Chapter 3

Linicrypt with Ideal Ciphers

3.1 Revisiting Algebraic Representations

We have constructed the algebraic representation of a Linicrypt program. A question that arises is: Which combination of matrices of the structure $(\boldsymbol{M}, \mathcal{C})$ correspond to some valid Linicrypt program?

To answer this question, we need to define the terminology more carefully.

Definition 3.1. *A random oracle constraint of dimension* base *with* k *inputs is a tuple* $(t, \boldsymbol{Q}, \boldsymbol{a})$ *for* $t \in \{0,1\}^*$, $\boldsymbol{Q} \in \mathbb{F}^{k \times \mathsf{base}}$ *and* $\boldsymbol{a} \in \mathbb{F}^{1 \times \mathsf{base}}$.

We call t the nonce and refer to $\boldsymbol{Q}$ and $\boldsymbol{a}$ as the query and answer to the random oracle. Usually we just say constraint when the other variables are clear from the context. The constraints $(t, \boldsymbol{Q}, \boldsymbol{a})$ encodes a relationship between the base variables $\boldsymbol{v}_{\mathsf{base}} \in \mathbb{F}^{\mathsf{base}}$ of a program. Namely $H(t, \boldsymbol{Q}\boldsymbol{v}_{\mathsf{base}}) = \boldsymbol{a}\boldsymbol{v}_{\mathsf{base}}$. Because H is a well-defined function, and not just any relation, these requirements extend to the constraints.

Definition 3.2. *A set of (random oracle) constraints* $\mathcal{C}$ *is well-defined if for any pair of constraints* $c_i, c_j \in \mathcal{C}$ *we have* $(t_i, \boldsymbol{Q}_i) = (t_j, \boldsymbol{Q}_j) \implies \boldsymbol{a}_i = \boldsymbol{a}_j$.

When we use a set of constraints, we will implicitly also require that it is well-defined. Now we can characterize which sets of constraints correspond to a Linicrypt program.

Definition 3.3 (Solvable)**.** *Let* $\mathcal{C}$ *be a finite set of valid constraints.* $\mathcal{C}$ *is (deterministically) solvable if there exists an ordering* $(c_1, \ldots, c_n)$ *of* $\mathcal{C}$ *and a subspace* $\mathcal{F}$ *of* $\mathbb{F}^{\mathsf{base}}$ *such that for all* $i = 1, \ldots n$*:*

1. $\mathsf{span}(\boldsymbol{Q}_i) \subset \mathcal{F} + \mathsf{span}\big(c_1, \ldots, c_{i-1}\big)$
2. $\boldsymbol{a}_i \notin \mathcal{F} + \mathsf{span}\big(c_1, \ldots, c_{i-1}\big) + \mathsf{span}(\boldsymbol{Q}_i)$
3. *TODO or this notation*

4. $\mathsf{rows}(\boldsymbol{Q}_i) \subset \mathsf{span}\big(\mathcal{F} \cup \mathsf{rows}(c_1) \cup \cdots \cup \mathsf{rows}(c_{i-1})\big)$
5. $\boldsymbol{a}_i \notin \mathsf{span}\big(\mathcal{F} \cup \mathsf{rows}(c_1) \cup \cdots \cup \mathsf{rows}(c_{i-1}) \cup \mathsf{rows}(\boldsymbol{Q}_i)\big)$

We call $\mathcal{F}$ the solvable space (TODO or free space or fixable space or fixed space) of $\mathcal{C}$ and write $\mathsf{sol}(\mathcal{C}) = \mathcal{F}$. We call $(c_1, \ldots, c_n)$ the (solution) ordering of $\mathcal{C}$.

If we construct the algebraic representation of a Linicrypt program $\mathcal{P}$, we get a solvable set of constraints. Indeed, the ordering of the constraints in the definition can be exactly the order of the corresponding queries in the execution of $\mathcal{P}$. In this case, the solvable space would be the space spanned by the corresponding vectors to the input variables and the randomly sampled variables.

The other direction is also true.

Lemma 3.4 (Solvable constraints)**.** *Let $\mathcal{C}$ be a set of solvable constraints and $k = \dim(\mathsf{sol}(\mathcal{C}))$. Let $\mathsf{out} \in \mathbb{N}$ be arbitrary and $\mathsf{in} \leq k$. Let $\boldsymbol{M} \in \mathbb{F}^{\mathsf{out} \times \mathsf{base}}$ be an arbitrary output matrix. Then there is a basis change $\boldsymbol{B} \in \mathbb{F}^{\mathsf{base} \times \mathsf{base}}$ and a Linicrypt program $\mathcal{P}$ taking in inputs such that $(\boldsymbol{M}, \mathcal{C}\boldsymbol{B}^{-1})$ is it's algebraic representation.*

Proof. Let $(\boldsymbol{x}_1, \ldots, \boldsymbol{x}_k)$ be a basis for $\mathsf{sol}(\mathcal{C})$ and let $(c_1, \ldots, c_n)$ be the ordering. The new basis for $\mathbb{F}^{\mathsf{base}}$ is $(\boldsymbol{x}_1, \ldots, \boldsymbol{x}_k, \boldsymbol{a}_1, \ldots, \boldsymbol{a}_n)$.

$$\boldsymbol{B} = \begin{bmatrix} \boldsymbol{x}_1 & \ldots & \boldsymbol{x}_k & \boldsymbol{a}_1 & \ldots & \boldsymbol{a}_n \end{bmatrix}^\top$$

TODO or

$$\boldsymbol{B} = \begin{bmatrix} \boldsymbol{x}_1 \\ \vdots \\ \boldsymbol{x}_k \\ \boldsymbol{a}_1 \\ \vdots \\ \boldsymbol{a}_n \end{bmatrix}$$

Then we define the constraints $\mathcal{C}'$ via $t'_i = t_i$, $\boldsymbol{Q}'_i B = \boldsymbol{Q}_i$ and $\boldsymbol{a}'_i B = \boldsymbol{a}_i$. Note, that, as $\mathcal{C}$ is solvable via the ordering $(c_1, \ldots, c_n)$, these constraints have the form

$$\begin{aligned} \boldsymbol{Q}'_i &= \begin{bmatrix} \lambda_1^i & \cdots & \lambda_{i-1}^i & 0 & 0 & \cdots & 0 \end{bmatrix} \qquad \text{for} \quad \lambda_j^i \in \mathbb{F} \\ \boldsymbol{a}'_i &= \begin{bmatrix} 0 & \ldots & 0 & 1 & 0 & \ldots & 0 \end{bmatrix} \end{aligned}$$

This is the correct form for an algebraic representation of a program $\mathcal{P}$ taking $\mathsf{in} \leq k$ inputs, randomly sampling the next $k - \mathsf{in} \geq 0$ base variables, and outputting according to $\boldsymbol{M}$. □

TODO maybe switch to using input matrices, this would maybe clean things up later

Lemma 3.5 (Solvable constraints)**.** *Let $\mathcal{C}$ be a set of solvable constraints and $k = \dim(\mathsf{sol}(\mathcal{C}))$. Let $\mathsf{out} \in \mathbb{N}$ be arbitrary and $\mathsf{in} \leq k$. Let $\boldsymbol{M} \in \mathbb{F}^{\mathsf{out}\times\mathsf{base}}$ be arbitrary and $\boldsymbol{I} \in \mathbb{F}^{\mathsf{in}\times\mathsf{base}}$ such that $\mathsf{span}(\boldsymbol{I}) \subseteq \mathsf{sol}(\mathcal{C})$. Then there is a basis change $\boldsymbol{B} \in \mathbb{F}^{\mathsf{base}\times\mathsf{base}}$ and a Linicrypt program $\mathcal{P}$ such that $(\boldsymbol{I}\boldsymbol{B}^{-1}, \boldsymbol{M}\boldsymbol{B}^{-1}, \mathcal{C}\boldsymbol{B}^{-1})$ is it's algebraic representation.*

3.2 Revisiting Collision Structures

Using this language we can argue about the invertibility of a Linicrypt program and about the possibility to directly calculate second preimages. Let $\mathcal{P} = (\boldsymbol{M}, \mathcal{C})$ be a Linicrypt program.

Lemma 3.6. *$\mathcal{P}$ is invertible if $\mathsf{rows}(\boldsymbol{M}) \subseteq \mathsf{sol}(\mathcal{C})$.*

Proof. TODO □

Lemma 3.7. *$\mathcal{P}$ has a collision structure if $\mathcal{C} = \mathcal{C}_{fixed} \sqcup \mathcal{C}_{cs}$ such that*

$$\mathsf{sol}(\mathcal{C}_{cs}) \supset \mathsf{span}\big(\mathsf{rows}(\mathcal{C}_{fixed}) \cup \mathsf{rows}(\boldsymbol{M})\big). \tag{3.1}$$

Proof. TODO □

Note, that it is crucial that the space on the left is $\supset$ and not only $\supseteq$, as this gives the extra degree of freedom to find a different preimage. This is the same role that w' plays in the example from [**?**].

This characterization directly includes the case of degeneracy, because then $\mathcal{C}_{\text{cs}} = \{\}$ and $\mathsf{sol}(\mathcal{C}_{\text{cs}}) = \mathbb{F}^{\mathsf{base}}$, while degeneracy means precisely $\mathbb{F}^{\mathsf{base}} \supset \mathsf{span}(\mathsf{rows}(\mathcal{C}) \cup \mathsf{rows}(\boldsymbol{M}))$.

The following example was slightly adapted so that it is invertible.

$\mathcal{P}^{\mathcal{E}}_{\mathsf{inv},1}(x, y, z)$

$w = H(x) + H(z) + y$

return $(H(w) + x, z)$

$$\boldsymbol{M} = \begin{bmatrix} 1 & 0 & 0 & 0 & 0 & 1 \\ 0 & 0 & 1 & 0 & 0 & 0 \end{bmatrix}$$
$$\boldsymbol{q}_1 = \begin{bmatrix} 1 & 0 & 0 & 0 & 0 & 0 \end{bmatrix}$$
$$\boldsymbol{a}_1 = \begin{bmatrix} 0 & 0 & 0 & 1 & 0 & 0 \end{bmatrix}$$
$$\boldsymbol{q}_2 = \begin{bmatrix} 0 & 0 & 1 & 0 & 0 & 0 \end{bmatrix}$$
$$\boldsymbol{q}_2 = \begin{bmatrix} 0 & 0 & 0 & 0 & 1 & 0 \end{bmatrix}$$
$$\boldsymbol{q}_3 = \begin{bmatrix} 0 & 1 & 0 & 1 & 1 & 0 \end{bmatrix}$$
$$\boldsymbol{q}_3 = \begin{bmatrix} 0 & 0 & 0 & 0 & 0 & 1 \end{bmatrix}$$

Note, that (c_3, c_2, c_1) is an ordering solving $\mathcal{C}$ and $\mathsf{rows}(M) \subset \mathsf{sol}(\mathcal{C})$.

TODO finish typing this example

3.3 Adapting the Linicrypt model to use block ciphers

In this chapter we modify the Linicrypt model to make use of the ideal cipher model instead of the random oracle model. This means that a Linicrypt program gets access to a block cipher $\mathcal{E} = (E, D)$ where E and D are functions $\mathbb{F} \times \mathbb{F} \to \mathbb{F}$ instead of the hash function $H : \{0,1\}^* \times \mathbb{F}^* \to \mathbb{F}$. By the definition of a block cipher, $E_k := E(k, \cdot)$ is a permutation of $\mathbb{F}$ for all $k \in \mathbb{F}$ and $D_k := D(k, \cdot)$ is its inverse. In the ideal cipher model, we assume that the block cipher has no weakness. This is modeled by choosing each permutation E_k uniformly at random at the beginning of every security game.

The command $y = E(k, x)$ in a Linicrypt program has to be treated differently from the command $y = H(k, x)$ when considering collision resistance, because an attacker has access to the deterministic Linicrypt program and both directions of the block cipher $\mathcal{E} = (E, D)$. Consider these two programs, $\mathcal{P}^H$ in standard Linicrypt and $\mathcal{P}^{\mathcal{E}}$ in ideal cipher Linicrypt.

$\mathcal{P}^H(k, x)$
return $H(k, x)$

$\mathcal{P}^{\mathcal{E}}(k, x)$
return $E(k, x)$

While $\mathcal{P}^H$ is collision resistant, it is trivial to find second preimages for $\mathcal{P}^{\mathcal{E}}$ For any $k' \in \mathbb{F}$ the pair $(k', D(k', E(k, x)))$ is a second preimage to (k, x).

This invertibility property of block ciphers has to be taken into account in both the algebraic representation and the characterization of collision resistance.

3.3.1 Algebraic representation for ideal cipher Linicrypt

Let $\mathcal{P}$ be a ideal cipher Linicrypt program. For each query to E of the form $y = E(k, x)$ we define the **associated ideal cipher constraint** $(E, \boldsymbol{k}, \boldsymbol{x}, \boldsymbol{y})$. Each query to D of the form $x = D(k, y)$, is associated with the constraint $(D, \boldsymbol{k}, \boldsymbol{y}, \boldsymbol{x})$.

As with standard Linicrypt, we want to exclude programs that make unnecessary queries to the block cipher. This way the base variables are linearly independent, except for the dependencies the adversary might introduce by carefully choosing the input. Hence we assume wlog that no two constraints have the same $(E, \boldsymbol{k}, \boldsymbol{x})$ or $(D, \boldsymbol{k}, \boldsymbol{y})$.

With ideal ciphers there is a second way to make an unnecessary query. That is by first computing $y = E(k, x)$ and then $x' = D(k, y)$. As D_k is the inverse of E_k we have $x = x'$ although $\boldsymbol{x}$ and $\boldsymbol{x}'$ are linearly independent.

Therefore for all $\boldsymbol{k}, \boldsymbol{x}, \boldsymbol{x}', \boldsymbol{y}, \boldsymbol{y}' \in \mathbb{F}^{\mathsf{base}}$ we can assume there are no two constraints $(E, \boldsymbol{k}, \boldsymbol{x}, \boldsymbol{y})$ and $(D, \boldsymbol{k}, \boldsymbol{y}, \boldsymbol{x}')$ in $\mathcal{C}$ for $\boldsymbol{x} \neq \boldsymbol{x}'$. Neither can there be $(D, \boldsymbol{k}, \boldsymbol{y}, \boldsymbol{x})$ and $(E, \boldsymbol{k}, \boldsymbol{x}, \boldsymbol{y}')$ in $\mathcal{C}$ for $\boldsymbol{y} \neq \boldsymbol{y}'$.