Complexity of Monomial Prediction in Cryptography and Machine Learning

Joint work with Pranjal Dutta (NUS) and Santanu Sarkar (IIT Madras).

Mahesh Sreekumar Rajasree

IIT Delhi

SYNASC 2024

Table of Contents

1. Monomial Prediction Problem
2. Definitions
3. Hardness result
4. Ascon and new zero sum distinguishers
5. Conclusion

Monomial Prediction Problem

Monomail prediction problem

Given a composition of **quadratic/PGC** functions $f := f_r \circ f_{r-1} \circ \ldots f_0$, and a monomial m, where each $f_i : \mathbb{F}_2^n \longrightarrow \mathbb{F}_2^n$, decide the **coefficient** of m in $f^{(1)}$.

Introduction

Monomail prediction problem

Given a composition of **quadratic/PGC** functions $f := f_r \circ f_{r-1} \circ \ldots f_0$, and a monomial m, where each $f_i : \mathbb{F}_2^n \longrightarrow \mathbb{F}_2^n$, decide the **coefficient** of m in $f^{(1)}$.

- ❑ Why quadratic?

Monomail prediction problem

Given a composition of **quadratic/PGC** functions $f := f_r \circ f_{r-1} \circ \dots f_0$, and a monomial m, where each $f_i : \mathbb{F}_2^n \longrightarrow \mathbb{F}_2^n$, decide the **coefficient** of m in $f^{(1)}$.

- Why quadratic? Almost every **symmetric key** cryptosystems are based on composition of **quadratic** functions.

Introduction

Monomail prediction problem

Given a composition of **quadratic/PGC** functions $f := f_r \circ f_{r-1} \circ \ldots f_0$, and a monomial m, where each $f_i : \mathbb{F}_2^n \longrightarrow \mathbb{F}_2^n$, decide the **coefficient** of m in $f^{(1)}$.

- Why quadratic? Almost every **symmetric key** cryptosystems are based on composition of **quadratic** functions.
- E.g. KECCAK, Trivium, Ascon, TinyJAMBU, etc.

Introduction

Monomail prediction problem

Given a composition of **quadratic/PGC** functions $f := f_r \circ f_{r-1} \circ \ldots f_0$, and a monomial m, where each $f_i : \mathbb{F}_2^n \longrightarrow \mathbb{F}_2^n$, decide the **coefficient** of m in $f^{(1)}$.

- Why quadratic? Almost every **symmetric key** cryptosystems are based on composition of **quadratic** functions.

- E.g. KECCAK, Trivium, Ascon, TinyJAMBU, etc. In-fact, in these systems, all f_i's are the same.

Monomail prediction problem

Given a composition of **quadratic/PGC** functions $f := f_r \circ f_{r-1} \circ \ldots f_0$, and a monomial m, where each $f_i : \mathbb{F}_2^n \longrightarrow \mathbb{F}_2^n$, decide the **coefficient** of m in $f^{(1)}$.

- Why quadratic? Almost every **symmetric key** cryptosystems are based on composition of **quadratic** functions.
- E.g. KECCAK, Trivium, Ascon, TinyJAMBU, etc. In-fact, in these systems, all f_i's are the same.
- Knowing the **coefficients** may lead to an attack.

Monomail prediction problem

Given a composition of **quadratic/PGC** functions $f := f_r \circ f_{r-1} \circ \ldots f_0$, and a monomial m, where each $f_i : \mathbb{F}_2^n \longrightarrow \mathbb{F}_2^n$, decide the **coefficient** of m in $f^{(1)}$.

- Why quadratic? Almost every **symmetric key** cryptosystems are based on composition of **quadratic** functions.
- E.g. KECCAK, Trivium, Ascon, TinyJAMBU, etc. In-fact, in these systems, all f_i's are the same.
- Knowing the **coefficients** may lead to an attack.
- **Cube attacks** can detect non-randomness if there are monomials missing.

Introduction

Monomail prediction problem

Given a composition of **quadratic/PGC** functions $f := f_r \circ f_{r-1} \circ \ldots f_0$, and a monomial m, where each $f_i : \mathbb{F}_2^n \longrightarrow \mathbb{F}_2^n$, decide the **coefficient** of m in $f^{(1)}$.

- Why quadratic? Almost every **symmetric key** cryptosystems are based on composition of **quadratic** functions.
- E.g. KECCAK, Trivium, Ascon, TinyJAMBU, etc. In-fact, in these systems, all f_i's are the same.
- Knowing the **coefficients** may lead to an attack.
- **Cube attacks** can detect non-randomness if there are monomials missing.
- Why PGCs?

Introduction

Monomail prediction problem

Given a composition of **quadratic/PGC** functions $f := f_r \circ f_{r-1} \circ \ldots f_0$, and a monomial m, where each $f_i : \mathbb{F}_2^n \longrightarrow \mathbb{F}_2^n$, decide the **coefficient** of m in $f^{(1)}$.

- Why quadratic? Almost every **symmetric key** cryptosystems are based on composition of **quadratic** functions.
- E.g. KECCAK, Trivium, Ascon, TinyJAMBU, etc. In-fact, in these systems, all f_i's are the same.
- Knowing the **coefficients** may lead to an attack.
- **Cube attacks** can detect non-randomness if there are monomials missing.
- Why PGCs? These are representations of multivariate probability generating polynomials (PGPs), which capture many tractable probabilistic models in machine learning.

Probability Generating Polynomials

Let $\Pr$ be a probability distribution over binary random variables $X_1, X_2, \cdots, X_n$, then the probability generating polynomial for the distribution is defined as

$$g(x_1, \ldots, x_n) = \sum_{S \subseteq \{1, \cdots, n\}} \Pr[X^S] \cdot x^S$$

where $\Pr[X^S] = \Pr[\{X_i = 1\}_{i \in S}, \{X_i = 0\}_{i \notin S}]$ and $x^S = \prod_{i \in S} x_i$

Probability Generating Polynomials

Let Pr be a probability distribution over binary random variables $X_1, X_2, \cdots, X_n$, then the probability generating polynomial for the distribution is defined as

$$g(x_1, \ldots, x_n) = \sum_{S \subseteq \{1,\cdots,n\}} \mathsf{Pr}[X^S] \cdot x^S$$

where $\mathsf{Pr}[X^S] = \mathsf{Pr}[\{X_i = 1\}_{i \in S}, \{X_i = 0\}_{i \notin S}]$ and $x^S = \prod_{i \in S} x_i$

- ❑ A circuit that generate a PGP is a PGC

Probability Generating Polynomials

Let $\Pr$ be a probability distribution over binary random variables $X_1, X_2, \cdots, X_n$, then the probability generating polynomial for the distribution is defined as

$$g(x_1, \ldots, x_n) = \sum_{S \subseteq \{1, \cdots, n\}} \Pr[X^S] \cdot x^S$$

where $\Pr[X^S] = \Pr[\{X_i = 1\}_{i \in S}, \{X_i = 0\}_{i \notin S}]$ and $x^S = \prod_{i \in S} x_i$

- A circuit that generate a PGP is a PGC where a circuit is a directed acyclic graph consisting of three types of nodes:

Probability Generating Circuits

Probability Generating Polynomials

Let $\Pr$ be a probability distribution over binary random variables $X_1, X_2, \cdots, X_n$, then the probability generating polynomial for the distribution is defined as

$$g(x_1, \ldots, x_n) = \sum_{S \subseteq \{1, \cdots, n\}} \Pr[X^S] \cdot x^S$$

where $\Pr[X^S] = \Pr[\{X_i = 1\}_{i \in S}, \{X_i = 0\}_{i \notin S}]$ and $x^S = \prod_{i \in S} x_i$

- A circuit that generate a PGP is a PGC where a circuit is a directed acyclic graph consisting of three types of nodes:

  Sum nodes + with weighted edges to children;

Probability Generating Polynomials

Let Pr be a probability distribution over binary random variables $X_1, X_2, \cdots, X_n$, then the probability generating polynomial for the distribution is defined as

$$g(x_1, \ldots, x_n) = \sum_{S \subseteq \{1, \cdots, n\}} \Pr[X^S] \cdot x^S$$

where $\Pr[X^S] = \Pr[\{X_i = 1\}_{i \in S}, \{X_i = 0\}_{i \notin S}]$ and $x^S = \prod_{i \in S} x_i$

- A circuit that generate a PGP is a PGC where a circuit is a directed acyclic graph consisting of three types of nodes:

  Sum nodes + with weighted edges to children;

  Product nodes $\times$ with unweighted edges to children;

Probability Generating Polynomials

Let Pr be a probability distribution over binary random variables $X_1, X_2, \cdots, X_n$, then the probability generating polynomial for the distribution is defined as

$$g(x_1, \dots, x_n) = \sum_{S \subseteq \{1, \cdots, n\}} \Pr[X^S] \cdot x^S$$

where $\Pr[X^S] = \Pr[\{X_i = 1\}_{i \in S}, \{X_i = 0\}_{i \notin S}]$ and $x^S = \prod_{i \in S} x_i$

- A circuit that generate a PGP is a PGC where a circuit is a directed acyclic graph consisting of three types of nodes:

  Sum nodes $+$ with weighted edges to children;

  Product nodes $\times$ with unweighted edges to children;

  Leaf nodes, which are variables x_i or constants.

Probability Generating Circuits

Probability Generating Polynomials

Let Pr be a probability distribution over binary random variables $X_1, X_2, \cdots, X_n$, then the probability generating polynomial for the distribution is defined as

$$g(x_1, \dots, x_n) = \sum_{S \subseteq \{1, \cdots, n\}} \Pr[X^S] \cdot x^S$$

where $\Pr[X^S] = \Pr[\{X_i = 1\}_{i \in S}, \{X_i = 0\}_{i \notin S}]$ and $x^S = \prod_{i \in S} x_i$

- A circuit that generate a PGP is a PGC where a circuit is a directed acyclic graph consisting of three types of nodes:

  Sum nodes $+$ with weighted edges to children;

  Product nodes $\times$ with unweighted edges to children;

  Leaf nodes, which are variables x_i or constants.

Prior works

- [Kayal, 2010] studied this problem in a generalization setting

Prior works

- [Kayal, 2010] studied this problem in a generalization setting, i.e., he considered arbitrary finite field $\mathbb{F}$.

Prior works

- [Kayal, 2010] studied this problem in a generalization setting, i.e., he considered arbitrary finite field $\mathbb{F}$.
- Showed that it is $\#P$-Complete.

Prior works

- [Kayal, 2010] studied this problem in a generalization setting, i.e., he considered arbitrary finite field $\mathbb{F}$.
- Showed that it is $\#P$-Complete.
- Studied by [Malod, 2003] in his PhD thesis.

Prior works

- [Kayal, 2010] studied this problem in a generalization setting, i.e., he considered arbitrary finite field $\mathbb{F}$.
- Showed that it is $\#P$-Complete.
- Studied by [Malod, 2003] in his PhD thesis.
- Cube testers can be used to decide existence of a monomial, but too expensive.

Prior works

- [Kayal, 2010] studied this problem in a generalization setting, i.e., he considered arbitrary finite field $\mathbb{F}$.
- Showed that it is $\#P$-Complete.
- Studied by [Malod, 2003] in his PhD thesis.
- Cube testers can be used to decide existence of a monomial, but too expensive.
- [Hu et al., 2020] presented **monomial trail** concept which decides when a monomial exists in such composition of functions.

Definitions

$\oplus\mathsf{P}$ Class

In computational complexity theory, the complexity class $\oplus\mathsf{P}$ (pronounced 'parity P') is the class of decision problems solvable by a nondeterministic Turing machine in polynomial time, where the acceptance condition is that the number of accepting computation paths is *odd*.

Complexity Classes

$\oplus$P Class

In computational complexity theory, the complexity class $\oplus$P (pronounced 'parity P') is the class of decision problems solvable by a nondeterministic Turing machine in polynomial time, where the acceptance condition is that the number of accepting computation paths is *odd*.

#P Class

The class #P is the class of function problems of the form "compute $f(x)$", where f is the number of accepting paths of a nondeterministic Turing machine running in polynomial time.

Complexity Classes

$\oplus$P Class

In computational complexity theory, the complexity class $\oplus$P (pronounced 'parity P') is the class of decision problems solvable by a nondeterministic Turing machine in polynomial time, where the acceptance condition is that the number of accepting computation paths is *odd*.

#P Class

The class #P is the class of function problems of the form "compute $f(x)$", where f is the number of accepting paths of a nondeterministic Turing machine running in polynomial time.

One can think of $\oplus$ as #P problems $\pmod 2$.

Hardness result

Theorem

Language L

Consider the following language.

Theorem

Language L

Consider the following language.

$$L := \{(f,m) \mid \mathrm{coef}_m(f_1) = 1\,, \text{ where } (f_1,\ldots,f_{n_{r+1}}) = g_r \circ g_{r-1} \circ \ldots g_0\,,$$
$$\text{and } g_i : \mathbb{F}_2^{n_i} \longrightarrow \mathbb{F}_2^{n_{i+1}}\,, n_i \in \mathbb{N}\ \forall\, i \in [r+1], \text{with } n_0 = n,$$
$$\text{monomial } m \in \mathbb{F}_2[x_1,\ldots,x_n], \text{and } \deg((g_i)_j) \leq 2\,\}\,.$$

Theorem

Language L

Consider the following language.

$$L := \{(f,m) \mid \text{coef}_m(f_1) = 1\text{, where } (f_1,\ldots,f_{n_{r+1}}) = g_r \circ g_{r-1} \circ \ldots g_0\text{,}$$
$$\text{and } g_i : \mathbb{F}_2^{n_i} \longrightarrow \mathbb{F}_2^{n_{i+1}}, n_i \in \mathbb{N}\ \forall\, i \in [r+1]\text{, with } n_0 = n\text{,}$$
$$\text{monomial } m \in \mathbb{F}_2[x_1,\ldots,x_n]\text{, and } \deg((g_i)_j) \leq 2\,\}\,.$$

- $f = (f_1,\ldots,f_{n_{r+1}})$.

Theorem

Language L

Consider the following language.

$$L := \{(f,m) \mid \mathrm{coef}_m(f_1) = 1\,,\ \text{where } (f_1,\ldots,f_{n_{r+1}}) = g_r \circ g_{r-1} \circ \ldots g_0\,,$$
$$\text{and } g_i : \mathbb{F}_2^{n_i} \longrightarrow \mathbb{F}_2^{n_{i+1}}\,, n_i \in \mathbb{N}\ \forall\, i \in [r+1], \text{with } n_0 = n,$$
$$\text{monomial } m \in \mathbb{F}_2[x_1,\ldots,x_n], \text{and } \deg((g_i)_j) \leq 2\,\}\,.$$

- $f = (f_1,\ldots,f_{n_{r+1}})$.
- g_i maps n_i bits to n_{i+1} bits.

Theorem

Language L

Consider the following language.

$$L := \{(f, m) \mid \text{coef}_m(f_1) = 1, \text{ where } (f_1, \ldots, f_{n_{r+1}}) = g_r \circ g_{r-1} \circ \ldots g_0,$$
$$\text{and } g_i : \mathbb{F}_2^{n_i} \longrightarrow \mathbb{F}_2^{n_{i+1}}, n_i \in \mathbb{N}\ \forall\, i \in [r+1], \text{with } n_0 = n,$$
$$\text{monomial } m \in \mathbb{F}_2[x_1, \ldots, x_n], \text{and } \deg((g_i)_j) \leq 2\}.$$

- $f = (f_1, \ldots, f_{n_{r+1}})$.
- g_i maps n_i bits to n_{i+1} bits.
- $(g_i)_j$'s are either constant, linear or quadratic (PGF).

Theorem

Language L

Consider the following language.

$$L := \{(f,m) \mid \mathrm{coef}_m(f_1) = 1 \text{, where } (f_1,\dots,f_{n_{r+1}}) = g_r \circ g_{r-1} \circ \dots g_0 \text{,}$$
$$\text{and } g_i : \mathbb{F}_2^{n_i} \longrightarrow \mathbb{F}_2^{n_{i+1}}, n_i \in \mathbb{N}\ \forall\, i \in [r+1], \text{with } n_0 = n,$$
$$\text{monomial } m \in \mathbb{F}_2[x_1,\dots,x_n], \text{and } \deg((g_i)_j) \leq 2\}.$$

- $f = (f_1,\dots,f_{n_{r+1}})$.
- g_i maps n_i bits to n_{i+1} bits.
- $(g_i)_j$'s are either constant, linear or quadratic (PGF).

Theorem: Hardness of monomial prediction

Given a composition of **quadratic (/PGP)** functions f and a monomial m, deciding whether $(f,m) \in L$ is $\oplus$**P**-complete (#P-complete).

Proof sketch: Hamiltonian problem

Proof sketch: Hamiltonian problem

Recall

- Hamiltonian cycle: it is a closed loop on a graph where every node (vertex) is visited *exactly* once.

Proof sketch: Hamiltonian problem

Recall

- Hamiltonian cycle: it is a closed loop on a graph where every node (vertex) is visited *exactly* once.
- Odd Hamiltonian Cycle – deciding whether a given graph $G = (V, E)$ has an odd number of Hamiltonian cycles, is $\oplus$**P**-hard.

Proof sketch: Hamiltonian problem

Recall

- Hamiltonian cycle: it is a closed loop on a graph where every node (vertex) is visited *exactly* once.
- Odd Hamiltonian Cycle – deciding whether a given graph $G = (V, E)$ has an odd number of Hamiltonian cycles, is $\oplus\mathbf{P}$-hard.
- Hamiltonian Cycle polynomial - $\mathsf{HC}_n\left(x_{1,1}, \ldots, x_{n,n}\right) = \sum_{\sigma \in S_n} \prod_{i=1}^{n} x_{i,\sigma(i)}$

Proof sketch: Hamiltonian problem

Recall

- Hamiltonian cycle: it is a closed loop on a graph where every node (vertex) is visited *exactly* once.
- Odd Hamiltonian Cycle – deciding whether a given graph $G = (V, E)$ has an odd number of Hamiltonian cycles, is $\oplus\mathbf{P}$-hard.
- Hamiltonian Cycle polynomial - $\mathrm{HC}_n(x_{1,1}, \ldots, x_{n,n}) = \sum_{\sigma \in S_n} \prod_{i=1}^{n} x_{i,\sigma(i)}$ where S_n is the symmetric group on a set of size n.

Proof sketch: Hamiltonian problem

Recall

- Hamiltonian cycle: it is a closed loop on a graph where every node (vertex) is visited *exactly* once.
- Odd Hamiltonian Cycle – deciding whether a given graph $G = (V, E)$ has an odd number of Hamiltonian cycles, is $\oplus$**P**-hard.
- Hamiltonian Cycle polynomial - $\mathsf{HC}_n(x_{1,1}, \ldots, x_{n,n}) = \sum_{\sigma \in S_n} \prod_{i=1}^{n} x_{i,\sigma(i)}$ where S_n is the symmetric group on a set of size n. If $(x_{1,1}, \ldots, x_{n,n})$ is adjacency matrix, then HC_n counts the number of Hamiltonian cycles.

Proof sketch: Hamiltonian problem

Recall

- Hamiltonian cycle: it is a closed loop on a graph where every node (vertex) is visited *exactly* once.
- Odd Hamiltonian Cycle – deciding whether a given graph $G = (V, E)$ has an odd number of Hamiltonian cycles, is $\oplus\mathbf{P}$-hard.
- Hamiltonian Cycle polynomial - $\mathsf{HC}_n\left(x_{1,1}, \ldots, x_{n,n}\right) = \sum_{\sigma \in S_n} \prod_{i=1}^{n} x_{i,\sigma(i)}$ where S_n is the symmetric group on a set of size n. If $(x_{1,1}, \ldots, x_{n,n})$ is adjacency matrix, then HC_n counts the number of Hamiltonian cycles.

We will show `Odd Hamiltonian Cycle` $\leq_p L$.

Proof sketch: Core lemma

Lemma:Composition lemma

Let $G = (V, E)$ be a given graph with the adjacency matrix $\boldsymbol{x} = (x_{i,j})_{i,j \in [n]}$.

Proof sketch: Core lemma

Lemma:Composition lemma

Let $G = (V, E)$ be a given graph with the adjacency matrix $\boldsymbol{x} = (x_{i,j})_{i,j \in [n]}$. Let $\boldsymbol{y} = (y_1, \ldots, y_n)$ and $\boldsymbol{z} = (z_1, \ldots, z_n)$ be $2n$ variables.

Proof sketch: Core lemma

Lemma:Composition lemma

Let $G = (V, E)$ be a given graph with the adjacency matrix $\boldsymbol{x} = (x_{i,j})_{i,j \in [n]}$. Let $\boldsymbol{y} = (y_1, \ldots, y_n)$ and $\boldsymbol{z} = (z_1, \ldots, z_n)$ be $2n$ variables. Then, there exist $g_0, \ldots, g_n$, polynomial maps such that

Proof sketch: Core lemma

Lemma:Composition lemma

Let $G = (V, E)$ be a given graph with the adjacency matrix $\boldsymbol{x} = (x_{i,j})_{i,j \in [n]}$. Let $\boldsymbol{y} = (y_1, \ldots, y_n)$ and $\boldsymbol{z} = (z_1, \ldots, z_n)$ be $2n$ variables. Then, there exist $g_0, \ldots, g_n$, polynomial maps such that

(i) $g_0 : \mathbb{F}_2^{n^2+2n} \longrightarrow \mathbb{F}_2^{2n^2}$, and

Proof sketch: Core lemma

Lemma:Composition lemma

Let $G = (V, E)$ be a given graph with the adjacency matrix $\boldsymbol{x} = (x_{i,j})_{i,j \in [n]}$. Let $\boldsymbol{y} = (y_1, \ldots, y_n)$ and $\boldsymbol{z} = (z_1, \ldots, z_n)$ be $2n$ variables. Then, there exist $g_0, \ldots, g_n$, polynomial maps such that

(i) $g_0 : \mathbb{F}_2^{n^2+2n} \longrightarrow \mathbb{F}_2^{2n^2}$, and $g_i : \mathbb{F}_2^{2n^2} \longrightarrow \mathbb{F}_2^{2n^2}$, for $i \in [n]$,

Proof sketch: Core lemma

Lemma:Composition lemma

Let $G = (V, E)$ be a given graph with the adjacency matrix $\boldsymbol{x} = (x_{i,j})_{i,j \in [n]}$. Let $\boldsymbol{y} = (y_1, \ldots, y_n)$ and $\boldsymbol{z} = (z_1, \ldots, z_n)$ be $2n$ variables. Then, there exist $g_0, \ldots, g_n$, polynomial maps such that

(i) $g_0 : \mathbb{F}_2^{n^2+2n} \longrightarrow \mathbb{F}_2^{2n^2}$, and $g_i : \mathbb{F}_2^{2n^2} \longrightarrow \mathbb{F}_2^{2n^2}$, for $i \in [n]$, with $\mathsf{deg}((g_i)_j) \leq 2$, and

Proof sketch: Core lemma

Lemma:Composition lemma

Let $G = (V, E)$ be a given graph with the adjacency matrix $\boldsymbol{x} = (x_{i,j})_{i,j \in [n]}$. Let $\boldsymbol{y} = (y_1, \ldots, y_n)$ and $\boldsymbol{z} = (z_1, \ldots, z_n)$ be $2n$ variables. Then, there exist $g_0, \ldots, g_n$, polynomial maps such that

(i) $g_0 : \mathbb{F}_2^{n^2+2n} \longrightarrow \mathbb{F}_2^{2n^2}$, and $g_i : \mathbb{F}_2^{2n^2} \longrightarrow \mathbb{F}_2^{2n^2}$, for $i \in [n]$, with $\mathrm{deg}((g_i)_j) \leq 2$, and

(ii) coef$_{y_1 \cdots y_n \cdot z_1 \cdots z_n}(f_1(\boldsymbol{x}, \boldsymbol{y}, \boldsymbol{z})) = \mathrm{HC}_n(\boldsymbol{x})$,

Proof sketch: Core lemma

Lemma:Composition lemma

Let $G = (V, E)$ be a given graph with the adjacency matrix $\boldsymbol{x} = (x_{i,j})_{i,j \in [n]}$. Let $\boldsymbol{y} = (y_1, \ldots, y_n)$ and $\boldsymbol{z} = (z_1, \ldots, z_n)$ be $2n$ variables. Then, there exist $g_0, \ldots, g_n$, polynomial maps such that

(i) $g_0 : \mathbb{F}_2^{n^2+2n} \longrightarrow \mathbb{F}_2^{2n^2}$, and $g_i : \mathbb{F}_2^{2n^2} \longrightarrow \mathbb{F}_2^{2n^2}$, for $i \in [n]$, with $\deg((g_i)_j) \leq 2$, and

(ii) $\mathrm{coef}_{y_1 \cdots y_n \cdot z_1 \cdots z_n}(f_1(\boldsymbol{x}, \boldsymbol{y}, \boldsymbol{z})) = \mathrm{HC}_n(\boldsymbol{x})$,where $(f_1, \ldots, f_{2n^2}) = g_n \circ \ldots \circ g_0$.

Proof sketch: The polynomials

$$(g_0(\boldsymbol{x}, \boldsymbol{y}, \boldsymbol{z}))_k := \begin{cases} x_{i,j}, & \text{when } k \leq n^2, \text{where } k-1 = (i-1) + n(j-1), \\ y_i \cdot z_j, & \text{when } n^2 < k \leq 2n^2, \text{where } k-1-n^2 = (i-1) + n(j-1). \end{cases}$$

Proof sketch: The polynomials

$$(g_0(\boldsymbol{x},\boldsymbol{y},\boldsymbol{z}))_k := \begin{cases} x_{i,j}, & \text{when } k \le n^2, \text{where } k-1 = (i-1)+n(j-1), \\ y_i \cdot z_j, & \text{when } n^2 < k \le 2n^2, \text{where } k-1-n^2 = (i-1)+n(j-1). \end{cases}$$

$$(g_1(\boldsymbol{w},\boldsymbol{s}))_k := \begin{cases} w_{i,j} \cdot s_{i,j}, & \text{when } k \le n^2, \text{where } k-1 = (i-1)+n(j-1), \\ (g_1(\boldsymbol{w},\boldsymbol{s}))_{k-n^2}, & \text{when } n^2 < k \le 2n^2. \end{cases}$$

Proof sketch: The polynomials

$$(g_0(\boldsymbol{x},\boldsymbol{y},\boldsymbol{z}))_k := \begin{cases} x_{i,j}, & \text{when } k \le n^2, \text{where } k-1 = (i-1)+n(j-1), \\ y_i \cdot z_j, & \text{when } n^2 < k \le 2n^2, \text{where } k-1-n^2 = (i-1)+n(j-1). \end{cases}$$

$$(g_1(\boldsymbol{w},\boldsymbol{s}))_k := \begin{cases} w_{i,j} \cdot s_{i,j}, & \text{when } k \le n^2, \text{where } k-1 = (i-1)+n(j-1), \\ (g_1(\boldsymbol{w},\boldsymbol{s}))_{k-n^2}, & \text{when } n^2 < k \le 2n^2. \end{cases}$$

$$(g_\ell(\boldsymbol{w},\boldsymbol{s}))_k := \begin{cases} \sum_{r=1}^{n} w_{i,r} \cdot s_{r,j}, & \text{when } k \le n^2, \text{where } k-1 = (i-1)+n(j-1), \\ s_{i,j}, & \text{when } n^2 < k \le 2n^2, \text{where } k-1-n^2 = (i-1)+n(j-1). \end{cases}$$

Proof sketch: The polynomials

$$(g_0(\boldsymbol{x},\boldsymbol{y},\boldsymbol{z}))_k := \begin{cases} x_{i,j}, & \text{when } k \le n^2, \text{where } k-1 = (i-1)+n(j-1), \\ y_i \cdot z_j, & \text{when } n^2 < k \le 2n^2, \text{where } k-1-n^2 = (i-1)+n(j-1). \end{cases}$$

$$(g_1(\boldsymbol{w},\boldsymbol{s}))_k := \begin{cases} w_{i,j} \cdot s_{i,j}, & \text{when } k \le n^2, \text{where } k-1 = (i-1)+n(j-1), \\ (g_1(\boldsymbol{w},\boldsymbol{s}))_{k-n^2}, & \text{when } n^2 < k \le 2n^2. \end{cases}$$

$$(g_\ell(\boldsymbol{w},\boldsymbol{s}))_k := \begin{cases} \sum_{r=1}^{n} w_{i,r} \cdot s_{r,j}, & \text{when } k \le n^2, \text{where } k-1 = (i-1)+n(j-1), \\ s_{i,j}, & \text{when } n^2 < k \le 2n^2, \text{where } k-1-n^2 = (i-1)+n(j-1). \end{cases}$$

Proof sketch

Claim 1

For any $\ell \geq 1$,

$$(g_\ell(\ldots(g_0(\boldsymbol{x}, \boldsymbol{y}, \boldsymbol{z})\ldots)_k = x_{i,j} \cdot y_i \cdot z_j$$

for $k \in [n^2+1, 2n^2]$, where $k - 1 - n^2 = (i-1) + n(j-1)$.

Proof sketch

Claim 1

For any $\ell \geq 1$,

$$(g_\ell(\ldots(g_0(\boldsymbol{x}, \boldsymbol{y}, \boldsymbol{z})\ldots)_k = x_{i,j} \cdot y_i \cdot z_j$$

for $k \in [n^2+1, 2n^2]$, where $k - 1 - n^2 = (i-1) + n(j-1)$.

Let us prove for $\ell = 1$, i.e., $g_1(g_0(\boldsymbol{x}, \boldsymbol{y}, \boldsymbol{z}))$.

Proof sketch

Claim 1

For any $\ell \geq 1$,

$$(g_\ell(\ldots(g_0(\boldsymbol{x},\boldsymbol{y},\boldsymbol{z})\ldots)_k = x_{i,j} \cdot y_i \cdot z_j$$

for $k \in [n^2+1, 2n^2]$, where $k-1-n^2 = (i-1)+n(j-1)$.

Let us prove for $\ell = 1$, i.e., $g_1(g_0(\boldsymbol{x},\boldsymbol{y},\boldsymbol{z}))$.

$$(g_0(\boldsymbol{x},\boldsymbol{y},\boldsymbol{z}))_k := \begin{cases} x_{i,j}, & \text{when } k \leq n^2, \text{where } k-1 = (i-1)+n(j-1), \\ y_i \cdot z_j, & \text{when } n^2 < k \leq 2n^2, \text{where } k-1-n^2 = (i-1)+n(j-1). \end{cases}$$

Proof sketch

Claim 1

For any $\ell \geq 1$,

$$(g_\ell(\ldots(g_0(\boldsymbol{x},\boldsymbol{y},\boldsymbol{z})\ldots)_k = x_{i,j} \cdot y_i \cdot z_j$$

for $k \in [n^2+1, 2n^2]$, where $k-1-n^2 = (i-1)+n(j-1)$.

Let us prove for $\ell = 1$, i.e., $g_1(g_0(\boldsymbol{x},\boldsymbol{y},\boldsymbol{z}))$.

$$(g_0(\boldsymbol{x},\boldsymbol{y},\boldsymbol{z}))_k := \begin{cases} x_{i,j}, & \text{when } k \leq n^2, \text{ where } k-1 = (i-1)+n(j-1), \\ y_i \cdot z_j, & \text{when } n^2 < k \leq 2n^2, \text{ where } k-1-n^2 = (i-1)+n(j-1). \end{cases}$$

$$(g_1(\boldsymbol{w},\boldsymbol{s}))_k := \begin{cases} w_{i,j} \cdot s_{i,j}, & \text{when } k \leq n^2, \text{ where } k-1 = (i-1)+n(j-1), \\ (g_1(\boldsymbol{w},\boldsymbol{s}))_{k-n^2}, & \text{when } n^2 < k \leq 2n^2. \end{cases}$$

Proof sketch

Claim 1

For any $\ell \geq 1$,

$$(g_\ell(\ldots(g_0(\boldsymbol{x},\boldsymbol{y},\boldsymbol{z})\ldots)_k = x_{i,j}\cdot y_i\cdot z_j$$

for $k \in [n^2+1, 2n^2]$, where $k-1-n^2 = (i-1)+n(j-1)$.

Let us prove for $\ell = 1$, i.e., $g_1(g_0(\boldsymbol{x},\boldsymbol{y},\boldsymbol{z}))$.

$$(g_0(\boldsymbol{x},\boldsymbol{y},\boldsymbol{z}))_k := \begin{cases} x_{i,j}, & \text{when } k \leq n^2, \text{where } k-1 = (i-1)+n(j-1), \\ y_i\cdot z_j, & \text{when } n^2 < k \leq 2n^2, \text{where } k-1-n^2 = (i-1)+n(j-1). \end{cases}$$

$$(g_1(\boldsymbol{w},\boldsymbol{s}))_k := \begin{cases} w_{i,j}\cdot s_{i,j}, & \text{when } k \leq n^2, \text{where } k-1 = (i-1)+n(j-1), \\ (g_1(\boldsymbol{w},\boldsymbol{s}))_{k-n^2}, & \text{when } n^2 < k \leq 2n^2. \end{cases}$$

For $\ell > 1$, observe that g_ℓ is an identity map in the last n^2 coordinates.

Proof sketch

Claim 2

For any $\ell \geq 2$ and $k \in [n^2]$,

$$
\begin{aligned}
&(g_\ell(\ldots(g_0(\boldsymbol{x},\boldsymbol{y},\boldsymbol{z})\ldots))_k \\
&= y_i z_j \cdot \sum_{1 \leq m_1,\ldots,m_{\ell-1} \leq n} x_{i,m_1} x_{m_1,m_2} \cdots x_{m_{\ell-2},m_{\ell-1}} x_{m_{\ell-1},j} \cdot \left(\prod_{s=1}^{\ell-1} y_{m_s} z_{m_s} \right) .
\end{aligned}
$$

Proof sketch

Claim 2

For any $\ell \geq 2$ and $k \in [n^2]$,

$$(g_\ell(\ldots(g_0(\boldsymbol{x}, \boldsymbol{y}, \boldsymbol{z})\ldots))_k$$

$$= y_i z_j \cdot \sum_{1 \leq m_1, \ldots, m_{\ell-1} \leq n} x_{i,m_1} x_{m_1,m_2} \cdots x_{m_{\ell-2},m_{\ell-1}} x_{m_{\ell-1},j} \cdot \left(\prod_{s=1}^{\ell-1} y_{m_s} z_{m_s} \right) .$$

Claim 2 with $k = 1$ (i.e. $i = j = 1$) and $\ell = n$, gives the following identity:

Proof sketch

Claim 2

For any $\ell \geq 2$ and $k \in [n^2]$,

$$(g_\ell(\ldots(g_0(\boldsymbol{x},\boldsymbol{y},\boldsymbol{z})\ldots))_k$$
$$= y_i z_j \cdot \sum_{1 \leq m_1,\ldots,m_{\ell-1} \leq n} x_{i,m_1} x_{m_1,m_2} \cdots x_{m_{\ell-2},m_{\ell-1}} x_{m_{\ell-1},j} \cdot \left(\prod_{s=1}^{\ell-1} y_{m_s} z_{m_s} \right) .$$

Claim 2 with $k = 1$ (i.e. $i = j = 1$) and $\ell = n$, gives the following identity:

$$(g_n(\ldots(g_0(\boldsymbol{x},\boldsymbol{y},\boldsymbol{z})\ldots)_1$$
$$= y_1 z_1 \cdot \sum_{1 \leq m_1,\ldots,m_{n-1} \leq n} x_{1,m_1} x_{m_1,m_2} \cdots x_{m_{n-2},m_{n-1}} x_{m_{n-1},1} \cdot \left(\prod_{s=1}^{n-1} y_{m_s} z_{m_s} \right)$$
$$= \left(\prod_{s=1}^{n} y_{m_s} z_{m_s} \right) \cdot \sum_{1 \leq m_1,\ldots,m_{n-1} \leq n} x_{1,m_1} x_{m_1,m_2} \cdots x_{m_{n-2},m_{n-1}} x_{m_{n-1},1} .$$

Proof sketch

Claim 2

For any $\ell \geq 2$ and $k \in [n^2]$,

$$\begin{aligned}&(g_\ell(\ldots(g_0(\boldsymbol{x},\boldsymbol{y},\boldsymbol{z})\ldots))_k \\ &= y_i z_j \cdot \sum_{1\leq m_1,\ldots,m_{\ell-1}\leq n} x_{i,m_1}x_{m_1,m_2}\cdots x_{m_{\ell-2},m_{\ell-1}}x_{m_{\ell-1},j} \cdot \left(\prod_{s=1}^{\ell-1} y_{m_s}z_{m_s}\right).\end{aligned}$$

Claim 2 with $k = 1$ (i.e. $i = j = 1$) and $\ell = n$, gives the following identity:

$$\begin{aligned}&(g_n(\ldots(g_0(\boldsymbol{x},\boldsymbol{y},\boldsymbol{z})\ldots)_1 \\ &= y_1 z_1 \cdot \sum_{1\leq m_1,\ldots,m_{n-1}\leq n} x_{1,m_1}x_{m_1,m_2}\cdots x_{m_{n-2},m_{n-1}}x_{m_{n-1},1} \cdot \left(\prod_{s=1}^{n-1} y_{m_s}z_{m_s}\right) \\ &= \left(\prod_{s=1}^{n} y_{m_s}z_{m_s}\right)\cdot \sum_{1\leq m_1,\ldots,m_{n-1}\leq n} x_{1,m_1}x_{m_1,m_2}\cdots x_{m_{n-2},m_{n-1}}x_{m_{n-1},1}.\end{aligned}$$

Ascon and new zero sum distinguishers

Ascon specification

- Ascon is a permutation-based family of authenticated encryption with associated data algorithms (AEAD).

Ascon specification

- Ascon is a permutation-based family of authenticated encryption with associated data algorithms (AEAD).
- It is the first choice for lightweight applications in the CAESAR competition and the NIST lightweight cryptography standardization.

Ascon specification

- Ascon is a permutation-based family of authenticated encryption with associated data algorithms (AEAD).
- It is the first choice for lightweight applications in the CAESAR competition and the NIST lightweight cryptography standardization.
- The core permutation p of Ascon is based on substitution permutation network (SPN) design paradigm.

- Ascon is a permutation-based family of authenticated encryption with associated data algorithms (AEAD).
- It is the first choice for lightweight applications in the CAESAR competition and the NIST lightweight cryptography standardization.
- The core permutation p of Ascon is based on substitution permutation network (SPN) design paradigm.
- It operates on a 320-bit state arranged into five 64-bit words and is defined as $p : p_L \circ p_S \circ p_C$.

p_C function

Addition of constants (p_C). We add an 8-bit constant to the bits $56, \cdots, 63$ of word X_2 at each round.

Substitution layer (p_S). We apply a 5-bit Sbox on each of the 64 columns. Let $(x_0, x_1, x_2, x_3, x_4)$ and $(y_0, y_1, y_2, y_3, y_4)$ denote the input and output of the Sbox, respectively.

Substitution layer (p_S). We apply a 5-bit Sbox on each of the 64 columns. Let $(x_0, x_1, x_2, x_3, x_4)$ and $(y_0, y_1, y_2, y_3, y_4)$ denote the input and output of the Sbox, respectively.

$$\begin{cases} y_0 = x_4x_1 + x_3 + x_2x_1 + x_2 + x_1x_0 + x_1 + x_0 \\ y_1 = x_4 + x_3x_2 + x_3x_1 + x_3 + x_2x_1 + x_2 + x_1 + x_0 \\ y_2 = x_4x_3 + x_4 + x_2 + x_1 + 1 \\ y_3 = x_4x_0 + x_4 + x_3x_0 + x_3 + x_2 + x_1 + x_0 \\ y_4 = x_4x_1 + x_4 + x_3 + x_1x_0 + x_1 \end{cases} \tag{1}$$

p_L function

Linear diffusion layer (p_L). Each 64-bit word is updated by a linear operation Σ_i which is defined below.

p_L function

Linear diffusion layer (p_L). Each 64-bit word is updated by a linear operation Σ_i which is defined below.

$$\begin{cases} X_0 \leftarrow \Sigma_0(Y_0) = Y_0 + (Y_0 \ggg 19) + (Y_0 \ggg 28) \\ X_1 \leftarrow \Sigma_1(Y_1) = Y_1 + (Y_1 \ggg 61) + (Y_1 \ggg 39) \\ X_2 \leftarrow \Sigma_2(Y_2) = Y_2 + (Y_2 \ggg 1) + (Y_2 \ggg 6) \\ X_3 \leftarrow \Sigma_3(Y_3) = Y_3 + (Y_3 \ggg 10) + (Y_3 \ggg 17) \\ X_4 \leftarrow \Sigma_4(Y_4) = Y_4 + (Y_4 \ggg 7) + (Y_4 \ggg 41) \end{cases} \tag{2}$$

New zero sum distinguishers

The state at the input of r-th round is denoted by $X_0^r \| X_1^r \| X_2^r \| X_3^r \| X_4^r$.

New zero sum distinguishers

The state at the input of r-th round is denoted by $X_0^r \| X_1^r \| X_2^r \| X_3^r \| X_4^r$. We first gave a new zero sum distinguisher for 5 rounds with complexity 2^{15} by finding a monomial that was missing from the output polynomial.

The state at the input of r-th round is denoted by $X_0^r \| X_1^r \| X_2^r \| X_3^r \| X_4^r$. We first gave a new zero sum distinguisher for 5 rounds with complexity 2^{15} by finding a monomial that was missing from the output polynomial.

Rounds	Cube size	Cube indices ($X_3^0 = X_4^0$)	Output indices (X_0^5)
5	13	0, 1, 2, 3, 4, 5, 7, 8, 10, 11, 12, 13, 16	4
		0, 1, 2, 3, 5, 6, 7, 8, 10, 11, 12, 13, 16	4
		0, 1, 2, 4, 5, 6, 7, 8, 10, 11, 12, 13, 16	4
5	14	0, 1, 2, 3, 4, 5, 6, 7, 8, 10, 11, 12, 13, 14	1, 4
		0, 1, 2, 3, 4, 5, 6, 7, 8, 10, 11, 12, 13, 16	4, 15, 24, 36
		0, 1, 2, 3, 4, 5, 6, 7, 8, 10, 11, 12, 13, 18	4

Table 1: List of cubes for 5-round Ascon-128

Conclusion

Concluding remarks

Concluding remarks

- Can we extend our theorem to composition of bounded degree functions?

Concluding remarks

- Can we extend our theorem to composition of bounded degree functions?
- Is it possible to model an MILP to find whether a monomial is missing?

Concluding remarks

- Can we extend our theorem to composition of bounded degree functions?
- Is it possible to model an MILP to find whether a monomial is missing?

Thank you. Questions?