2} & \cdots & 0 \\ \vdots & \vdots & \vdots & & \vdots \\ \mathbf{y}_{m,f_2} & 0 & t_{m,2} & \cdots & 0 \\ \mathbf{y}^*_{f_2} & 0 & t^*_2 & \cdots & 0 \\ \vdots & \vdots & \vdots & & \vdots \\ \mathbf{y}_{1,f_\ell} & 0 & 0 & \cdots & t_{1,\ell} \\ \mathbf{y}_{2,f_\ell} & 0 & 0 & \cdots & t_{2,\ell} \\ \vdots & \vdots & \vdots & & \vdots \\ \mathbf{y}_{m,f_\ell} & 0 & 0 & \cdots & t_{m,\ell} \\ \mathbf{y}^*_{f_\ell} & 0 & 0 & \cdots & t^*_\ell \end{bmatrix} \cdot \begin{bmatrix} k_1 \\ \vdots \\ k_{n+m} \\ k_{n+m+1} \\ \vdots \\ k_{n+m+\ell} \end{bmatrix} = \mathbf{0} \tag{14}$$

  - let R be the rank of the system (13) consists of $m\ell$ equations and $n + m + \ell$ unknowns.
  - $R \leq m\ell \wedge m\ell < n + m + \ell \implies R < n + m + \ell \rightsquigarrow$ the system (13) has solutions of cardinality $q^{n+m+\ell-R}$.

- after inserting the forged one, the rank of the system (14) increases by $\ell \rightsquigarrow$ the system (14) has solutions of cardinality $q^{n+m+\ell-R-\ell} = q^{n+m-R}$.
- therefore

$$\Pr[A \text{ wins attack game } 1] = \frac{q^{n+m-R}}{q^{n+m+\ell-R}} = \frac{1}{q^{\ell}}$$

◊ Performance Evaluation:

- Construction 1:

$$\left.\begin{array}{ll} & \underline{\text{Verify}} \\ s & \text{multiplications/tag} \\ \ell & \text{tags/packet} \end{array}\right\} \rightsquigarrow s\ell = n+m \text{ multiplications/packet}$$

- Construction 2:

$$\left.\begin{array}{ll} & \underline{\text{Verify}} \\ n+m & \text{multiplications/tag} \\ \ell & \text{tags/packet} \end{array}\right\} \rightsquigarrow (n+m)\ell \text{ multiplications/packet}$$

> ? the paper says it's $n+m+\ell$ not $(n+m)\ell$.

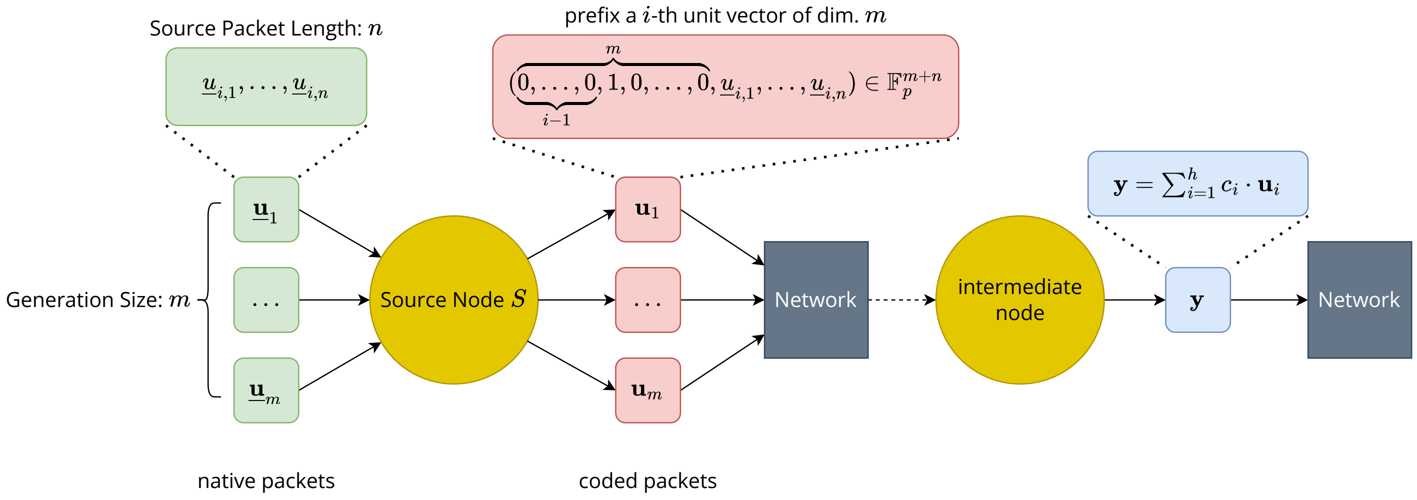
2017 - An efficient homomorphic MAC-based scheme against data and tag pollution attacks in network coding-enabled wireless networks

> by Alireza Esfahani

❖ Symbols Replacement:

Symbols in the paper	⥋	Symbols in this draft
$\mathcal{K}_i$	⥋	$\mathbf{k}_i$
$\mathcal{K'}_i$	⥋	$\mathbf{k'}_i$
Pr_K	⥋	$\mathbf{k}_{\text{pr}}$
PU_K	⥋	$\mathbf{k}_{\text{pu}}$

❖ **RLNC Network-Coding w/o Model**:

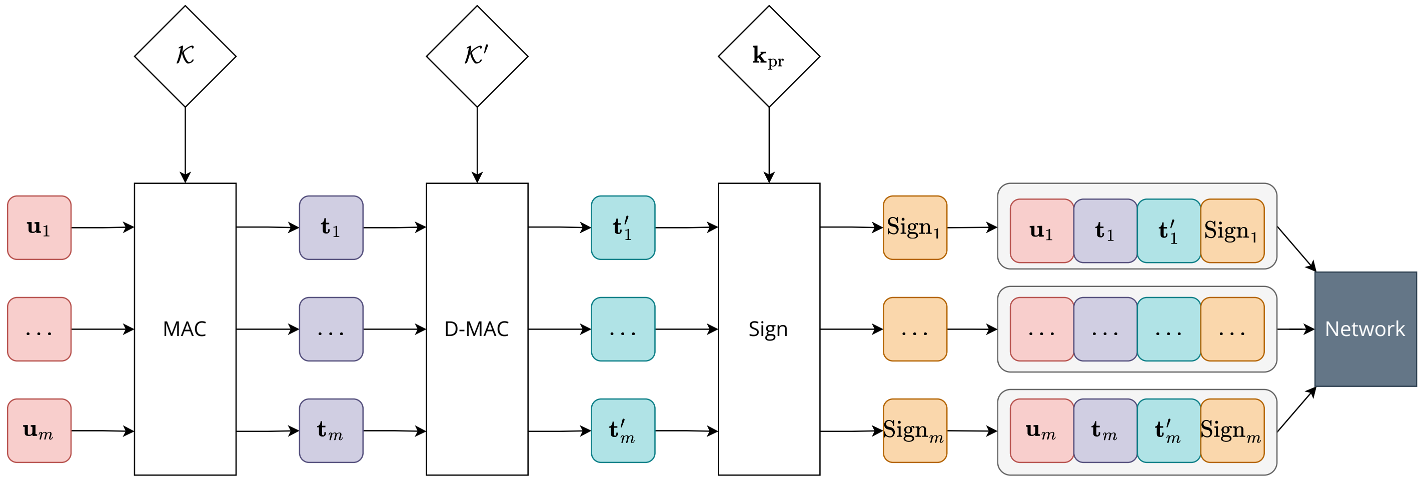

Source Packet Length: n
$\underline{u}_{i,1}, \dots, \underline{u}_{i,n}$
prefix a i-th unit vector of dim. m
$(\overbrace{\underbrace{0,\dots,0}_{i-1},1,0,\dots,0}^{m}, \underline{u}_{i,1},\dots,\underline{u}_{i,n}) \in \mathbb{F}_p^{m+n}$
Generation Size: m
$\underline{\mathbf{u}}_1$
$\dots$
$\underline{\mathbf{u}}_m$
Source Node S
$\mathbf{u}_1$
$\dots$
$\mathbf{u}_m$
Network
intermediate node
$\mathbf{y} = \sum_{i=1}^{h} c_i \cdot \mathbf{u}_i$
$\mathbf{y}$
Network
native packets
coded packets

$$\underline{\mathbf{u}}_i = (\underline{u}_{i,1}, \dots, \underline{u}_{i,n})$$

$$\mathbf{u}_i = (\overbrace{\underbrace{0,\dots,0}_{i-1},1,0,\dots,0}^{m}, \underline{u}_{i,1},\dots,\underline{u}_{i,n}) \in \mathbb{F}_p^{m+n}$$

- we use i-th unit vectors in order to carry *coding coefficients*

❖ **NC w/ HMAC in Source Node**:



$$\begin{aligned} \mathbf{k}_i \in \mathcal{K} = \{\,\mathbf{k}_1,\dots,\mathbf{k}_\ell\,\}, &\qquad \mathbf{k}_i \in \mathbb{F}_p^{m+n+1},\ i=1,\dots,\ell \\ \mathbf{k'}_i \in \mathcal{K}' = \{\,\mathbf{k'}_1,\dots,\mathbf{k'}_{\ell'}\,\}, &\qquad \mathbf{k'}_i \in \mathbb{F}_p^{\ell+1},\ i=1,\dots,\ell' \\ \mathbf{k}_{\text{pr}} = (\beta_1,\dots,\beta_{\ell'+1}) &\quad \rightsquigarrow\ \mathbf{k}_{\text{pu}} = (g^{\beta_1},\dots,g^{\beta_{\ell'+1}}) \end{aligned}$$

$$\ell = \ell' = \frac{L}{2} = \frac{1}{1-\delta} e(c+1) \ln q^{-2} \tag{9}$$

[? 10.27] $\frac{1}{2} \cdot \ln q$ should equal to $\ln \sqrt{q}$ instead of $\ln q^{-2}$.

- L is determined in MacSig [16].

δ, q are secuirity parameters, c is the number of compromised nodes.

❖ HMAC Scheme Construction:

- Setup, distributed by KDC:

$$\boxed{\text{KDC}}$$

$$\begin{tikzcd} {\color{red}\mathcal{K},\ \mathcal{K}',\ \mathbf{k}_{\text{pr}}} \arrow[d] & {\color{red}\mathcal{K}_i,\ \mathcal{K}'_i,}\ {\color{green}\mathbf{k}_{\text{pu}}} \arrow[d] \\ \boxed{\text{Source}} & \boxed{\text{Node } i} \end{tikzcd}$$

$$\mathcal{K}_i \subset \mathcal{K}$$
$$\mathcal{K}'_i \subset \mathcal{K}'$$

  - key distribution model is based on cover free set systems proposed in HomMac [14] and [30].
  - It is worthwhile to mention that our key distribution model takes into consideration that the number of keys at the source should be ***c* times more** than the number of the keys assigned to each intermediate or sink node in order to provide resistance against c compromised nodes.
  - when the key assignment has been completed, resistance against c compromised nodes can be achieved.
  - we consider each node has **only one** assigned key (similar to D-MAC [2015])
- Tag and Sign Generation:

$$\forall i = 1,\dots,\ell: \qquad t_i = -\frac{\sum_{j=1}^{m+n} k_{i,j} \cdot u_j}{k_{i,m+n+1}} \tag{3}$$

$$\overset{(3)}{\Longrightarrow} \ (\mathbf{u} \quad t_i) \perp \mathbf{k}_i$$

$$\forall i' = 1,\dots,\ell': \qquad t'_{i'} = -\frac{\sum_{r=1}^{\ell} k'_{i',r} \cdot t_r}{k'_{i',\ell+1}} \tag{4}$$

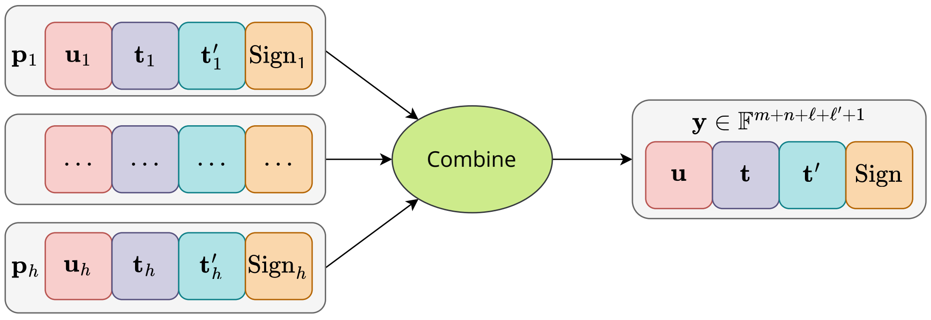
$$\overset{(4)}{\Longrightarrow} \ (t_1 \quad \cdots \quad t_\ell \quad t'_{i'}) \perp \mathbf{k}'_{i'}$$

$$\text{Sign} = -\frac{\sum_{i=1}^{\ell'} k_{\text{pr},i} \cdot t'_i}{k_{\text{pr},\ell'+1}} \tag{5}$$

$$\overset{(5)}{\Longrightarrow} \ (t'_1 \quad \cdots \quad t'_{\ell'} \quad \text{Sign}) \perp \mathbf{k}_{\text{pr}}$$

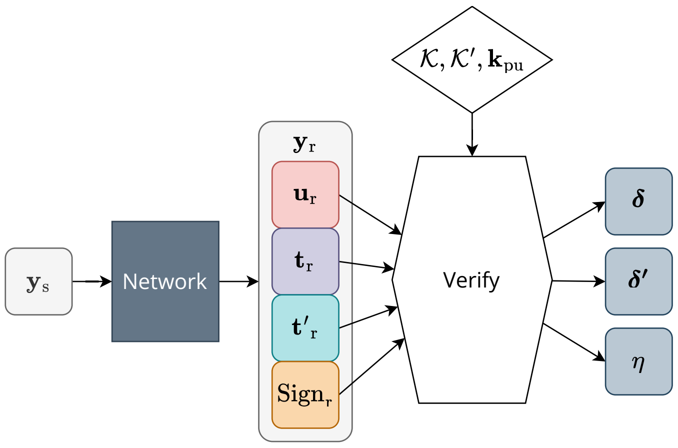
- The generation of D-MACs differ from D-MAC [2015], same as Dual-MAC.

- Combine:



$$\mathbf{y} := \sum_{i=1}^{h} c_i \cdot \mathbf{p}_i \in \mathbb{F}^{m+n+\ell+\ell'+1}, \qquad c_i \in \mathbb{F}_p$$

- Verify:


$\mathcal{K}, \mathcal{K}', \mathbf{k}_{\mathrm{pu}}$
$\mathbf{y}_{\mathrm{s}}$
Network
$\mathbf{y}_{\mathrm{r}}$
$\mathbf{u}_{\mathrm{r}}$
$\mathbf{t}_{\mathrm{r}}$
$\mathbf{t}'_{\mathrm{r}}$
$\mathrm{Sign}_{\mathrm{r}}$
Verify
$\boldsymbol{\delta}$
$\boldsymbol{\delta}'$
η

$$\forall i = 1, \dots, \ell: \quad \delta_i = \left(\sum_{j=1}^{m+n} y_j \cdot k_{i,j}\right) + t_i \cdot k_{i,m+n+1} = \begin{pmatrix} \mathbf{y} & t_i \end{pmatrix} \cdot \mathbf{k}_i^{\mathsf{T}} \overset{?}{=} 0 \overset{\text{yes}}{\rightsquigarrow} \text{pass} \tag{6}$$

$$\forall i' = 1, \dots, \ell': \quad \delta'_{i'} = \left(\sum_{r=1}^{\ell} t_r \cdot k'_{i',r}\right) + t'_{i'} \cdot k'_{i',\ell+1} = \begin{pmatrix} t_1 & \cdots & t_\ell & t'_{i'} \end{pmatrix} \cdot \mathbf{k'}_{i'}^{\mathsf{T}} \overset{?}{=} 0 \overset{\text{yes}}{\rightsquigarrow} \text{pass} \tag{7}$$

$$\eta = \left(\prod_{i=1}^{\ell'} k_{\mathrm{pu},i}{}^{t'_i}\right) \cdot k_{\mathrm{pu},\ell'+1}{}^{\mathrm{Sign}} \overset{?}{=} 1 \overset{\text{yes}}{\rightsquigarrow} \text{pass} \tag{8}$$

❖ Security analysis of HMAC:

- Data pollution
  - **Theorem 1.**: Without making any changes in the MACs, the D-MACs and the signature, the probability that the adversary can pass the verification of the polluted packet, is not greater than $\frac{1}{p}$.
    - The adversary aims to modify a legitimate packet in order to create a polluted packet X' with the *same* MACs, the D-MACs and signature which can pass verification
    - MACs, D-MACs and Sign are not changed $\rightsquigarrow$ the adversary needs to change ≥ 2 symbols of the legitimate packet $\rightsquigarrow$ change one symbol in content and find another one in order to pass (6).
    - Thus, the probability that the polluted packet passes the verification is equal to find a symbol from the finite field $\mathbb{F}_p$. This probability is equal to $\frac{1}{p}$.
  - **Theorem 2.**: Without making any changes in the MACs, the D-MACs and the signature, and by considering that there are d neighbor nodes for the adversary, the probability that the polluted packet can pass the verification of the neighbor nodes depends on which keys they have and is $\leq \frac{1}{\ell^d}$.

- the polluted packet has passed the verification through (6) (in comprised node), so that the probability that the next hops will consider the polluted packet as a legitimate one depends on the keys that they have.
- each neighbor has only one key $\rightsquigarrow$ the probability that the next node has the same key with the compromised node is $\frac{1}{\ell}$
- For d nodes, it happens with the probability of $\frac{1}{\ell^d}$

- Tag pollution
  - **Theorem 3.**: If the adversary modifies any of the MACs or the D-MACs of a packet that he intends to pollute, the probability that the polluted packet will pass the verification through the (8) is similar to the probability of solving a discrete logarithm problem, which is negligible $(\frac{1}{p})$.
    - We assume that the adversary has to calculate a new Sign', which can satisfy (8), for the polluted MACs and D-MACs.
    - Derive Sign' from (8) is similar to solve a DL-problem, which is equivalent to randomly choose one element from $\mathbb{F}_p$.
  - **Theorem 4.**: If the adversary modifies the signature of a packet that he intends to pollute, the probability of the polluted packet to pass the verification through (8), on the next node, is 0.
    - since every node holds the public key $\mathbf{k}_{\text{pu}}$.
  - **Theorem 5.**: The probability that the adversary has the same key vectors with d neighbor nodes is almost $\leq \dfrac{1}{(\ell \cdot \ell')^d}$.

❖ Performance Evaluation:

- Computational complexity
  1. At source node (tags and sign generation)

$$\left.\begin{array}{ll} \ell \text{ MACs} & m+n+1 \text{ times/MAC} \\ \ell' \text{ D-MACs} & \ell+1 \text{ times/D-MAC} \\ 1 \text{ Sign} & \ell'+1 \text{ times/Sign} \end{array}\right\}$$

  2. At Non-source nodes (verification)

$$\left.\begin{array}{ll} m+n+1 \text{ times/MAC} & \\ \ell+1 \text{ times/D-MAC} & \\ \ell'+1 \text{ exponentiations/Sign} & \equiv \frac{3}{2}|p| \text{ times/Sign} \end{array}\right\}$$

- Communication overhead

$$L = \ell + \ell' \rightsquigarrow L+1 \text{ symbols/packet} = \frac{1}{1-\delta} e(c+1) \ln q + 1 \text{ bits/packet}$$

> ❓ $\ln q$ or $\ln \frac{1}{q}$?

- Key storage overhead
  1. At source node

$$\left.\begin{array}{ll} \ell \text{ keys for MACs} & m+n+1 \text{ symbols/key} \\ \ell' \text{ keys for D-MACs} & \ell+1 \text{ symbols/key} \\ 1 \text{ key for Sign} & \ell'+1 \text{ symbols/key} \end{array}\right\} \rightsquigarrow \text{Total} = (\ell \cdot (m+n+1) + \ell' \cdot (\ell+1) + (\ell'+1)) \cdot \lceil \log_2 p \rceil \text{ bits}$$

  2. At non-source nodes

$$\left.\begin{array}{ll} 1 \text{ key for MACs} & m+n+1 \text{ symbols/key} \\ 1 \text{ key for D-MACs} & \ell+1 \text{ symbols/key} \\ 1 \text{ key for Sign} & \ell'+1 \text{ symbols/key} \end{array}\right\} \rightsquigarrow \text{Total} = ((m+n+1) + (\ell+1) + (\ell'+1)) \cdot \lceil \log_2 p \rceil \text{ bits}$$

2022 - E-HMAC: An Efficient Secure Homomorphic MAC Scheme for NC-Enabled WSNs

> by Haythem Hayouni

5, 9

▷ Related Work:

- [9] has drawback in terms of security against tag pollution attack $\rightsquigarrow$ improve Dual-HMAC $\rightsquigarrow$ proposed **E-HMAC** scheme based on multi-linear space.

▷ E-HMAC Scheme:

- key setup

$$\begin{aligned}
&\text{pseudo random generator } G: && \mathrm{KE}_G \rightarrow M(r+s,x) \\
&\text{pseudo random field } F: && \mathrm{KE}_F \times x \times (s+q) \rightarrow \mathbb{F}_q \\
&\text{random secret key }: && \mathrm{SK} = (\mathrm{KE}_1, \mathrm{KE}_2) \overset{R}{\leftarrow} \mathrm{KE}_G \times \mathrm{KE}_F
\end{aligned} \tag{5}$$

output:

$$\begin{aligned}
\mathrm{PK} &= (\ell; r; s; q; G; F) \\
\mathrm{SK} &= (\mathrm{KE}_1, \mathrm{KE}_2)
\end{aligned}$$

> ? what is space key KE?
> ? what is operation M?
> ? what is pseudo random field?
> ? what does $(r+s)$ mean? in other reference, it is $(m+n)$, and m is the number of base vectors, n is the number of data symbols.
> There is nothing to do with the field size.

- MAC Generation:

$$\mathrm{MAC}(\mathrm{PK}; \mathrm{SK}; \mathrm{id}; \mathrm{Vect}; b)$$

  - compute transition matrix $\mathbf{SK}'$:

$$\begin{aligned}
\mathbf{SK}' &= G(\mathrm{KE}_1) \\
\rightsquigarrow \quad & f_{\mathbf{SK}'}: \mathbb{F}_\ell^{r+s} \rightarrow \mathbb{F}_\ell^x
\end{aligned} \tag{6}$$

  - compute the dimensional vector $\mathbf{a}(\mathrm{Vect})$, $\mathrm{Vect} \in \mathbb{F}_\ell^r$:

$$\mathbf{a}(\mathrm{Vect}) = \left(\sum_{b=1}^s \left(\mathrm{Vect}_{r+i} \cdot F(\mathbf{SK}', \mathrm{id}, 1, b)\right) \quad \dots \quad \sum_{i=1}^s \left(\mathrm{Vect}_{r+i} \cdot F(\mathbf{SK}', \mathrm{id}, x, b)\right)\right) \in \mathbb{F}_\ell^x \tag{7}$$

  - denote $\mathbf{y}(\mathrm{Vect})$ the x-dimensional vector:

$$\mathbf{y}(\mathrm{Vect}) = f_{\mathbf{SK}'}(\mathrm{Vect}) + \mathbf{a}(\mathrm{Vect}) \tag{8}$$

  - let $\mathbf{Mt}$ be the $(q \times x)$-Matrix over $\mathbb{F}_\ell$:

$$\mathbf{Mt} = \begin{bmatrix} F(\mathbf{SK}', \mathrm{id}, 1, s+1) & \cdots & F(\mathbf{SK}', \mathrm{id}, x, s+1) \\ \vdots & \ddots & \vdots \\ F(\mathbf{SK}', \mathrm{id}, 1, s+q) & \cdots & F(\mathbf{SK}', \mathrm{id}, x, s+q) \end{bmatrix} = \begin{bmatrix} u_{11} & \cdots & u_{1x} \\ \vdots & \ddots & \vdots \\ u_{q1} & \cdots & u_{qx} \end{bmatrix} \tag{9}$$

  - compute the tags:

$$\begin{aligned}
\forall i = 1, \dots, q;\ j = 1, \dots, x: \quad \mathrm{tg}_{ij}(\mathrm{Vect}) &= \frac{y_j(\mathrm{Vect})}{u_{ij}} && (10) \\
\mathbf{TAG}(\mathrm{Vect}) &= (\mathrm{tg}_{ij})^{q \times x} && (11)
\end{aligned}$$

> ? here is also a problem: the element is u_{ij} then for $i = 1, \dots, q$ and $j = 1, \dots, x$, there should be u_{1x} and u_{qx}

- Combine:

$$\mathbf{TAG}' = \sum_{h=1}^{w} c_h \cdot \mathbf{TAG}(\mathrm{Vect}_h) \tag{12}$$

  - w is the number of relay nodes.
- Verify:
  - compute transition matrix $\mathbf{SK}''$:

$$\mathbf{SK}'' = G(\mathrm{KE}_1) \tag{13}$$

  - compute the dimensional vector $\mathbf{a}(\mathrm{Vect})$:

$$\mathbf{a}(\mathrm{Vect}) = \left(\sum_{b=1}^s \left(\mathrm{Vect}_{r+i} \cdot F(\mathbf{SK}'', \mathrm{id}, 1, b)\right) \quad \dots \quad \sum_{i=1}^s \left(\mathrm{Vect}_{r+i} \cdot F(\mathbf{SK}'', \mathrm{id}, x, b)\right)\right) \tag{14}$$

  - denote $\mathbf{y}(\mathrm{Vect})$ the x-dimensional vector:

$$\mathbf{y}(\mathrm{Vect}) = f_{\mathbf{SK}''}(\mathrm{Vect}) + \mathbf{a}(\mathrm{Vect}) \tag{15}$$

  - let $\mathbf{Mt}'$ be:

$$\mathbf{Mt}' = \begin{bmatrix} F(\mathbf{SK}'', \mathrm{id}, 1, s+1) & \cdots & F(\mathbf{SK}'', \mathrm{id}, x, s+1) \\ \vdots & \ddots & \vdots \\ F(\mathbf{SK}'', \mathrm{id}, 1, s+q) & \cdots & F(\mathbf{SK}'', \mathrm{id}, x, s+q) \end{bmatrix} = \begin{bmatrix} u'_{11} & \cdots & u'_{1x} \\ \vdots & \ddots & \vdots \\ u'_{q1} & \cdots & u'_{qx} \end{bmatrix} \tag{16}$$

- compute all the items $\mathbf{I}$:

$$\forall i = 1, \ldots, q;\ j = 1, \ldots, x: \quad I_{ij}(\mathrm{Vect}) = y_j(\mathrm{Vect}) \cdot u'_{ij} \overset{?}{=} \mathrm{tg}_{ij}(\mathrm{Vect}) \tag{17}$$
$$\overset{\text{yes}}{\rightsquigarrow} \text{pass}$$

▷ Security Analysis:

- Theorem 1: Without making any changes in the MACs, and suppose that there are m MACs and N neighbor nodes for the adversary A, the probability that the polluted data can succeeds the verification of the neighbor nodes depends on *number of keys they have* and $\leq \frac{1}{m^N}$
  - if the next hops only have the same key with the compromised node, it will treat the polluted data as a legitimate one.

> in order to get the same MAC, a node needs the same key, so a node has all m the same keys by the probability of $\frac{1}{m}$. There are N neighbor nodes in the next hop, so the overall probability is $\frac{1}{m^N}$

- Theorem 2: An adversary A which is one of the N legitimate nodes ants to provide changes in the tg. This attack can be detected by the next hop by the probability of $\frac{1}{2(c+1)}$, where c is a random coefficient.
  - In fact, the polluted $\mathbf{TAG}$ can traverse some hops, before it is detected, by this probability $\leq (1 - \frac{1}{2(c+1)})^{N-i}$ (same as Dual-HMAC)

▷ Performance Evaluation:

- Communication Overhead
- Storage Overhead

> almost say nothing about the performance evaluation in the thesis, only conclusion...

Comparison

Scheme	Verify Complexity (multiplications/packet)	Pr[DA succeed]	Pr[TA succeed]
HomMac	$(2m+n)\ell$	$\leq \frac{1}{q^d}$	100
MacSig	$\frac{3}{2}\lvert p\rvert(m+\ell+1)+(m+n+1)\ell$	$\leq \frac{1}{p^d}$	0
KEPTE	$N+m+n$	$\leq \frac{1}{p^{N-r}}$	$\frac{1}{p^d}$
D-MAC	$\ell(m+n+1)+(\ell+\ell')$	same as MacSig	$\begin{cases} \text{for MACs} & \frac{1}{p^d} \\ \text{for D-MACs} & 0 \end{cases}$
Dual-HMAC	$L(m+n+1)+L'(L+1)$	same as MacSig	$\begin{cases} \text{for MACs} & \frac{1}{p^d} \\ \text{for D-MACs} & 100 \end{cases}$
DIP MAC	$(n+m)$	$\frac{1}{q^\ell}$	100
HMAC	$\ell(m+n+1)+\ell'(\ell+1)+\frac{3}{2}\lvert p\rvert$	$\frac{1}{p}$	$\frac{1}{p}$

> Symbols used in different scheme may differ.

Attack games:

- attack games used in HomMac can run several queries (for different generation).
- attack games used in MacSig run only one query ($\equiv$ type 2 forgery in HomMac).
  - HSM game studies type 2 forgery in detail (3 cases).
- in KEPTE, the attack games consider one more case - The First Pollution Attack Behavior.
  - The Second Pollution Attack Behavior: Finding a Deceptive Data Packet coresspodding the normal attack game.

Key distribution:

- MacSig considers the situation when a good user has d safe keys, but only give the guarantee when $d = 1$.

Security analysis:

- for data pollution:
  - the case MAC generated with $\begin{pmatrix}\mathbf{u} & t\end{pmatrix} \perp \mathbf{k}$ is studied in MacSig (HSM), the security level is $\frac{1}{q}$, which is often as "negligible" in other works named (suppose only one safe key).
- for tag pollution:
  - MacSig takes MAC and Sign as a whole.
  - other schemes mostly consider MAC, DMAC, sign separately.