Characterizing Collision and Second-Preimage Resistance in Linicrypt

Ian McQuoid Trevor Swope Mike Rosulek

Oregon State University

An Introduction

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

$$v_4 := H(v_1)$$

$$v_5 := H(v_3)$$

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

$$v_7 := H(v_6)$$

$$v_8 := v_7 + v_1 + v_2$$

$$return(v_8, v_5)$$

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

$$v_4 := H(v_1)$$

$$v_5 := H(v_3)$$

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

$$v_7 := H(v_6)$$

$$v_8 := v_7 + v_1$$

$$return(v_8, v_5)$$

Which program is collision resistant?

What can we do?

Linicrypt programs are a class of algorithms

Introduced by Carmer and Rosulek, Crypto 2016

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

$$v_4 := H(v_1)$$

$$v_5 := H(v_3)$$

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

$$v_7 := H(v_6)$$

$$v_8 := v_7 + v_1$$

$$return(v_8, v_5)$$

What can we do?

Take field elements as input

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

$$v_4 := H(v_1)$$

Query the random oracle

$$v_5 := H(v_3)$$

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

$$v_7 := H(v_6)$$

$$v_8 := v_7 + v_1$$

$$return(v_8, v_5)$$

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

What can we do?

Take field elements as input

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

$$v_4 := H(v_1)$$

Query the random oracle

$$v_5 := H(v_3)$$

Use a fixed linear combination

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

$$v_7 := H(v_6)$$

$$v_8 := v_7 + v_1$$

$$return(v_8, v_5)$$

What can we do?

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

$$v_4 := H(v_1)$$

Query the random oracle $v_5 := H(v_3)$

Use a fixed linear combination $v_6 := v_4 + v_5 + v_2$

$$v_7 := H(v_6)$$

$$v_8 := v_7 + v_1$$

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

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

$$v_4 := H(v_1)$$

$$v_5 := H(v_3)$$

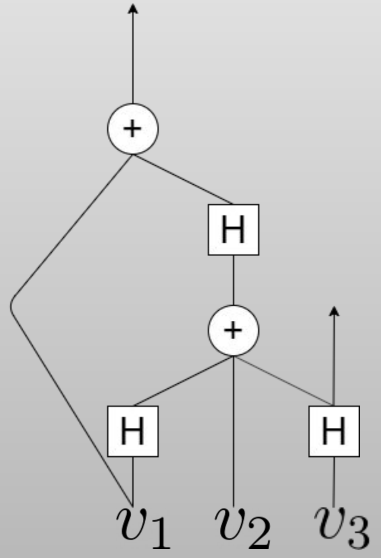
$$v_6 := v_4 + v_5 + v_2$$

$$v_7 := H(v_6)$$

$$v_8 := v_7 + v_1$$

$$return(v_8, v_5)$$

Graphically



Modeling Linicrypt Programs

Algorithmically

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

$$v_4 := H(v_1)$$

$$v_5 := H(v_3)$$

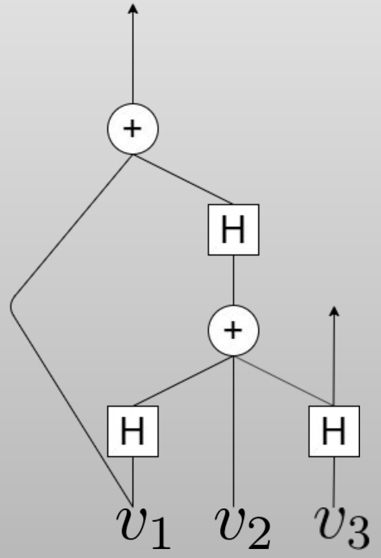
$$v_6 := v_4 + v_5 + v_2$$

$$v_7 := H(v_6)$$

$$v_8 := v_7 + v_1$$

$$return(v_8, v_5)$$

Graphically



Algebraically

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

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

Previous Work

When are two randomized Linicrypt programs indistinguishable?

Carmer and Rosulek 2016

But what about collision resistance?

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

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

1. The set of input variables are different

Second Preimages in Linicrypt

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

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

Collisions in Linicrypt

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

$$v_4 := H(v_1)$$

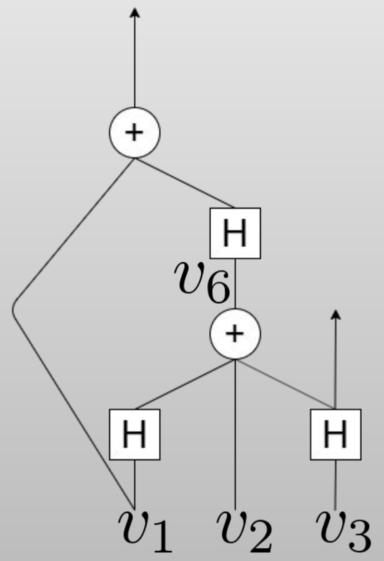
$$v_5 := H(v_3)$$

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

$$v_7 := H(v_6)$$

$$v_8 := v_7 + v_1$$

$$return(v_8, v_5)$$



Collisions in Linicrypt

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

$$v_4 := H(v_1)$$

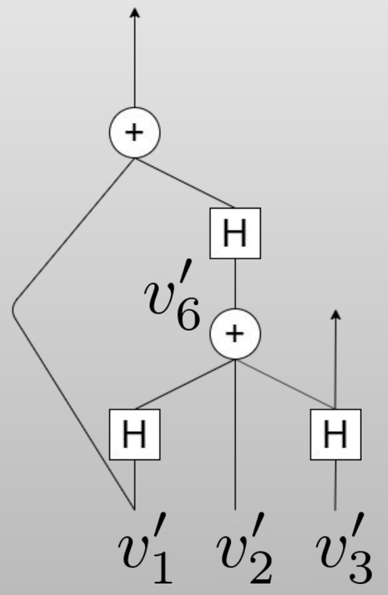
$$v_5 := H(v_3)$$

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

$$v_7 := H(v_6)$$

$$v_8 := v_7 + v_1$$

$$return(v_8, v_5)$$



Collisions in Linicrypt

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

$$v_4 := H(v_1)$$

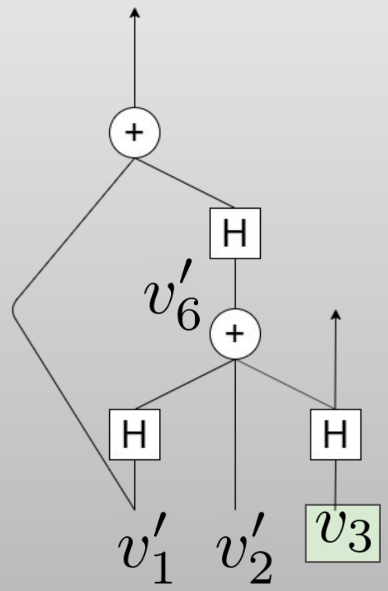
$$\boxed{v_5 := H(v_3)}$$

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

$$v_7 := H(v_6)$$

$$v_8 := v_7 + v_1$$

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



Collisions in Linicrypt

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

$$v_4 := H(v_1)$$

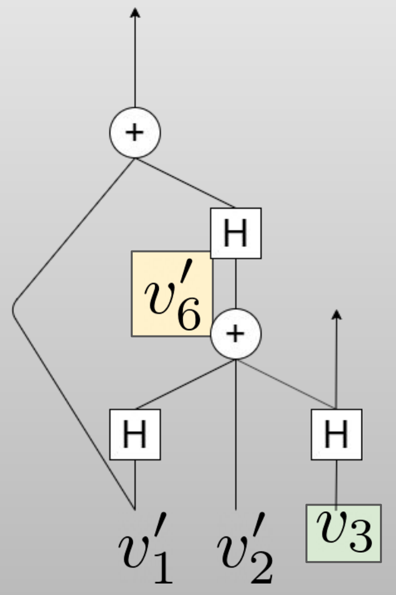
$$\boxed{v_5 := H(v_3)}$$

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

$$v_7 := H(v_6)$$

$$v_8 := v_7 + v_1$$

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



Collisions in Linicrypt

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

$$v_4 := H(v_1)$$

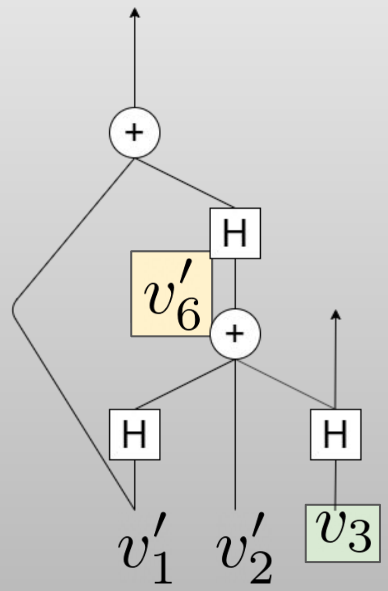
$$\boxed{v_5 := H(v_3)}$$

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

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

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

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



Collisions in Linicrypt

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

$$v_4 := H(v_1)$$

$$v_5 := H(v_3)$$

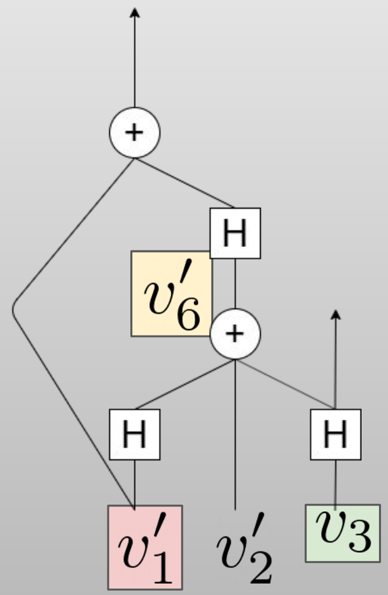
$$v_6 := v_4 + v_5 + v_2$$

$$v_7 := H(v_6)$$

$$v_8 := v_7 + v_1$$

$$return(v_8, v_5)$$

$$v_1' \neq v_1$$



Collisions in Linicrypt

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

$v_4 := H(v_1)$

$v_5 := H(v_3)$

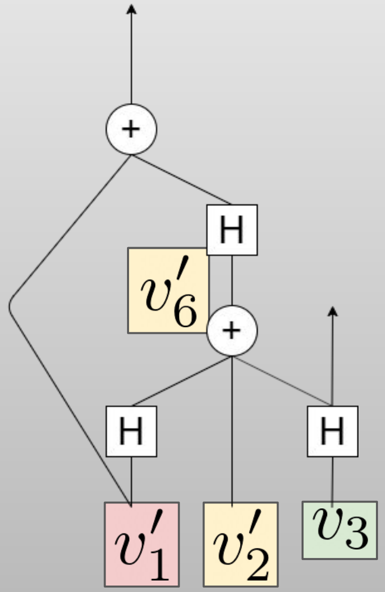
$v_6 := v_4 + v_5 + v_2$

$v_7 := H(v_6)$

$v_8 := v_7 + v_1$

$return(v_8, v_5)$

$v_1' \neq v_1$



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

Collisions in Linicrypt

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

$$v_4 := H(v_1)$$

$$v_5 := H(v_3)$$

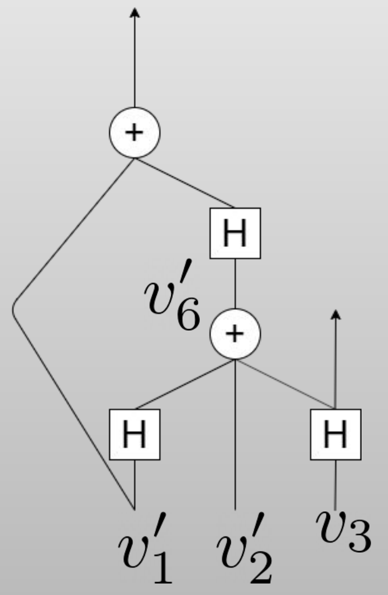
$$v_6 := v_4 + v_5 + v_2$$

$$v_7 := H(v_6)$$

$$v_8 := v_7 + v_1$$

$$return(v_8, v_5)$$

$$v_1' \neq v_1$$



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

Finding Collisions

1. Identify oracle queries that are the same between runs

$$v_3$$

2. Identify an oracle query that is different

$$v_6'$$

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

$$v_1' \; v_2'$$

Algebraic Representation

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

$$v_4 := H(v_1)$$

$$v_5 := H(v_3)$$

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

$$v_7 := H(v_6)$$

$$v_8 := v_7 + v_1$$

$$return(v_8, v_5)$$

$$\begin{pmatrix} v_1 \\ v_2 \\ v_3 \\ v_4 \\ v_5 \\ v_6 \\ v_7 \\ v_8 \end{pmatrix} = \begin{pmatrix} 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 1 & 0 & 1 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 \\ 1 & 0 & 0 & 0 & 0 & 1 \end{pmatrix} \cdot \begin{pmatrix} v_1 \\ v_2 \\ v_3 \\ v_5 \\ v_7 \end{pmatrix}$$

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

Algebraic Representation

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

$$v_4 := H(v_1)$$

$$v_5 := H(v_3)$$

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

$$v_7 := H(v_6)$$

$$v_8 := v_7 + v_1$$

$$return(v_8, v_5)$$

$$\begin{pmatrix} v_1 \\ v_2 \\ v_3 \\ v_4 \\ v_5 \\ v_6 \\ v_7 \\ v_8 \end{pmatrix} = \begin{pmatrix} 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 1 & 0 & 1 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 \\ 1 & 0 & 0 & 0 & 0 & 1 \end{pmatrix} \cdot \begin{pmatrix} v_1 \\ v_2 \\ v_3 \\ v_5 \\ v_7 \end{pmatrix}$$

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

Every intermediate value is a linear combo of base vars

Algebraic Representation

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

$$\begin{pmatrix} v_1 \\ v_2 \\ v_3 \\ v_4 \\ v_5 \\ v_6 \\ v_7 \\ v_8 \end{pmatrix} = \begin{pmatrix} 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 1 & 0 & 1 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 \\ 1 & 0 & 0 & 0 & 0 & 1 \end{pmatrix} \cdot \begin{pmatrix} v_1 \\ v_2 \\ v_3 \\ v_5 \\ v_7 \end{pmatrix}$$

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

The outputs of a linicrypt program are held in $\boldsymbol{M}$

Algebraic Representation

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

$$v_4 := H(v_1)$$

$$v_5 := H(v_3)$$

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

$$v_7 := H(v_6)$$

$$v_8 := v_7 + v_1$$

$$return(v_8, v_5)$$

$$\begin{pmatrix} v_1 \\ v_2 \\ v_3 \\ v_4 \\ v_5 \\ v_6 \\ v_7 \\ v_8 \end{pmatrix} = \begin{pmatrix} 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 1 & 0 & 1 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 \\ 1 & 0 & 0 & 0 & 0 & 1 \end{pmatrix} \cdot \begin{pmatrix} v_1 \\ v_2 \\ v_3 \\ v_5 \\ v_7 \end{pmatrix}$$

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

But how do we represent queries to the oracle?

Algebraic Representation

$$\begin{pmatrix} v_1 \\ v_2 \\ v_3 \\ v_4 \\ v_5 \\ v_6 \\ v_7 \\ v_8 \end{pmatrix} = \begin{pmatrix} 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 1 & 0 & 1 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 \\ 1 & 0 & 0 & 0 & 0 & 1 \end{pmatrix} \cdot \begin{pmatrix} v_1 \\ v_2 \\ v_3 \\ v_5 \\ v_7 \end{pmatrix}$$

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

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

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

Algebraic Representation

$$\begin{pmatrix} v_1 \\ v_2 \\ v_3 \\ v_4 \\ v_5 \\ v_6 \\ v_7 \\ v_8 \end{pmatrix} = \begin{pmatrix} 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 1 & 0 & 1 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 \\ 1 & 0 & 0 & 0 & 0 & 1 \end{pmatrix} \cdot \begin{pmatrix} v_1 \\ v_2 \\ v_3 \\ v_5 \\ v_7 \end{pmatrix}$$

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

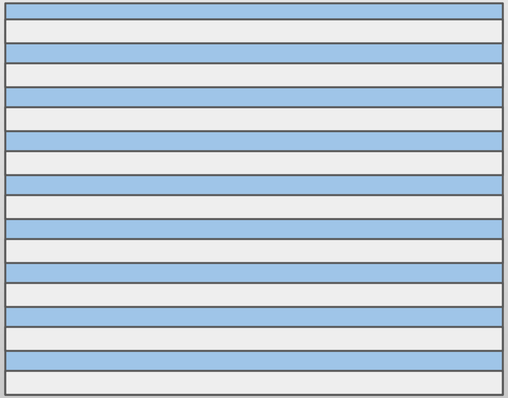
$$\mathcal{C} = \left\{ \begin{array}{l} \left([1\ 0\ 0\ 0\ 0\ 0], [0\ 0\ 0\ 1\ 0\ 0] \right), \\ \left([0\ 0\ 1\ 0\ 0\ 0], [0\ 0\ 0\ 0\ 1\ 0] \right), \\ \left([0\ 1\ 0\ 1\ 1\ 0], [0\ 0\ 0\ 0\ 0\ 1] \right) \end{array} \right\}$$

This query corresponds to $v_4 := H(v_1)$

What is a collision structure?

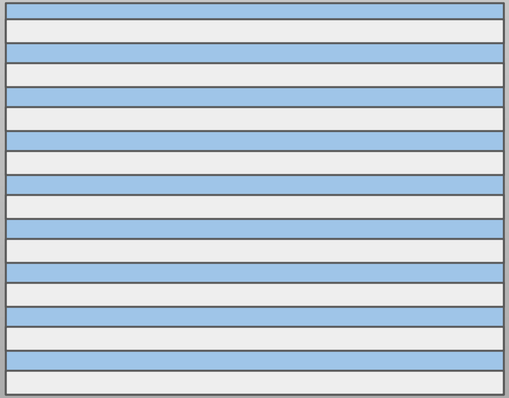
Here are the internals of two runs of $\mathcal{P}^H$

$v =$



$\cdot [v_{\text{base}}]$

$v' =$



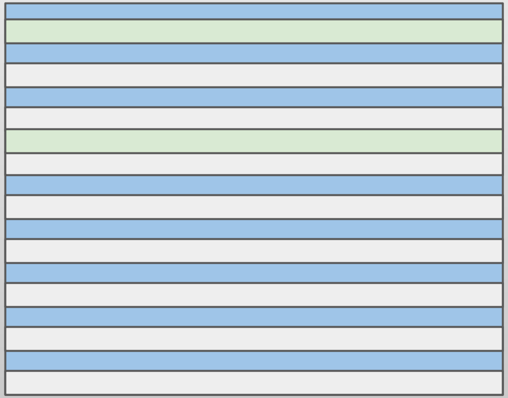
$\cdot [v'_{\text{base}}]$

What is a collision structure?

Here are the internals of two runs of $\mathcal{P}^H$

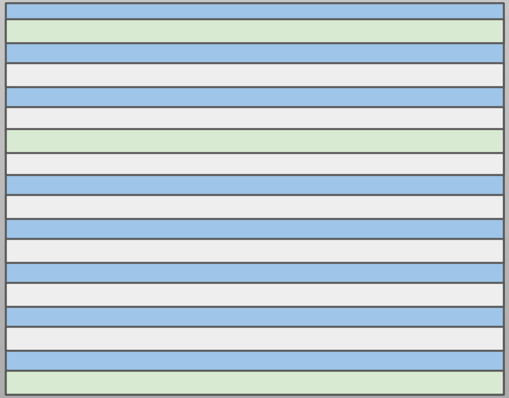
1. Identify base variables shared

$v =$



$\cdot [v_{\text{base}}]$

$v' =$



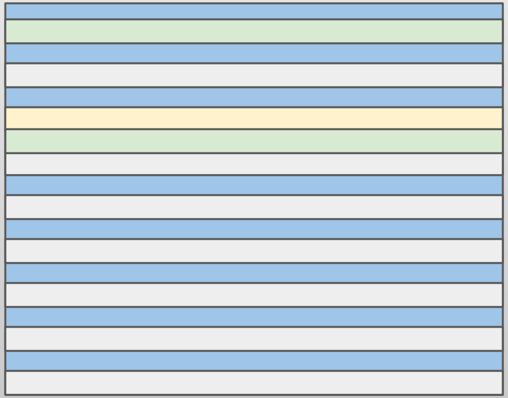
$\cdot [v'_{\text{base}}]$

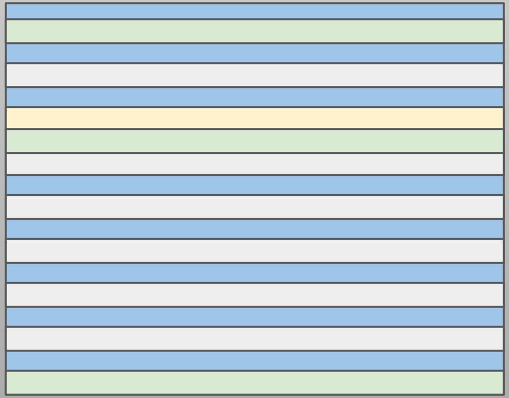
What is a collision structure?

Here are the internals of two runs of $\mathcal{P}^H$

1. Identify base variables shared

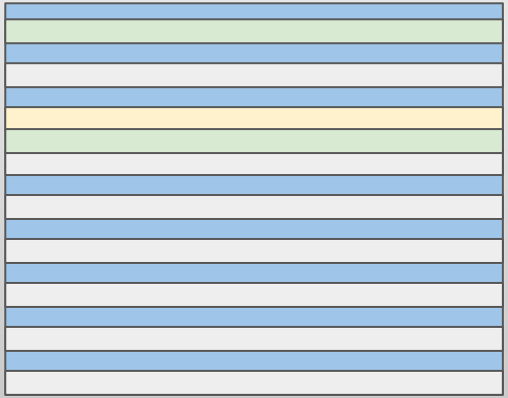
2. Identify a base variable that is different

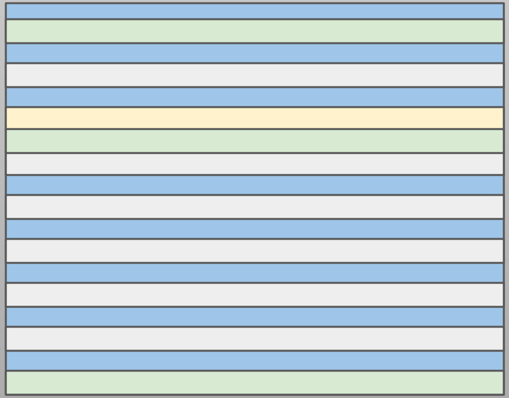
$v =$  $\cdot[v_{\text{base}}]$

$v' =$  $\cdot[v'_{\text{base}}]$

What is a collision structure?

Here are the internals of two runs of $\mathcal{P}^H$

$v =$  $\cdot[v_{\text{base}}]$

$v' =$  $\cdot[v'_{\text{base}}]$

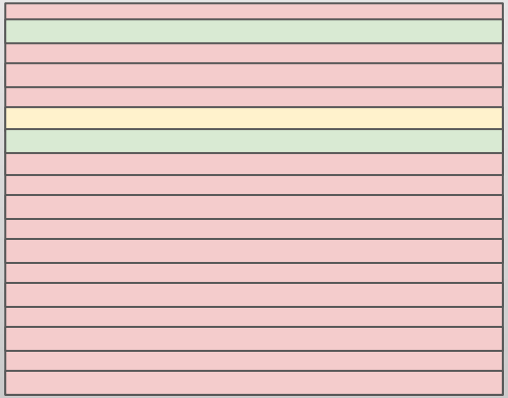
1. Identify base variables shared
2. Identify a base variable that is different

Corresponding query must be lin. indep!

What is a collision structure?

Here are the internals of
two runs of $\mathcal{P}^H$

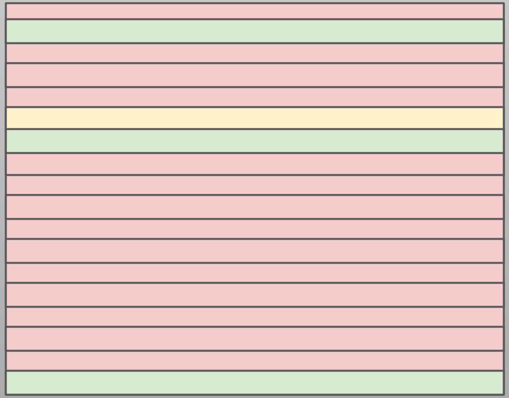
$$v = $$



$$\cdot [v_{\text{base}}]$$

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

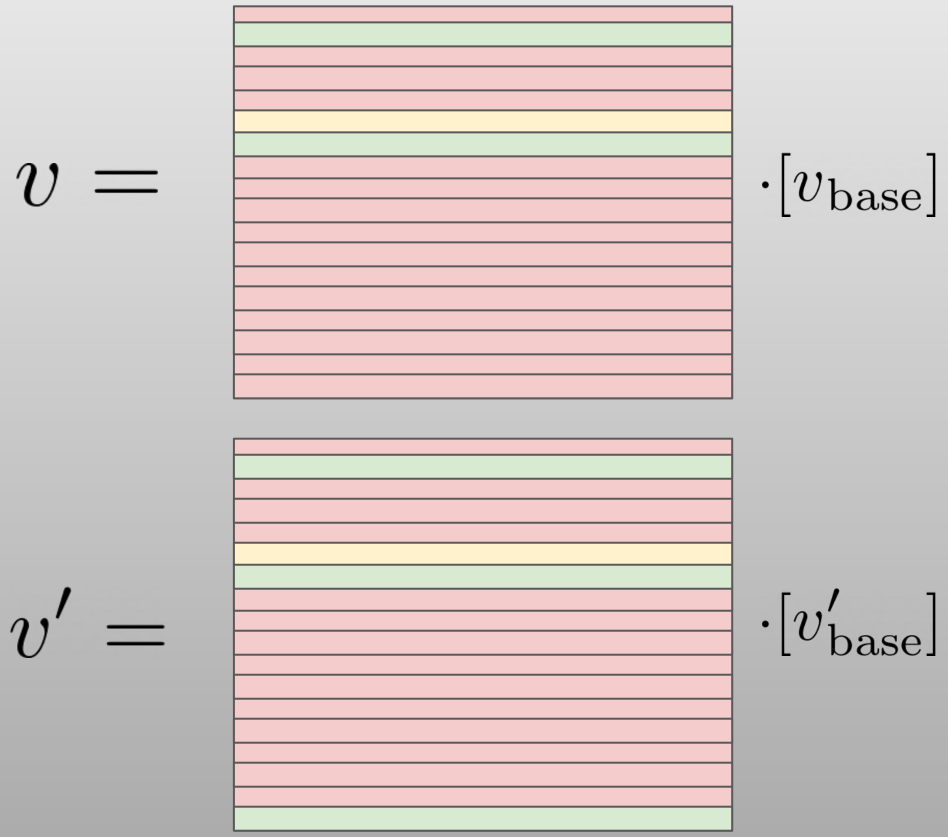
$$v' = $$



$$\cdot [v'_{\text{base}}]$$

What is a collision structure?

Here are the internals of two runs of $\mathcal{P}^H$



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

$$v'_7 = H(v'_6)$$

What if either is fixed?

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. The inputs to c_{i^*} are not in the span of the queries before it union the total output.

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. The inputs to c_{i^*} are not in the span of the queries before it union the total output.

3. All following queries' $(c_j \mid j \geq i^*)$ outputs are not in the span of the queries before them union the total output.

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

$$v_6'$$

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

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

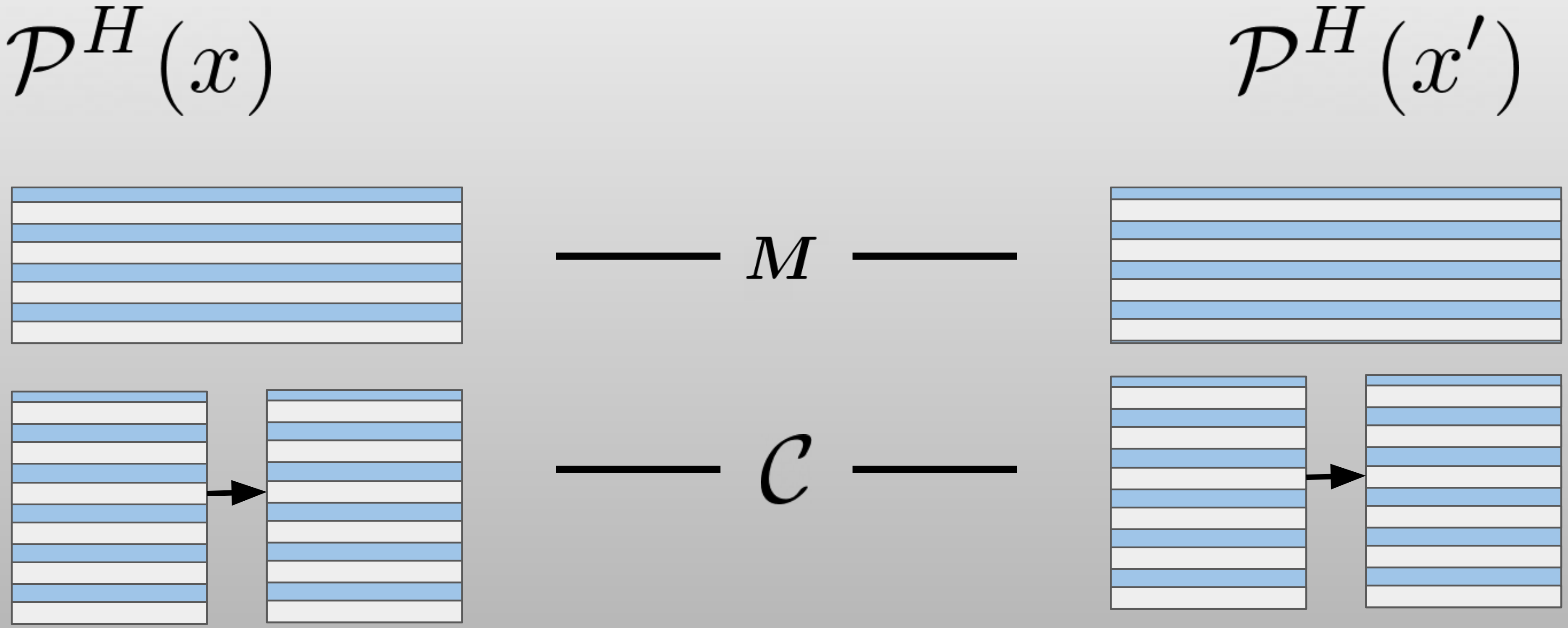
2. Identify an oracle query that is different

$$c_{i^*}$$

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

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

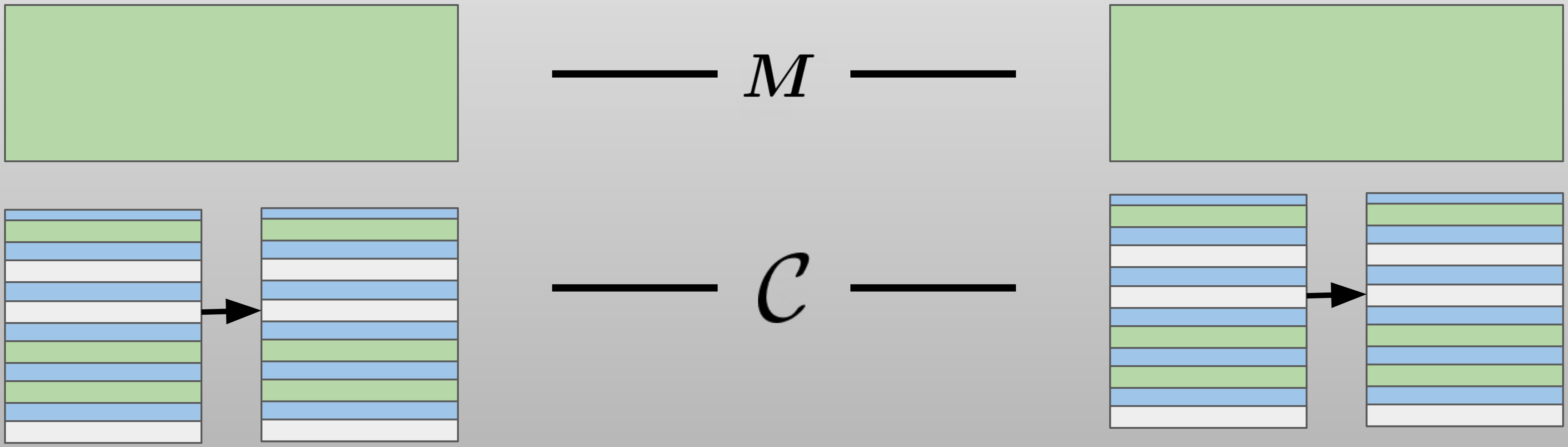
Finding Collision Structures From Second Preimages



Which queries are green?

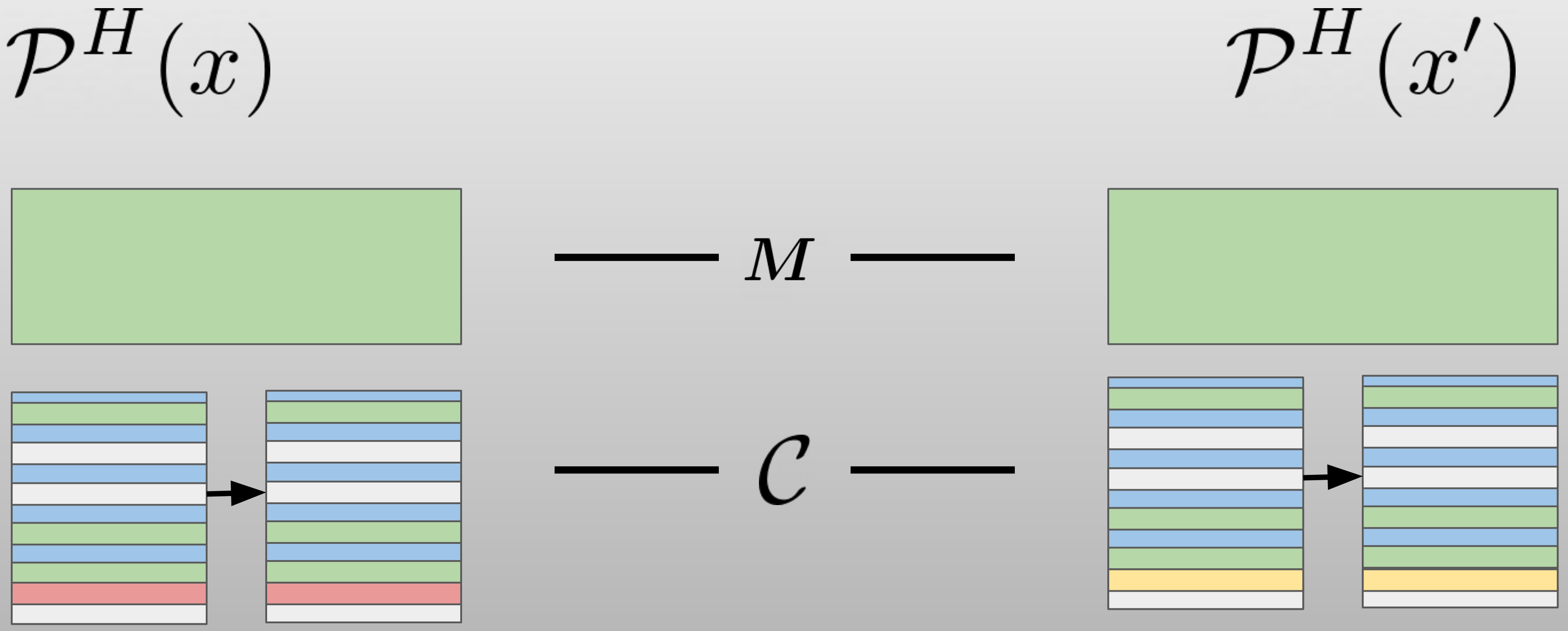
Finding Collision Structures From Second Preimages

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

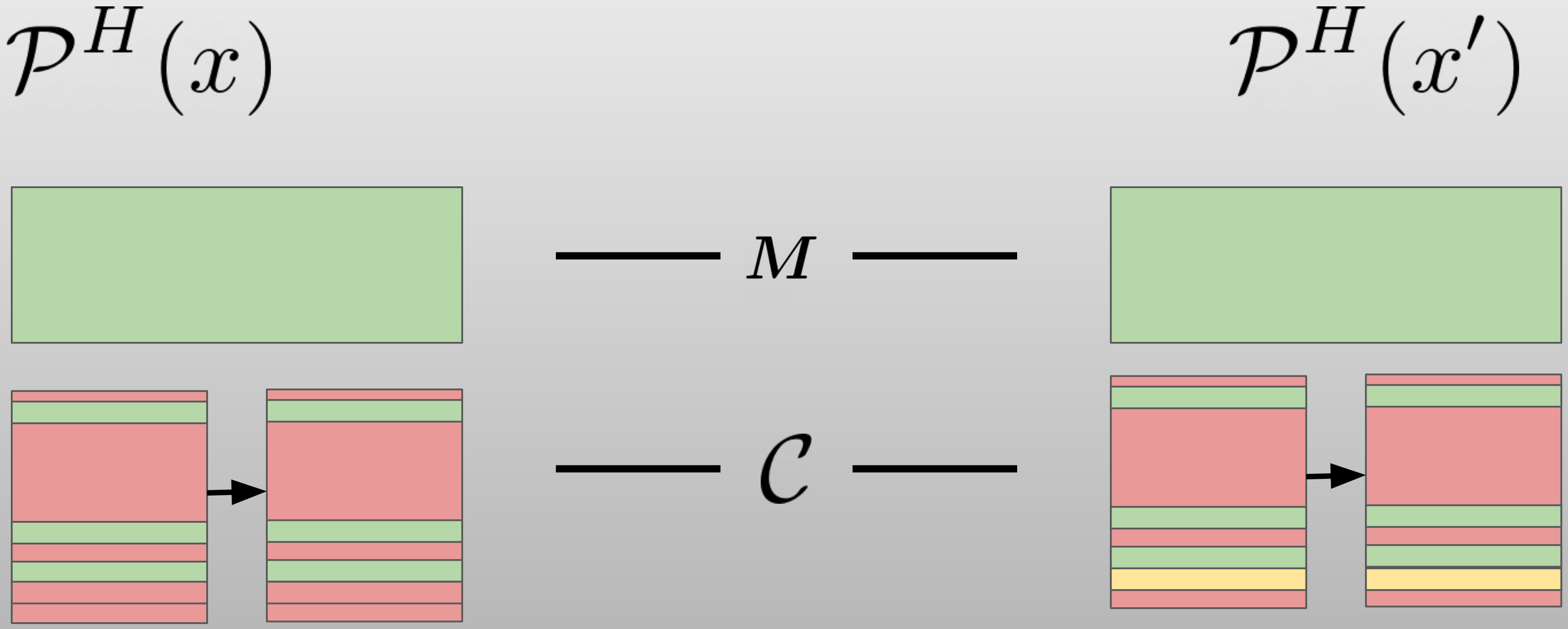


What's our special query?

Finding Collision Structures From Second Preimages



Finding Collision Structures From Second Preimages



Now we can create our Collision Structure

Theorem Statement

For a linicrypt program with distinct nonces:

$$\text{Collision Structures in a Program} \iff \text{No Collision Resistance}^* \iff \text{No 2nd Preimage Resistance}^*$$

* modulo degeneracy

Wrap Up

1. Collisions and second preimages can be boiled down to algebra

2. Properties can be determined in poly time

3. We often only have to worry about second preimages

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

What is degeneracy?

Degeneracy?

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

$$H(x + y)$$

All queries are identical, but inputs are different!

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

We have an entire space of collision inputs!