Characterizing Collision and Second-Preimage Resistance in Linicrypt

Ian McQuoid   Trevor Swope   Mike Rosulek

Oregon State University

An Introduction

$$
\begin{aligned}
&\mathcal{P}^H(v_1, v_2, v_3): \\
&v_4 := H(1, v_2) \\
&v_5 := H(2, v_2) \\
&v_6 := H(3, v_1 + v_4) \\
&v_7 := H(4, v_3 + v_5) \\
&v_8 := H(5, v_6 + v_7) \\
&return(v_1, v_8)
\end{aligned}
$$

$$
\begin{aligned}
&\mathcal{P}^H(v_1, v_2, v_3): \\
&v_4 := H(1, v_2) \\
&v_5 := H(2, v_2) \\
&v_6 := H(3, v_1 + v_4) \\
&v_7 := H(4, v_3 + v_5) \\
&v_8 := H(5, v_6 + v_7) \\
&return(v_8)
\end{aligned}
$$

Which program is secure?

What is Linicrypt?

Linicrypt programs are a class of algorithms

Introduced by Carmer and Rosulek, Crypto 2016

Operations:

- **Sampling** uniformly from a field.
- **Querying** a random oracle over the field.
- **Performing a fixed linear combination** on field elements.

Sometimes to be explicit we write: $\mathcal{P}^H$

Useful for algebraically analyzing those algorithms.

What can we do?

$$\mathcal{P}^H(v_1, v_2, v_3):$$

$$v_4 := H(v_1)$$

$$v_5 := H(v_3)$$

$$v_6 := v_4 + v_5 + v_2$$

$$v_7 := H(v_6)$$

$$v_8 := v_7 + v_1$$

$$return(v_8, v_5)$$

What can we do?

Take field elements as input $\mathcal{P}^H(v_1, v_2, v_3)$:

$$v_4 := H(1; v_1)$$

$$v_5 := H(2; v_3)$$

$$v_6 := v_4 + v_5 + v_2$$

$$v_7 := H(3; v_6)$$

$$v_8 := v_7 + v_1$$

$$return(v_8, v_5)$$

What can we do?

Take field elements as input

$$\mathcal{P}^H(v_1, v_2, v_3) :$$

$$v_4 := H(1; v_1)$$

Query the random oracle

$$v_5 := H(2; v_3)$$

$$v_6 := v_4 + v_5 + v_2$$

$$v_7 := H(3; v_6)$$

$$v_8 := v_7 + v_1$$

$$return(v_8, v_5)$$

What can we do?

Take field elements as input	$\mathcal{P}^H(v_1, v_2, v_3):$
	$v_4 := H(1; v_1)$
Query the random oracle	$v_5 := H(2; v_3)$
Use a fixed linear combination	$v_6 := v_4 + v_5 + v_2$
	$v_7 := H(3; v_6)$
	$v_8 := v_7 + v_1$
	$return(v_8, v_5)$

What can we do?

Take field elements as input

$$\mathcal{P}^H(v_1, v_2, v_3):$$

$$v_4 := H(1; v_1)$$

Query the random oracle

$$v_5 := H(2; v_3)$$

Use a fixed linear combination

$$v_6 := v_4 + v_5 + v_2$$

$$v_7 := H(3; v_6)$$

$$v_8 := v_7 + v_1$$

Return any number of elements

$$return(v_8, v_5)$$

Modeling Linicrypt Programs

Algorithmically

$$\mathcal{P}^H(v_1, v_2, v_3):$$

$$v_4 := H(v_1)$$

$$v_5 := H(v_3)$$

$$v_6 := v_4 + v_5 + v_2$$

$$v_7 := H(v_6)$$

$$v_8 := v_7 + v_1$$

$$return(v_8, v_5)$$

Modeling Linicrypt Programs

Algorithmically

$$\mathcal{P}^H(v_1, v_2, v_3):$$

$$v_4 := H(v_1)$$

$$v_5 := H(v_3)$$

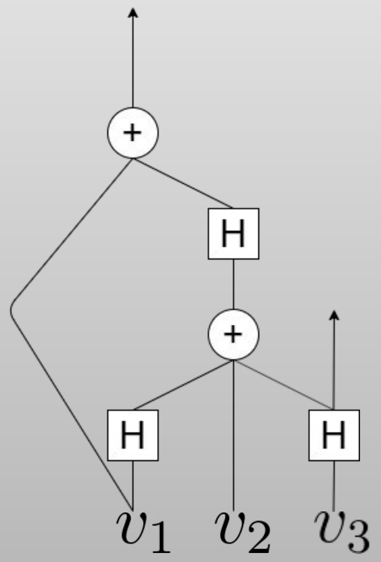
$$v_6 := v_4 + v_5 + v_2$$

$$v_7 := H(v_6)$$

$$v_8 := v_7 + v_1$$

$$return(v_8, v_5)$$

Graphically

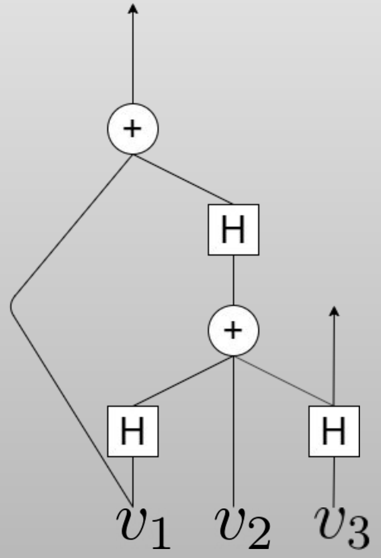


Modeling Linicrypt Programs

Algorithmically

$$\mathcal{P}^H(v_1, v_2, v_3):$$
$$v_4 := H(v_1)$$
$$v_5 := H(v_3)$$
$$v_6 := v_4 + v_5 + v_2$$
$$v_7 := H(v_6)$$
$$v_8 := v_7 + v_1$$
$$return(v_8, v_5)$$

Graphically



Algebraically

$$\boldsymbol{M} = \begin{bmatrix} 1 & 0 & 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 & 1 & 0 \end{bmatrix};$$

$$\mathcal{C} = \left\{ \begin{array}{l} \big([1\ 0\ 0\ 0\ 0\ 0], [0\ 0\ 0\ 1\ 0\ 0]\big), \\ \big([0\ 0\ 1\ 0\ 0\ 0], [0\ 0\ 0\ 0\ 1\ 0]\big), \\ \big([0\ 1\ 0\ 1\ 1\ 0], [0\ 0\ 0\ 0\ 0\ 1]\big) \end{array} \right\}$$

Informal Main Theorem

When is this class of linicrypt programs not resistant to collisions/second preimages?

Characterizable by algebraic properties!

Informal Main Theorem

When is this class of linicrypt programs not resistant to collisions/second preimages?

Characterizable by algebraic properties!

Corollary:
Second preimage resistance and collision resistance are the same (asymptotically)

Second Preimages in Linicrypt

$$(\boldsymbol{x} \neq \boldsymbol{x}') \wedge (\mathcal{P}^H(\boldsymbol{x}) = \mathcal{P}^H(\boldsymbol{x}'))$$

Second Preimages in Linicrypt

$$\boxed{(\boldsymbol{x} \neq \boldsymbol{x}')} \wedge (\mathcal{P}^H(\boldsymbol{x}) = \mathcal{P}^H(\boldsymbol{x}'))$$

1. The set of input base variables are different

Second Preimages in Linicrypt

$$(\boldsymbol{x} \neq \boldsymbol{x}') \wedge \boxed{(\mathcal{P}^H(\boldsymbol{x}) = \mathcal{P}^H(\boldsymbol{x}'))}$$

1. The set of input base variable are different
2. The outputs are the same

Collisions in Linicrypt

$$\mathcal{P}^H(v_1, v_2, v_3):$$

$$v_4 := H(v_1)$$

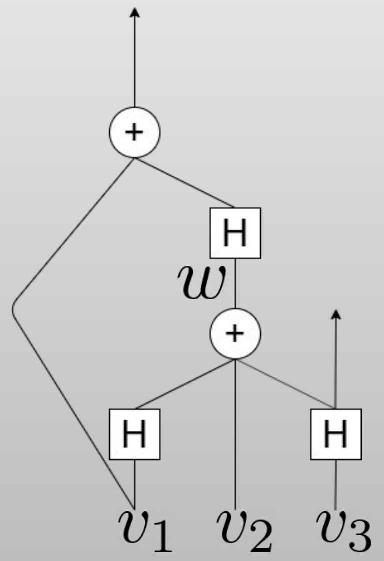
$$v_5 := H(v_3)$$

$$v_6 := v_4 + v_5 + v_2$$

$$v_7 := H(v_6)$$

$$v_8 := v_7 + v_1$$

$$return(v_8, v_5)$$



Collisions in Linicrypt

$$\mathcal{P}^H(v_1, v_2, v_3):$$

$$v_4 := H(v_1)$$

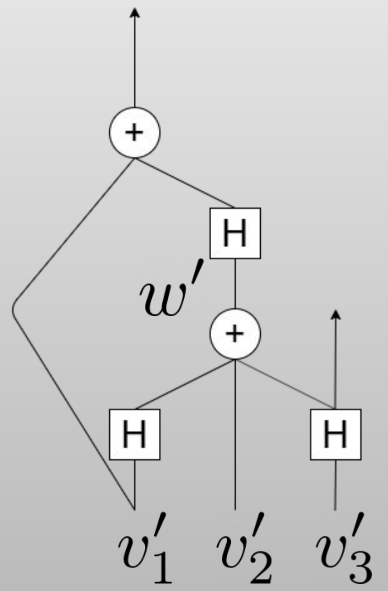
$$v_5 := H(v_3)$$

$$v_6 := v_4 + v_5 + v_2$$

$$v_7 := H(v_6)$$

$$v_8 := v_7 + v_1$$

$$return(v_8, v_5)$$



Collisions in Linicrypt

$$\mathcal{P}^H(v_1, v_2, \boxed{v_3}):$$

$$v_4 := H(v_1)$$

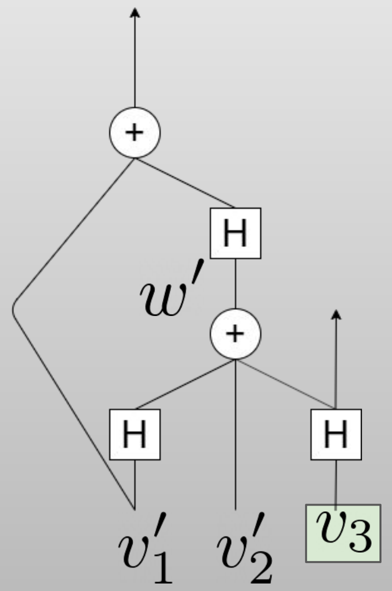
$$\boxed{v_5 := H(v_3)}$$

$$v_6 := v_4 + v_5 + v_2$$

$$v_7 := H(v_6)$$

$$v_8 := v_7 + v_1$$

$$return(v_8, \boxed{v_5})$$



Collisions in Linicrypt

$$\mathcal{P}^H(v_1, v_2, \boxed{v_3}):$$

$$\boxed{v_4 := H(v_1)}$$

Correction — plain:

$$v_4 := H(v_1)$$

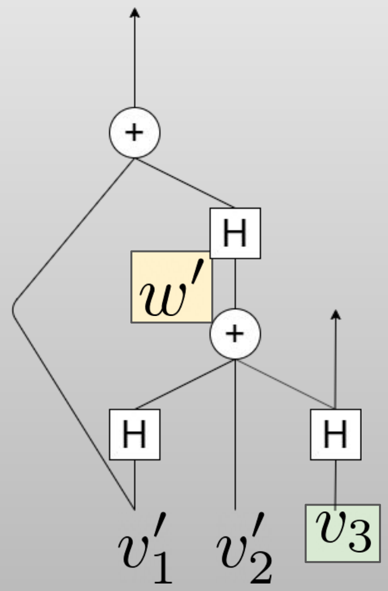
$$\boxed{v_5 := H(v_3)}$$

$$\boxed{v_6 := v_4 + v_5 + v_2}$$

$$v_7 := H(v_6)$$

$$v_8 := v_7 + v_1$$

$$return(v_8, \boxed{v_5})$$



Collisions in Linicrypt

$$\mathcal{P}^H(v_1, v_2, \boxed{v_3}):$$

$$v_4 := H(v_1)$$

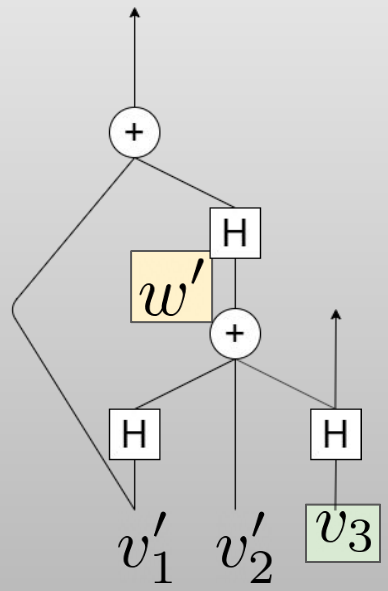
$$\boxed{v_5 := H(v_3)}$$

$$\boxed{v_6 := v_4 + v_5 + v_2}$$

$$\boxed{v_7 := H(v_6)}$$

$$v_8 := \boxed{v_7} + v_1$$

$$return(v_8, \boxed{v_5})$$



Collisions in Linicrypt

$$\mathcal{P}^H(\boxed{v_1}, v_2, \boxed{v_3}) :$$

$$\boxed{v_4 := H(v_1)}$$

$$\boxed{v_5 := H(v_3)}$$

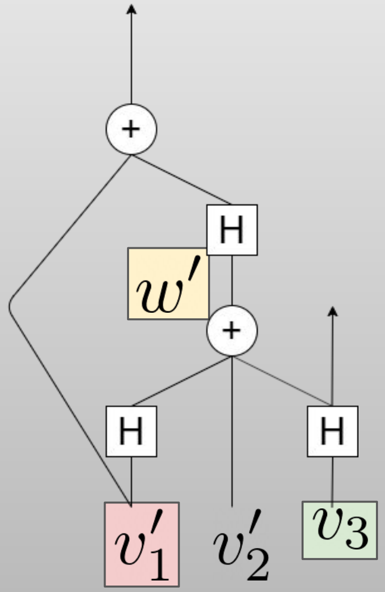
$$\boxed{v_6 := v_4 + v_5 + v_2}$$

$$\boxed{v_7 := H(v_6)}$$

$$v_8 := \boxed{v_7} + \boxed{v_1}$$

$$return(v_8, \boxed{v_5})$$

$$\boxed{v_1' \neq v_1}$$



Collisions in Linicrypt

$\mathcal{P}^H(v_1, v_2, v_3)$:

$v_4 := H(v_1)$

$v_5 := H(v_3)$

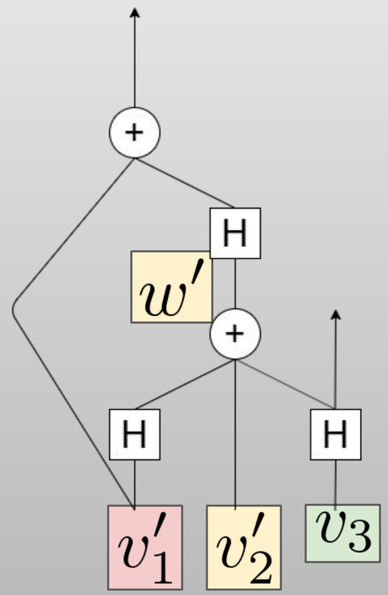
$v_6 := v_4 + v_5 + v_2$

$v_7 := H(v_6)$

$v_8 := v_7 + v_1$

$return(v_8, v_5)$

$v_1' \neq v_1$

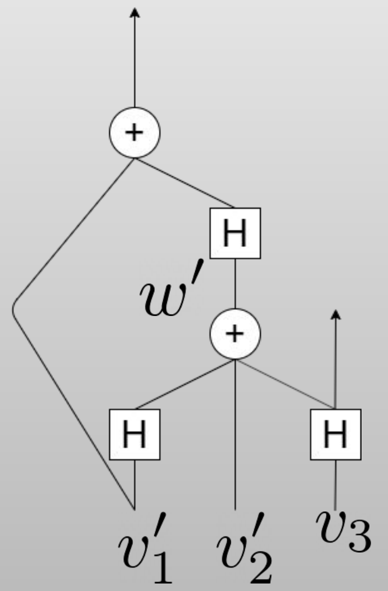


$$(\mathcal{P}^H(\boldsymbol{x}) = \mathcal{P}^H(\boldsymbol{x}'))$$

Collisions in Linicrypt

$$\mathcal{P}^H(v_1, v_2, v_3):$$
$$v_4 := H(v_1)$$
$$v_5 := H(v_3)$$
$$v_6 := v_4 + v_5 + v_2$$
$$v_7 := H(v_6)$$
$$v_8 := v_7 + v_1$$
$$return(v_8, v_5)$$

$$\boxed{v_1' \neq v_1}$$



$$\boxed{(\mathcal{P}^H(\boldsymbol{x}) = \mathcal{P}^H(\boldsymbol{x}'))}$$

Finding Collisions

1. Identify oracle queries that are the same between runs

$$v_3$$

2. Identify an oracle query that is different

$$w'$$

3. Solve backwards using linear algebra until all queries are defined

$$v_1' \; v_2'$$

Theorem Statement

For a linicrypt program with distinct nonces:

Lack of collision resistance **iff** there exists a collision structure (or degeneracy)

Existence of a collision structure **iff** no second preimage resistance

Algebraic Representation

$$\boldsymbol{M} = \begin{bmatrix} 1 & 0 & 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 & 1 & 0 \end{bmatrix};$$

$$\mathcal{C} = \left\{ \begin{array}{l} \big([1\ 0\ 0\ 0\ 0\ 0], [0\ 0\ 0\ 1\ 0\ 0]\big), \\ \big([0\ 0\ 1\ 0\ 0\ 0], [0\ 0\ 0\ 0\ 1\ 0]\big), \\ \big([0\ 1\ 0\ 1\ 1\ 0], [0\ 0\ 0\ 0\ 0\ 1]\big) \end{array} \right\}$$

$\boldsymbol{M}$ holds the returned vectors from a program

Algebraic Representation

$$\boldsymbol{M} = \begin{bmatrix} \boxed{1 \quad 0 \quad 0 \quad 0 \quad 0 \quad 1} \\ \boxed{0 \quad 0 \quad 0 \quad 0 \quad 1 \quad 0} \end{bmatrix};$$

$$\mathcal{C} = \left\{ \begin{array}{l} \big([1\ 0\ 0\ 0\ 0\ 0], [0\ 0\ 0\ 1\ 0\ 0]\big), \\ \big([0\ 0\ 1\ 0\ 0\ 0], [0\ 0\ 0\ 0\ 1\ 0]\big), \\ \big([0\ 1\ 0\ 1\ 1\ 0], [0\ 0\ 0\ 0\ 0\ 1]\big) \end{array} \right\}$$

$\boldsymbol{M}$ holds the returned vectors from a program

$$return(\boxed{v_7 + v_1}, \boxed{v_5})$$

Algebraic Representation

$$\boldsymbol{M} = \begin{bmatrix} 1 & 0 & 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 & 1 & 0 \end{bmatrix};$$

$$\mathcal{C} = \left\{ \begin{array}{c} \big([1\ 0\ 0\ 0\ 0\ 0], [0\ 0\ 0\ 1\ 0\ 0]\big), \\ \big([0\ 0\ 1\ 0\ 0\ 0], [0\ 0\ 0\ 0\ 1\ 0]\big), \\ \big([0\ 1\ 0\ 1\ 1\ 0], [0\ 0\ 0\ 0\ 0\ 1]\big) \end{array} \right\}$$

$\mathcal{C}$ holds the oracle queries a program makes

Algebraic Representation

$$\boldsymbol{M} = \begin{bmatrix} 1 & 0 & 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 & 1 & 0 \end{bmatrix};$$

$$\mathcal{C} = \left\{ \begin{array}{l} \Big([1\ 0\ 0\ 0\ 0\ 0], [0\ 0\ 0\ 1\ 0\ 0]\Big), \\ \Big([0\ 0\ 1\ 0\ 0\ 0], [0\ 0\ 0\ 0\ 1\ 0]\Big), \\ \Big([0\ 1\ 0\ 1\ 1\ 0], [0\ 0\ 0\ 0\ 0\ 1]\Big) \end{array} \right\}$$

Each query is written (t, q, a) where:
t is our nonce, q is our input vector
and a is the oracle result

Algebraic Representation

$$\boldsymbol{M} = \begin{bmatrix} 1 & 0 & 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 & 1 & 0 \end{bmatrix};$$

$$\mathcal{C} = \left\{ \begin{array}{l} \boxed{\left(\boxed{[1\ 0\ 0\ 0\ 0\ 0]}, \boxed{[0\ 0\ 0\ 1\ 0\ 0]}\right)}, \\ \left([0\ 0\ 1\ 0\ 0\ 0], [0\ 0\ 0\ 0\ 1\ 0]\right), \\ \left([0\ 1\ 0\ 1\ 1\ 0], [0\ 0\ 0\ 0\ 0\ 1]\right) \end{array} \right\}$$

This query corresponds to $\boxed{v_4} := \boxed{H(v_1)}$

What is a collision structure?

Let $\mathcal{P} = (\mathbf{M}, \mathcal{C})$ be a Linicrypt program.
A **collision structure** for $\mathcal{P}$ is a tuple $(i^*; c_1, \ldots, c_n)$, where:

What is a collision structure?

Let $\mathcal{P} = (\mathbf{M}, \mathcal{C})$ be a Linicrypt program.
A **collision structure** for $\mathcal{P}$ is a tuple $(i^*; c_1, \ldots, c_n)$, where:

1. $c_1, \ldots, c_n$ is an ordering of $\mathcal{C}$

What is a collision structure?

Let $\mathcal{P} = (\mathbf{M}, \mathcal{C})$ be a Linicrypt program.
A **collision structure** for $\mathcal{P}$ is a tuple $(i^*; c_1, \ldots, c_n)$, where:

1. $c_1, \ldots, c_n$ is an ordering of $\mathcal{C}$

2. c_{i^*} is not determined already by the queries before it

What is a collision structure?

Let $\mathcal{P} = (\mathbf{M}, \mathcal{C})$ be a Linicrypt program.
A **collision structure** for $\mathcal{P}$ is a tuple $(i^*; c_1, \ldots, c_n)$, where:

1. $c_1, \ldots, c_n$ is an ordering of $\mathcal{C}$

2. c_{i^*} is not determined already by the queries before it

3. all following queries $(c_j \mid j \geq i^*)$ are

not fixed by queries $(c_k \mid k < j)$ before them

What is a collision structure?

1. Identify oracle queries that are the same between runs

$$v_3$$

2. Identify an oracle query that is different

$$w'$$

3. Solve backwards using linear algebra until all queries are defined

$$v_1' \; v_2'$$

What is a collision structure?

1. Identify oracle queries that are the same between runs

$$c_1, \ldots, c_{i^*-1}$$

2. Identify an oracle query that is different

$$c_{i^*}$$

3. Solve backwards using linear algebra until all queries are defined

$$c_{i^*+1}, \ldots, c_n$$

Finding Collision Structures

$$\mathbf{M} = \begin{bmatrix} 1 & 0 & 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 & 1 & 0 \end{bmatrix};$$

$$\mathcal{C} = \left\{ \begin{array}{l} \big(q_1 = [0\ 0\ 1\ 0\ 0\ 0], a_1 = [0\ 0\ 0\ 0\ 1\ 0]\big), \\ \big(q_2 = [0\ 1\ 0\ 1\ 1\ 0], a_2 = [0\ 0\ 0\ 0\ 0\ 1]\big), \\ \big(q_3 = [1\ 0\ 0\ 0\ 0\ 0], a_3 = [0\ 0\ 0\ 1\ 0\ 0]\big), \end{array} \right\}$$

Finding Collision Structures

$$\mathbf{M} = \begin{bmatrix} 1 & 0 & 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 & 1 & 0 \end{bmatrix};$$

$$\mathcal{C} = \left\{ \begin{array}{l} \left(q_1 = [0\ 0\ 1\ 0\ 0\ 0], a_1 = [0\ 0\ 0\ 0\ 1\ 0]\right), \\ \left(\boxed{q_2 = [0\ 1\ 0\ 1\ 1\ 0]}, a_2 = [0\ 0\ 0\ 0\ 0\ 1]\right), \\ \left(q_3 = [1\ 0\ 0\ 0\ 0\ 0], a_3 = [0\ 0\ 0\ 1\ 0\ 0]\right), \end{array} \right\}$$

2. c_{i^*} is not determined already by the queries before it

Finding Collision Structures

$$\mathbf{M} = \begin{bmatrix} 1 & 0 & 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 & 1 & 0 \end{bmatrix};$$

$$\mathcal{C} = \left\{ \begin{array}{l} \big(q_1 = [0\ 0\ 1\ 0\ 0\ 0], a_1 = [0\ 0\ 0\ 0\ 1\ 0]\big), \\ \big(q_2 = [0\ 1\ 0\ 1\ 1\ 0], \boxed{a_2 = [0\ 0\ 0\ 0\ 0\ 1]}\big), \\ \big(q_3 = [1\ 0\ 0\ 0\ 0\ 0], \boxed{a_3 = [0\ 0\ 0\ 1\ 0\ 0]}\big), \end{array} \right\}$$

2. c_{i^*} is not determined already by the queries before it

3. all following queries $(c_j \mid j \geq i^*)$ are not fixed by queries $(c_k \mid k < j)$ before them

Collision Structures Imply Second Preimages

Degeneracy?

$$\mathcal{P}^H(x, y) \; :$$

$$H(x + y)$$

The set of inputs to H do not determine its output

Collision Structures Imply Second Preimages

Degeneracy?

$$\mathcal{P}^H(x, y) :$$

$$H(x + y)$$

The set of inputs to H do not determine it's output

$$\boxed{H(\mathbf{a} + \mathbf{b}) = \mathbf{c} = H(\mathbf{d} + (\mathbf{a} + \mathbf{b} - \mathbf{d}))}$$

Collision Structures Imply Second Preimages

Degeneracy?

$$\mathcal{P}^H(x, y) :$$

$$H(x + y)$$

The set of inputs to H do not determine it's output

$$H(\mathbf{a} + \mathbf{b}) = \mathbf{c} = H(\mathbf{d} + (\mathbf{a} + \mathbf{b} - \mathbf{d}))$$

We have an entire space of collision inputs!

Collision Structures Imply Second Preimages

1. c_{i^*}'s input isn't determined by previous queries.

2. All following queries aren't fixed by those preceding.

$$H(x) = y$$

What if y is already fixed?

Collision Structures Imply Second Preimages

1. C_{i^*}'s input isn't determined by previous queries.

2. All following queries aren't fixed by those preceding.

3. Finally, our outputs are the same, but our inputs differ!

Any Collision Implies Collision Structure

$$(\boldsymbol{x} \neq \boldsymbol{x}') \wedge (\mathcal{P}^H(\boldsymbol{x}) = \mathcal{P}^H(\boldsymbol{x}'))$$

Let $c_1, \ldots, c_n$ be the order of oracle calls computed by $\mathcal{P}^H(\boldsymbol{x})$
and $c'_1, \ldots, c'_n$ be the order of oracle calls computed by $\mathcal{P}^H(\boldsymbol{x}')$

Any Collision Implies Collision Structure

$$(\boldsymbol{x} \neq \boldsymbol{x}') \wedge (\mathcal{P}^H(\boldsymbol{x}) = \mathcal{P}^H(\boldsymbol{x}'))$$

Let $c_1, \ldots, c_n$ be the order of oracle calls computed by $\mathcal{P}^H(\boldsymbol{x})$
and $c_1', \ldots, c_n'$ be the order of oracle calls computed by $\mathcal{P}^H(\boldsymbol{x}')$

Assuming non-degeneracy, there must be a call c_{i^*} that differs.

Any Collision Implies Collision Structure

$$(\boldsymbol{x} \neq \boldsymbol{x}') \wedge (\mathcal{P}^H(\boldsymbol{x}) = \mathcal{P}^H(\boldsymbol{x}'))$$

Let $c_1, \ldots, c_n$ be the order of oracle calls computed by $\mathcal{P}^H(\boldsymbol{x})$
and $c_1', \ldots, c_n'$ be the order of oracle calls computed by $\mathcal{P}^H(\boldsymbol{x}')$

Assuming non-degeneracy, there must be an earliest call c_{i^*} that differs.

Claim: If we order the c_i's so that all queries that differ between the runs (here out, other divergent queries) come after those which are not, then we have a collision structure.

Any Collision Implies Collision Structure

$$\boxed{\begin{aligned}\forall i < i^*, \mathbf{q}_i \cdot \mathbf{v} &= \mathbf{q}_i' \cdot \mathbf{v}' \\ \mathbf{a}_i \cdot \mathbf{v} &= \mathbf{a}_i' \cdot \mathbf{v}'\end{aligned}} \qquad \begin{aligned}\forall j \geq i^*, \mathbf{q}_j \cdot \mathbf{v} &\neq \mathbf{q}_j' \cdot \mathbf{v}' \\ \mathbf{a}_j \cdot \mathbf{v} &\neq \mathbf{a}_j' \cdot \mathbf{v}'\end{aligned}$$

$$\boldsymbol{q}_{i^*} \notin \mathsf{span}\left(\{\boldsymbol{q}_1, \ldots, \boldsymbol{q}_{i^*-1}\} \cup \{\boldsymbol{a}_1, \ldots, \boldsymbol{a}_{i^*-1}\} \cup \mathsf{rows}(\mathbf{M})\right)$$

Otherwise $\boldsymbol{q}_{i^*}$ would be completely determined!

Any Collision Implies Collision Structure

$$\boxed{\begin{aligned}\forall i < i^*, \mathbf{q}_i \cdot \mathbf{v} &= \mathbf{q}'_i \cdot \mathbf{v}' \\ \mathbf{a}_i \cdot \mathbf{v} &= \mathbf{a}'_i \cdot \mathbf{v}'\end{aligned}} \qquad \begin{aligned}\forall j \geq i^*, \mathbf{q}_j \cdot \mathbf{v} &\neq \mathbf{q}'_j \cdot \mathbf{v}' \\ \mathbf{a}_j \cdot \mathbf{v} &\neq \mathbf{a}'_j \cdot \mathbf{v}'\end{aligned}$$

$$\boxed{\boldsymbol{q}_{i^*} \notin \mathsf{span}\left(\{\boldsymbol{q}_1, \ldots, \boldsymbol{q}_{i^*-1}\} \cup \{\boldsymbol{a}_1, \ldots, \boldsymbol{a}_{i^*-1}\} \cup \mathsf{rows}(\mathbf{M})\right)}$$

Otherwise $\boldsymbol{q}_{i^*}$ would be completely determined!

Any Collision Implies Collision Structure

$$\begin{aligned} \forall j \geq i^*, \mathbf{q}_j \cdot \mathbf{v} &\neq \mathbf{q}'_j \cdot \mathbf{v}' \\ \mathbf{a}_j \cdot \mathbf{v} &\neq \mathbf{a}'_j \cdot \mathbf{v}' \end{aligned}$$

For $j \geq i^*$:

$$\boldsymbol{a}_j \notin \mathsf{span}\left(\{\boldsymbol{q}_1, \ldots, \boldsymbol{q}_j\} \cup \{\boldsymbol{a}_1, \ldots, \boldsymbol{a}_{j-1}\} \cup \mathsf{rows}(\mathbf{M})\right)$$

Wrap Up

1. Collisions and preimages can be boiled down to algebra

2. Asymptotically, collision resistance <=> second preimage resistance

3. Properties can be determined in poly time

Limitations and future work

Distinct nonces:

$$H(\mathbf{x}, \mathbf{y}) = H(2; H(1; \mathbf{x})) - H(3; \mathbf{y})$$

$$H(\mathbf{x}, \mathbf{y}) = H(H(\mathbf{x})) - H(\mathbf{y})$$

Limitations and future work

Distinct nonces:

$$H(\mathbf{x},\mathbf{y}) = H(2; H(1;\mathbf{x})) - H(3;\mathbf{y})$$

$$H(\mathbf{x},\mathbf{y}) = H(H(\mathbf{x})) - H(\mathbf{y})$$

$$\mathbf{y} := H(\mathbf{x})?$$

NP complete problem!

Limitations and future work

Distinct nonces:

$$H(\mathbf{x},\mathbf{y}) = H(2; H(1;\mathbf{x})) - H(3;\mathbf{y})$$
$$H(\mathbf{x},\mathbf{y}) = H(H(\mathbf{x})) - H(\mathbf{y})$$

$$\mathbf{y} := H(\mathbf{x})?$$

NP complete problem!

Ideal cipher model?

Thank you

Following this slide are deleted slides

Simple Application

$$\mathcal{P}^H(x_1, x_2, \ldots, x_k):$$
$$y_1 := x_1$$
$$y_2 := H(2; y_1, x_2)$$
$$y_3 := H(3; y_2, x_3)$$
$$\vdots$$
$$y_k := H(k; y_{k-1}, x_k)$$

Collision Structure Algorithm!

Simple Application

$$\mathcal{P}^H(x_1, x_2, \ldots, x_k):$$
$$y_1 := x_1$$
$$y_2 := H(2; y_1, x_2)$$
$$y_3 := H(3; y_2, x_3)$$
$$\vdots$$
$$y_k := H(k; y_{k-1}, x_k)$$

Collision Structure Algorithm!

Steps:

1. Put all queries in LEFT
2. Move queries to RIGHT if $\boldsymbol{a}$ is not fixed by queries in LEFT or M

Simple Application

$$\mathcal{P}^H(x_1, x_2, \ldots, x_k):$$
$$y_1 := x_1$$
$$y_2 := H(2; y_1, x_2)$$
$$y_3 := H(3; y_2, x_3)$$
$$\vdots$$
$$y_k := H(k; y_{k-1}, x_k)$$

Collision Structure Algorithm!

Steps:

1. Put all queries in LEFT
2. Move queries to RIGHT if $\boldsymbol{a}$ is not fixed by queries in LEFT or M
3. Move queries to LEFT if $\boldsymbol{q}$ is fixed by queries in LEFT or M
4. Output : $(i^*; c_1, \ldots, c_n) =$

$$(|\mathtt{LEFT}|; \mathtt{LEFT}||\mathtt{RIGHT}(\text{in reverse order}))$$

Or fail to produce a collision structure

Simple Application

$$\mathcal{P}^H(x_1, x_2, \ldots, x_k):$$

$$y_1 := x_1$$

$$y_2 := H(2; y_1, x_2)$$

$$y_3 := H(3; y_2, x_3)$$

$$\vdots$$

$$y_k := H(k; y_{k-1}, x_k)$$

$\mathtt{LEFT}$

$\mathtt{RIGHT}$

1. Put all queries in LEFT

Simple Application

$\mathcal{P}^H(x_1, x_2, \ldots, x_k)$:	LEFT	RIGHT
$y_1 := x_1$	$y_1 := x_1$	
$y_2 := H(2; y_1, x_2)$	$y_2 := H(2; y_1, x_2)$	
$y_3 := H(3; y_2, x_3)$	$y_3 := H(3; y_2, x_3)$	
$\vdots$	$\vdots$	
$y_k := H(k; y_{k-1}, x_k)$	$y_k := H(k; y_{k-1}, x_k)$	

2. Move queries to RIGHT if $\boldsymbol{a}$ is not fixed by queries in LEFT or M

Simple Application

$$\mathcal{P}^H(x_1, x_2, \ldots, x_k):$$

$$y_1 := x_1$$
$$y_2 := H(2; y_1, x_2)$$
$$y_3 := H(3; y_2, x_3)$$
$$\vdots$$
$$y_k := H(k; y_{k-1}, x_k)$$

LEFT

$$y_1 := x_1$$
$$\boxed{y_2} := H(2; y_1, x_2)$$
$$y_3 := H(3; \boxed{y_2}, x_3)$$
$$\vdots$$
$$y_k := H(k; y_{k-1}, x_k)$$

RIGHT

2. Move queries to RIGHT if $\boldsymbol{a}$ is not fixed by queries in LEFT or M

Simple Application

$$\mathcal{P}^H(x_1, x_2, \ldots, x_k):$$

$$y_1 := x_1$$

$$y_2 := H(2; y_1, x_2)$$

$$y_3 := H(3; y_2, x_3)$$

$$\vdots$$

$$y_k := H(k; y_{k-1}, x_k)$$

LEFT

$$y_1 := x_1$$

$$y_2 := H(2; y_1, x_2)$$

$$y_3 := H(3; y_2, x_3)$$

$$\vdots$$

$$\boxed{y_k} := H(k; y_{k-1}, x_k)$$

RIGHT

2. Move queries to RIGHT if $\boldsymbol{a}$ is not fixed by queries in LEFT or M

Any Collision Implies Collision Structure

For $j \geq i^*$:

$$\boldsymbol{a}_j \notin \mathsf{span}\left(\{\boldsymbol{q}_1, \ldots, \boldsymbol{q}_j\} \cup \{\boldsymbol{a}_1, \ldots, \boldsymbol{a}_{j-1}\} \cup \mathsf{rows}(\mathbf{M})\right)$$

Proof (by contradiction):

$$\exists i : \mathbf{a}_i = \sum_{j \leq i} \alpha_j \mathbf{q}_j + \sum_{j < i} \beta_j \mathbf{a}_j + \gamma \mathbf{M}$$

Any Collision Implies Collision Structure

For $j \geq i^*$:

$$\boldsymbol{a}_j \notin \mathsf{span}\left(\{\boldsymbol{q}_1, \ldots, \boldsymbol{q}_j\} \cup \{\boldsymbol{a}_1, \ldots, \boldsymbol{a}_{j-1}\} \cup \mathsf{rows}(\mathbf{M})\right)$$

Proof (by contradiction):

$$\exists i : \mathbf{a}_i = \sum_{j \leq i} \alpha_j \mathbf{q}_j + \sum_{j<i} \beta_j \mathbf{a}_j + \gamma \mathbf{M}$$

$$\mathbf{a}_i \cdot (\mathbf{v}' - \mathbf{v}) = \sum_{j \leq i} \alpha_j \mathbf{q}_j \cdot (\mathbf{v}' - \mathbf{v}) + \sum_{j<i} \boxed{\beta_j \mathbf{a}_j \cdot (\mathbf{v}' - \mathbf{v})} + \boxed{\gamma \mathbf{M}(\mathbf{v}' - \mathbf{v})}$$

Any Collision Implies Collision Structure

For $j \geq i^*$:

$$\boldsymbol{a}_j \notin \mathsf{span}\left(\{\boldsymbol{q}_1, \ldots, \boldsymbol{q}_j\} \cup \{\boldsymbol{a}_1, \ldots, \boldsymbol{a}_{j-1}\} \cup \mathsf{rows}(\mathbf{M})\right)$$

Proof (by contradiction):

$$\exists i : \mathbf{a}_i = \sum_{j \leq i} \alpha_j \mathbf{q}_j + \sum_{j<i} \beta_j \mathbf{a}_j + \gamma \mathbf{M}$$

$$\mathbf{a}_i \cdot (\mathbf{v}' - \mathbf{v}) = \sum_{j \leq i} \alpha_j \mathbf{q}_j \cdot (\mathbf{v}' - \mathbf{v}) + \sum_{j<i} \boxed{\beta_j \mathbf{a}_j \cdot (\mathbf{v}' - \mathbf{v})} + \boxed{\gamma \mathbf{M}(\mathbf{v}' - \mathbf{v})}$$

$$\mathbf{a}_i \cdot (\mathbf{v}' - \mathbf{v}) = \sum_{j=i^*}^{i} \alpha_j \mathbf{q}_j \cdot (\mathbf{v}' - \mathbf{v}) + \sum_{j=i^*}^{i-1} \beta_j \mathbf{a}_j \cdot (\mathbf{v}' - \mathbf{v})$$

Any Collision Implies Collision Structure

For $j \geq i^*$:

$$\boldsymbol{a}_j \notin \mathsf{span}\left(\{\boldsymbol{q}_1, \ldots, \boldsymbol{q}_j\} \cup \{\boldsymbol{a}_1, \ldots, \boldsymbol{a}_{j-1}\} \cup \mathsf{rows}(\mathbf{M})\right)$$

Proof (by contradiction):

$$\exists i : \mathbf{a}_i = \sum_{j \leq i} \alpha_j \mathbf{q}_j + \sum_{j < i} \beta_j \mathbf{a}_j + \gamma \mathbf{M}$$

$$\mathbf{a}_i \cdot (\mathbf{v}' - \mathbf{v}) = \sum_{j \leq i} \alpha_j \mathbf{q}_j \cdot (\mathbf{v}' - \mathbf{v}) + \sum_{j < i} \beta_j \mathbf{a}_j \cdot (\mathbf{v}' - \mathbf{v}) + \gamma \mathbf{M}(\mathbf{v}' - \mathbf{v})$$

$$\mathbf{a}_i \cdot (\mathbf{v}' - \mathbf{v}) = \sum_{j = i^*}^{i} \alpha_j \mathbf{q}_j \cdot (\mathbf{v}' - \mathbf{v}) + \sum_{j = i^*}^{i-1} \beta_j \mathbf{a}_j \cdot (\mathbf{v}' - \mathbf{v})$$

$$\mathbf{a}_i \cdot \mathbf{v}' = \mathbf{a}_i \cdot \mathbf{v} + \sum_{j = i^*}^{i} \alpha_j \mathbf{q}_j \cdot (\mathbf{v}' - \mathbf{v}) + \sum_{j = i^*}^{i-1} \beta_j \mathbf{a}_j \cdot (\mathbf{v}' - \mathbf{v})$$

Any Collision Implies Collision Structure

For $j \geq i^*$:

$$\boldsymbol{a}_j \notin \mathsf{span}\left(\{\boldsymbol{q}_1, \ldots, \boldsymbol{q}_j\} \cup \{\boldsymbol{a}_1, \ldots, \boldsymbol{a}_{j-1}\} \cup \mathsf{rows}(\mathbf{M})\right)$$

Proof (by contradiction):

$$\exists i : \mathbf{a}_i = \sum_{j \leq i} \alpha_j \mathbf{q}_j + \sum_{j<i} \beta_j \mathbf{a}_j + \gamma \mathbf{M}$$

$$\mathbf{a}_i \cdot (\mathbf{v}' - \mathbf{v}) = \sum_{j \leq i} \alpha_j \mathbf{q}_j \cdot (\mathbf{v}' - \mathbf{v}) + \sum_{j<i} \beta_j \mathbf{a}_j \cdot (\mathbf{v}' - \mathbf{v}) + \gamma \mathbf{M}(\mathbf{v}' - \mathbf{v})$$

$$\mathbf{a}_i \cdot (\mathbf{v}' - \mathbf{v}) = \sum_{j=i^*}^{i} \alpha_j \mathbf{q}_j \cdot (\mathbf{v}' - \mathbf{v}) + \sum_{j=i^*}^{i-1} \beta_j \mathbf{a}_j \cdot (\mathbf{v}' - \mathbf{v})$$

$$\boxed{\mathbf{a}_i \cdot \mathbf{v}'} = \boxed{\mathbf{a}_i \cdot \mathbf{v}} + \sum_{j=i^*}^{i} \boxed{\alpha_j \mathbf{q}_j \cdot (\mathbf{v}' - \mathbf{v})} + \sum_{j=i^*}^{i-1} \boxed{\beta_j \mathbf{a}_j \cdot (\mathbf{v}' - \mathbf{v})}$$