IDEAS ON A CHARACTERIZATION FOR COLLISION RESISTANCE IN LINICRYPT WITHOUT NONCES

FREDERIK SEMMEL

26 August 2024

Abstract. TODO

1. Definitions and building blocks

Remark 1.1: Notation

This is a summary of the notation and definitions that I am using.

- d is the number of base variables of $\mathcal{P}$, k the number of input variables, l the number of output variables and n the number of constraints.
- The state space of the program $\mathcal{P}$ is $\mathbb{F}^d$
- The function going from the inputs of a deterministic $\mathcal{P}$ to the values of the base variables (the state space) is called $b_{\mathcal{P}} : \mathbb{F}^k \to \mathbb{F}^d$
- Instead of $\boldsymbol{v}_{\mathsf{base}}$ I will just write $\boldsymbol{v}$
- The set of base variables one can get by executing a Linicrypt program $\mathcal{P}$ on all of its inputs is called $\mathsf{exec}(\mathcal{P})$
- The set of solutions to a set of constraints $\mathcal{C}$ is called $\mathsf{sol}(\mathcal{P})$
- The input matrix to $\mathcal{P}$ is called $\boldsymbol{I}$ and the output matrix is called $\boldsymbol{O}$ so I will write $\mathcal{P} = (\boldsymbol{I}, \boldsymbol{O}, \mathcal{C})$ for a Linicrypt program and its algebraic representation
- The function that a determinsitic $\mathcal{P}$ computes is called $f_{\mathcal{P}} = \boldsymbol{O} \circ b_{\mathcal{P}}$

Additionally, I want to note these facts about basis change.

- We can apply a basis change $\boldsymbol{B}$ to a program $\mathcal{P} = (\boldsymbol{I}, \boldsymbol{O}, \mathcal{C})$. If $\boldsymbol{B} \in F^{d \times d}$ is invertible then $\mathcal{P}\boldsymbol{B}$ is defined as the program corresponding to $(\boldsymbol{I}\boldsymbol{B}, \boldsymbol{O}\boldsymbol{B}, \mathcal{C}\boldsymbol{B})$. There is a formal framework to show that this represents the same Linicrypt program.

The main idea for tackling Linicrypt programs in the no-nonces setting is to use basis change and generalize it to any non-square, non-invertible matrices. This was the driving reason for the generalizations we defined in my master's thesis.

We have to allow for the "algebraic representations" $\mathcal{P} = (\boldsymbol{I}, \boldsymbol{O}, \mathcal{C})$ to be much freer. Essentially anything with consistent dimensions should be allowed, and all definitions and theorems should still be applicable. A triple $(\boldsymbol{I}, \boldsymbol{O}, \mathcal{C})$ is called a "pseudo-algebraic representation" if all the rows of the matrices are in $F^{1 \times d}$. Then we can use a core lemma that relates the solution spaces $\mathsf{sol}(\mathcal{C})$ between transformed constraints, which follows essentially from the associativity of the matrix product.

I want to state the definition of collision structure for these pseudo-algebraic representations.

Definition 1.1 (Collision structure): Let $\mathcal{P} = (\boldsymbol{I}, \boldsymbol{O}, \mathcal{C})$ be a "pseudo-algebraic representation". We say $\mathcal{C}$ has a collision structure if we can split $\mathcal{C}$ as $\mathcal{C} = \mathcal{C}_{cs} \sqcup \mathcal{C}_{fix}$ and find a non-trivial subspace $\mathcal{F} \subseteq \mathbb{F}^{1\times d}$ such that

$$\mathcal{C}_{cs} \text{ is deterministically solvable fixing } \left(\mathsf{rowsp}\left(\mathcal{C}_{fix}\right) + \mathsf{rowsp}(O)\right) \oplus \mathcal{F}. \tag{1}$$

Remark 1.2:

- The fixed constraints $\mathcal{C}_{fix}$ and the backwards solvable constraints $\mathcal{C}_{cs}$ are equivalent to the constraints before and after i^* in the original collision structure definition.
- Degeneracy is a natural edge case of this defintion, i.e. $\mathcal{C}_{cs} = \{\}$.
- The subspace $\mathcal{F}$ corresponds to the variables in $\mathcal{P}$ which we can set freely and still solve the constraints. It plays the same role as the $\boldsymbol{q}_{i^*}$, which is not in the span of previously fixed variables.

The attack follows from the definition of the collision structure: We are given a $\boldsymbol{v} \in \mathsf{sol}(\mathcal{C})$. In particular, it also solves $\mathcal{C}_{fix}$. We fix the values for all the dual vectors contained in $\mathsf{rowsp}\left(\mathcal{C}_{fix}\right) + \mathsf{rowsp}(O)$. Then we choose any other value than $\boldsymbol{fv}$ for $\boldsymbol{f} \neq 0 \in \mathcal{F}$. By the lemma on the structure of $\mathsf{sol}(\mathcal{C})$, we have a deterministic Linicrypt program computing a solution $\boldsymbol{v}'$ for $\mathcal{C}_{cs}$ taking these fixed values as input. This solution is our collision, as we have $\boldsymbol{Iv} \neq \boldsymbol{Iv}'$ (injectivity of $\boldsymbol{I}$ on the solutions), $\boldsymbol{v}' \in \mathsf{sol}(\mathcal{C}) = \mathsf{sol}(\mathcal{C}_{cs}) \cap \mathsf{sol}\left(\mathcal{C}_{fix}\right)$ (by the definitions) and $\boldsymbol{Ov} = \boldsymbol{Ov}'$ (by the choice of $\boldsymbol{v}'$).

We can therefore state the following theorem, which combines all the previously described attacks for both the random oracle model, and the ideal cipher model.

Theorem 1.1: Let $(\boldsymbol{I}, \boldsymbol{O}, \mathcal{C})$ be a pseudo-algebraic representation with a collision structure $\mathcal{C} = \mathcal{C}_{cs} \sqcup \mathcal{C}_{fix}$. Assume we already know a $\boldsymbol{v}$ in $\mathsf{sol}(\mathcal{C})$.

Then there is a Linicrypt program that can output a $\boldsymbol{v}'$ in $\mathsf{sol}(\mathcal{C})$ with $\boldsymbol{v} \neq \boldsymbol{v}'$ and $\boldsymbol{Ov} = \boldsymbol{Ov}'$ by making $|\mathcal{C}_{cs}|$ queries to H.

The definition of the solution set (in the ROM) is the following.

Definition 1.2 (Solution of constraints): Let $\mathcal{C}$ be a set of constraints of dimension d. We say a vector $\boldsymbol{v} \in \mathbb{F}^d$ **solves** $\mathcal{C}$ if $\boldsymbol{av} = H(\boldsymbol{Qv})$ for all $\boldsymbol{Q} \mapsto \boldsymbol{a}$ in $\mathcal{C}$. Such a vector $\boldsymbol{v}$ is called a **solution** of $\mathcal{C}$. The set of all solutions to $\mathcal{C}$ is denoted by $\mathsf{sol}_H(\mathcal{C})$.

The definition works for any function H, and we will just write $\mathsf{sol}(\mathcal{C})$ and omit the H from the notation.

TODO: maybe I should put here the definition for deterministically solvable, and the lemma about the structure of $\mathsf{sol}(\mathcal{C})$.

The central lemma that is used in the conjecture below is:

Lemma 1.1 (Transformation of constraints): Let $\mathcal{C}$ be a set of constraints of dimension d and let $\boldsymbol{T}$ be a matrix in $F^{d\times d'}$ where d' can be arbitrary. Then we have

$$\boldsymbol{v} \in \mathsf{sol}(\mathcal{C}\boldsymbol{T}) \iff \boldsymbol{T}\boldsymbol{v} \in \mathsf{sol}(\mathcal{C}) \tag{2}$$

Proof: Let $\mathcal{C}$ and $\boldsymbol{T}$ be as in the lemma. The following equivalences prove the lemma:

$$\begin{aligned}
&\boldsymbol{v} \in \mathsf{sol}(\mathcal{C}\boldsymbol{T}) \\
\Leftrightarrow\ &(\boldsymbol{a}')\boldsymbol{v} = H((\boldsymbol{Q}')\boldsymbol{v}) \quad \forall(\boldsymbol{Q}', \boldsymbol{a}') \in \mathcal{C}\boldsymbol{T} \\
\Leftrightarrow\ &(\boldsymbol{a}\boldsymbol{T})\boldsymbol{v} = H((\boldsymbol{Q}\boldsymbol{T})\boldsymbol{v}) \quad \forall(\boldsymbol{Q}, \boldsymbol{a}) \in \mathcal{C} \\
\Leftrightarrow\ &\boldsymbol{a}(\boldsymbol{T}\boldsymbol{v}) = H(\boldsymbol{Q}(\boldsymbol{T}\boldsymbol{v})) \quad \forall(\boldsymbol{Q}, \boldsymbol{a}) \in \mathcal{C} \\
\Leftrightarrow\ &\boldsymbol{T}\boldsymbol{v} \in \mathsf{sol}(\mathcal{C})
\end{aligned} \tag{3}$$

□

The equivalence between lines 2 and 3 looks strange, because $\mathcal{C}'$ could have fewer constraints than $\mathcal{C}$, but it is correct. This lemma holds for *any* function H.

2. Conjecture

I am sure that the conjecture still needs tweaking. But in this form, the attack side already kind of works, and it explains all the examples I have found. The next step would be to try to break the conjecture by finding more complicated counter examples.

Conjeture 2.1 (Formulation version 1): Let $\mathcal{P} = (\boldsymbol{I}, \boldsymbol{O}, \mathcal{C})$ be a Linicrypt program. and assume there is an injective $\boldsymbol{T}$ in $M^{d\times d'}$ such that

(a) $\mathcal{P}\boldsymbol{T}$ has a collision structure, or

(b) There exists a basis change $\boldsymbol{B} \neq \boldsymbol{b}(1)$ in $M^{d'\times d'}$ with $\mathcal{C}\boldsymbol{T}\boldsymbol{B} = \mathcal{C}\boldsymbol{T}$ and $\boldsymbol{O}\boldsymbol{T}\boldsymbol{B} = \boldsymbol{O}\boldsymbol{T}$ and $\boldsymbol{I}\boldsymbol{T} \neq \boldsymbol{b}(0)$

Then there exists an attack on collision resistance against $\mathcal{P}$. If both (a) and (b) are not the case for any injective $\boldsymbol{T}$, then $\mathcal{P}$ is collision-resistant.

With this formulation there are these issues:

1. For (a), we need that $\mathcal{C}\boldsymbol{T}$ is solvable, so that we can actually find any $\boldsymbol{v}$ in $\mathsf{sol}(\mathcal{C}\boldsymbol{T})$
2. For (b), we need to be able to find a $\boldsymbol{v}$ in $\mathsf{sol}(\mathcal{C}\boldsymbol{T})$ with $\boldsymbol{B}\boldsymbol{v} \neq \boldsymbol{v}$
3. Also, $\boldsymbol{I}\boldsymbol{T}$ needs to be injective on $\mathsf{sol}(\mathcal{C}\boldsymbol{T})$

The last two issues are not issues if $\boldsymbol{T}$ is the identity. Then 2. is satisfied because H is assumed to look random, and cannot fulfill a linear relationship. And 3. is satisfied anyways by the Lemma on the structure of a solvable $\mathcal{C}$.

Conjeture 2.2 (Formulation version 2): Let $\mathcal{P} = (\boldsymbol{I}, \boldsymbol{O}, \mathcal{C})$ be a pseudo algebraic representation of a Linicrypt program. $\mathcal{P}$ is not collision resistant if and only if at least one of these conditions hold:
(a) $\mathcal{P}$ has a collision structure.
(b) There exists a basis change $\boldsymbol{B} \neq \boldsymbol{b}(1)$ with $\mathcal{C}\boldsymbol{B} = \mathcal{C}$ and $\boldsymbol{O}\boldsymbol{B} = \boldsymbol{O}$.
(c) There is an injective map $\boldsymbol{T}$ in $\mathbb{F}^{d' \times d}$ and a submatrix $\boldsymbol{I}'$ of $\boldsymbol{I}$ such that $\mathcal{P}' = (\boldsymbol{I}'\boldsymbol{T}, \boldsymbol{O}\boldsymbol{T}, \mathcal{C}\boldsymbol{T})$ is a pseudo algebraic representation for which (a) or (b) hold.

3. Attack Side

The attack side works for this conjecture. The conjecture is enough to explain all CR-insecure examples that I tried. But the security proof still has a few "holes" in the argumentation, even though a lot of things seem to go right. Here I want to prove the direction: If (a), (b) or (c) is true then there is a collision resistance attack on $\mathcal{P}$.

Proof:

3.1. **Case (a).** This is given by Theorem 1.1

3.2. **Case (b), the permutation attack.** Assume that we have $\mathcal{C}\boldsymbol{B} = \mathcal{C}$ and $\boldsymbol{O}\boldsymbol{B} = \boldsymbol{O}$ for a non-trivial basis change $\boldsymbol{B}$. Note, that this is only possible due to there being no nonces in the constraints. Because $\mathcal{C}$ is an unordered set of constraints, $\mathcal{C}\boldsymbol{B}$ can be equal to $\mathcal{C}$ if $\boldsymbol{B}$ manages to permute the constraints in $\mathcal{C}$.

Then let $\boldsymbol{v}$ be in $\mathsf{sol}(\mathcal{C})$ such that $\boldsymbol{B}\boldsymbol{v} \neq \boldsymbol{v}$. It is possible to find such a $\boldsymbol{v}$ with very high probability simply by choosing any $\boldsymbol{v}$ randomly out of $\mathsf{sol}(\mathcal{C})$. If $\mathsf{sol}(\mathcal{C})$ was contained to any subspace of $\mathbb{F}^d$ like the eigenspace of $\boldsymbol{B}$, then H is definitively not independently random.

Remark 3.2.1: This can be made precise by switching to the canonical algebraic representation, where the input and the answer vectors are all orthogonal. Then $\mathsf{sol}(\mathcal{C})$ is a graph on top of $X \coloneqq \mathsf{ker}(\boldsymbol{I})^{\perp}$. Because H is a random oracle, any point above X is equally likely, so the probability that the graph lies in a subspace is negligible.

By the Lemma 1.1 we have $\boldsymbol{v} \in \mathsf{sol}(\mathcal{C}) = \mathsf{sol}(\mathcal{C}\boldsymbol{B}) \Leftrightarrow \boldsymbol{B}\boldsymbol{v} \in \mathsf{sol}(\mathcal{C})$. Finally, as we have assumed that $\boldsymbol{O}\boldsymbol{B} = \boldsymbol{O}$ we get that:

$$\boldsymbol{O}\boldsymbol{B}\boldsymbol{v} = (\boldsymbol{O}\boldsymbol{B})\boldsymbol{v} = \boldsymbol{O}\boldsymbol{v} \tag{4}$$

By choice of $\boldsymbol{v}$ we have $\boldsymbol{B}\boldsymbol{v} \neq \boldsymbol{v}$, therefore $\boldsymbol{I}\boldsymbol{B}\boldsymbol{v} \neq \boldsymbol{I}\boldsymbol{v}$ are a collision for $\mathcal{P}$.

3.3. **Case (c), the collapse of queries.** Assume that $\mathcal{P}' = (\boldsymbol{I}'\boldsymbol{T}, \boldsymbol{O}\boldsymbol{T}, \mathcal{C}\boldsymbol{T})$ is a pseudo algebraic representation for which (a) or (b) hold. Now by the attack proofs for case (a) and case (b) we get a $\boldsymbol{v} \neq \boldsymbol{v}'$ in $\mathsf{sol}(\mathcal{C}\boldsymbol{T})$ such that $(\boldsymbol{O}\boldsymbol{T})\boldsymbol{v} = (\boldsymbol{O}\boldsymbol{T})\boldsymbol{v}'$. We have $\boldsymbol{I}'\boldsymbol{T}\boldsymbol{v} \neq \boldsymbol{I}'\boldsymbol{T}\boldsymbol{v}'$ by the bijection between $\mathsf{sol}(\mathcal{C}\boldsymbol{T})$ and $\boldsymbol{I}'\boldsymbol{T}$ because $\mathcal{P}'$ is a pseudo algebraic representation.

Now we can apply Lemma 1.1 which gives us in this context the following equivalence:

$$\boldsymbol{v} \in \mathsf{sol}(\mathcal{C}\boldsymbol{T}) \Longleftrightarrow \boldsymbol{T}\boldsymbol{v} \in \mathsf{sol}(\mathcal{C}) \tag{5}$$

Putting it all together we have $\boldsymbol{T}\boldsymbol{v} \neq \boldsymbol{T}\boldsymbol{v}'$, which are both in $\mathsf{sol}(\mathcal{C})$ and fulfill the equation $\boldsymbol{O}(\boldsymbol{T}\boldsymbol{v}) = \boldsymbol{O}(\boldsymbol{T}\boldsymbol{v}')$. So $\boldsymbol{I}\boldsymbol{T}\boldsymbol{v}$ and $\boldsymbol{I}\boldsymbol{T}\boldsymbol{v}'$ are collisions for $\mathcal{P}$.

Remark 3.3.1 (Unsolvable constraints): Something that could be interesting to look at is what algebraic conditions does $\boldsymbol{T}$ need to fulfill so that $\mathcal{C}\boldsymbol{T}$ is still solvable. Intuitively $\boldsymbol{T}$ should not create loops in the constraints, where we need the output of some queries to determine the inputs to exactly these queries. This plays into the same problem of trying to better understand unsolvable set of constraints. Sometimes properties of H can also make unsolvable $\mathcal{C}$ actually easy to solve, for example if H has fixed points, then the "point loop" $\mathcal{C} = \{\boldsymbol{q} \mapsto \boldsymbol{q}\}$ has a solution.

□

4. Examples for the attack side

4.1. **Collapsing program from McQuoid, Swope, Rosulek.** This is the example that the authors of [1] used to highlight why the plain collision structure characterization fails in the setting without nonces.

$\underline{\mathcal{P}(x,y)}$
$a_1 \coloneqq H(x)$
$a_2 \coloneqq H(y)$
$a_3 \coloneqq H(a_1)$
return $a_3 - a_2$

$\underline{\text{Algebraic Representation of } \mathcal{P}(x,y)}$

$$\boldsymbol{O} \coloneqq \begin{bmatrix} 0 & 0 & 0 & -1 & 1 \end{bmatrix}$$

$$\boldsymbol{I} \coloneqq \begin{bmatrix} 1 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 \end{bmatrix}$$

$$\mathcal{C} = \{[1\ 0\ 0\ 0\ 0] \mapsto [0\ 0\ 1\ 0\ 0],\ [0\ 1\ 0\ 0\ 0] \mapsto [0\ 0\ 0\ 1\ 0],\ [0\ 0\ 1\ 0\ 0] \mapsto [0\ 0\ 0\ 0\ 1]\}$$

The attack for this program is to input $(x, H(x))$ for any $x \in \mathbb{F}$ because $\mathcal{P}(x, H(x)) = 0$.

What happens is that $\boldsymbol{a}_2$ and $\boldsymbol{a}_3$ collapse during the execution of $\mathcal{P}$ on these inputs. This can be represented by the matrix $\boldsymbol{T}$ defined as:

$$\boldsymbol{T} = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \\ 0 & 0 & 1 \end{bmatrix} \tag{6}$$

Then the program $\mathcal{P}\boldsymbol{T}$ is specified by $(\boldsymbol{I}'\boldsymbol{T}, \boldsymbol{O}\boldsymbol{T}, \mathcal{C}\boldsymbol{T})$ with $\boldsymbol{I}'$ being only the first row of $\boldsymbol{I}$ and

$$\mathcal{C}\boldsymbol{T} = \{[1\ 0\ 0] \mapsto [0\ 1\ 0], [0\ 1\ 0] \mapsto [0\ 0\ 1]\} \qquad \boldsymbol{I}'\boldsymbol{T} = [1\ 0\ 0] \qquad \boldsymbol{O}\boldsymbol{T} = [0\ 0\ 0].$$

Note, that $\mathcal{C}\boldsymbol{T}$ is not deterministically solvable fixing $\mathsf{rowsp}(\boldsymbol{I}\boldsymbol{T})$. But it is solvable fixing only $\mathsf{rowsp}(\boldsymbol{I}'\boldsymbol{T})$. The key thing is that $\mathcal{P}\boldsymbol{T}$ now has a collision structure!

We can split $\mathcal{C}\boldsymbol{T} = \mathcal{C}_{cs}\boldsymbol{T} \sqcup \mathcal{C}_{fix}\boldsymbol{T}$ with $\mathcal{C}_{cs}\boldsymbol{T} = \mathcal{C}\boldsymbol{T}$ and $\mathcal{C}_{fix}\boldsymbol{T} = \{\}$. If we set $\mathcal{F} = \mathsf{rowsp}([1\ 0\ 0])$ then we get that $\mathcal{C}_{cs}\boldsymbol{T}$ is detereminstically solvable fixing

$$(\mathsf{rowsp}(\mathcal{C}_{fix}\boldsymbol{T}) + \mathsf{rowsp}(\boldsymbol{OT}) \oplus \mathcal{F} = \mathsf{rowsp}([1\ \ 0\ \ 0]). \tag{7}$$

Therefore case (c) of Conjeture 2.2 is fulfilled.

Here I write out exactly what happens in the proof for this concrete example. By solving $\mathcal{C}_{cs}\boldsymbol{T}$ (i.e. running the attacking Linicrypt program) in two different ways by fixing a different value for $[1\ 0\ 0]\boldsymbol{v}$, we get two distinct $\boldsymbol{v}$ and $\boldsymbol{v}'$ in $\mathsf{sol}(\mathcal{C}\boldsymbol{T})$. They have the form:

$$\boldsymbol{v} = \begin{bmatrix} x \\ H(x) \\ H(H(x)) \end{bmatrix} \neq \begin{bmatrix} x' \\ H(x') \\ H(H(x')) \end{bmatrix} = \boldsymbol{v}' \tag{8}$$

Now, by the Lemma 1.1 we get the solutions for $\mathcal{C}$ by applying $\boldsymbol{T}$ from the other side.

$$\boldsymbol{Tv} = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \\ 0 & 0 & 1 \end{bmatrix} \boldsymbol{v} = \begin{bmatrix} x \\ H(x) \\ H(x) \\ H(H(x)) \\ H(H(x)) \end{bmatrix} \tag{9}$$

These lead to the expected colliding inputs to $\mathcal{P}$: $\boldsymbol{ITv} = \begin{bmatrix} x \\ H(x) \end{bmatrix}$ and $\boldsymbol{ITv}' = \begin{bmatrix} x' \\ H(x') \end{bmatrix}$.

4.2. **Collapse Attack: One of the 5 broken PGV compression functions in the rate-2 setting.** This example should work for any of the 5 compression functions listed in [2] in the section "Fatal Attacks on Five of PGV's B-Labeled Schemes".

I will choose the compression function they call $\hat{f}_{39}$ and call it f.

$$f(h, m) = E(h + m, h + m) + m \tag{10}$$

The Merkle-Damgard like construction that I will use starts with the IV = 0. This constant could also be passed in as another input and output, or it could be set to IV = H(0). I believe all three approaches are equivalent in the end.

$\underline{H_f^2(m_1, m_2)}$
$h_0 := 0$
$y_1 := E(h_0 + m_1, h_0 + m_1)$
$h_1 := y_1 + m_1$
$y_2 := E(h_1 + m_2, h_1 + m_2)$
$h_2 := y_2 + m_2$
return h_2

$\underline{\text{Algebraic Representation of } \mathcal{P}(x, y)}$

$\boldsymbol{I} := \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \end{bmatrix}$

$\boldsymbol{O} := [0\ 1\ 0\ 1]$

$\mathcal{C} = \Big\{ [1\ 0\ 0\ 0] \overset{[1\ 0\ 0\ 0]}{\longleftrightarrow} [0\ 0\ 1\ 0],$

$[1\ 1\ 1\ 0] \overset{[1\ 1\ 1\ 0]}{\longleftrightarrow} [0\ 0\ 0\ 1] \Big\}$

The goal is now to find a $\boldsymbol{T}'$ such that the two queries collapse. I first try to find a $\boldsymbol{T}'$ in $\mathbb{F}^{4\times 4}$ and then I remove the zero columns to make it an injective matrix. I write down the equations to make the second query collapse onto the first query:

$$
\begin{aligned}
[1\ 0\ 0\ 0]\boldsymbol{T}' &= [1\ 0\ 0\ 0] \\
[0\ 0\ 1\ 0]\boldsymbol{T}' &= [0\ 0\ 1\ 0] \\
[1\ 1\ 1\ 0]\boldsymbol{T}' &= [1\ 0\ 0\ 0] \\
[0\ 0\ 0\ 1]\boldsymbol{T}' &= [0\ 0\ 1\ 0]
\end{aligned}
\tag{11}
$$

This system of equations has a solution:

$$
\boldsymbol{T}' = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 0 & -1 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 1 & 0 \end{bmatrix} \tag{12}
$$

And if we remove the second and fourth column, we get a $\boldsymbol{T}$ as in Conjeture 2.1:

$$
\boldsymbol{T} = \begin{bmatrix} 1 & 0 \\ 0 & -1 \\ 0 & 1 \\ 0 & 1 \end{bmatrix} \tag{13}
$$

Remark 4.2.1: I don't have an algorithm for finding these $\boldsymbol{T}$ yet. Also, I believe that such an algorithm has to be in NP-hard in the number of constraints, because in [3] they showed that finding the collapsing potential is NP-hard. This doesn't seem to be a problem to me, as exponential computation in the number of base variables is still doable. And if we had lots of base variables, we probably have a very specific structure (like in the MD-construction) which we can exploit for faster algorithms.

The collapsed program is then

$$
\mathcal{C}\boldsymbol{T} = \{([1\ 0], [1\ 0], [0\ 1])\} \qquad \boldsymbol{I}'\boldsymbol{T} = [1\ 0] \qquad \boldsymbol{O}\boldsymbol{T} = [0\ 0].
$$

This is the same case as the previous example, the $\mathcal{C}\boldsymbol{T}$ is solvable fixing the output and fixing $[1\ 0]$. Because this program has a collision structure (specifically it is degenerate) we get solutions in $\mathsf{sol}(\mathcal{C}\boldsymbol{T})$, i.e. any $\boldsymbol{v} = (x, E(x, x))$ for x in $\mathbb{F}$ is in $\mathsf{sol}(\mathcal{C}\boldsymbol{T})$. We map it to collisions of $\mathcal{P}$ via $\boldsymbol{I}\boldsymbol{T}$, so every $\boldsymbol{I}\boldsymbol{T}\boldsymbol{v} = (x, -E(x, x))$ for x in $\mathbb{F}$ collides with each other.

This is the same result as in [2], except that they work in a field of characteristic 2 where $-1 = 1$ and hence they omit the minus sign.

This example highlights the power of the Linicrypt approach. It seems to me that the original PGV paper did not find these attacks, but here these attacks can be discovered by solving algebraic conditions on the algebraic representation of the program.

4.3. **Non example for the collapse attack: The 8 group-2 schemes.** I tried the scheme $f(h, m) = E(m, h) + m$. For it, the equations to make the constraints collapse are inconsistent, so no $\boldsymbol{T}$ can exist that collapses the queries.

Remark 4.3.1: It also doesn't work if one tries to collapse y_2 onto x_1. This "reverse and collapse" seems like an interesting attack. I want to check if this attack is possible for any of the PGV functions in the Merkle Damgard construction. But considering the amount of work that has been done on this topic, I don't think there will be a PGV compression function for which this attack is actually a new attack.

4.4. **Permuation attack: The compression functions marked as (P) in PGV.** One of the compression functions that is attackable by the Permutation Attack in PGV is $f(h,m) = E(m,m) + h$.

$\underline{H_f^2(m_1, m_2)}$
$h_0 := 0$
$y_1 := E(m_1, m_1)$
$h_1 := y_1 + h_0$
$y_2 := E(m_2, m_2)$
$h_2 := y_2 + h_1$
return h_2

$\underline{\text{Algebraic Representation of } \mathcal{P}(x,y)}$

$$\boldsymbol{I} := \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \end{bmatrix}$$

$$\boldsymbol{O} := \begin{bmatrix} 0 & 0 & 1 & 1 \end{bmatrix}$$

$$\mathcal{C} = \left\{ \begin{bmatrix} 1 & 0 & 0 & 0 \end{bmatrix} \overset{\begin{bmatrix} 1 & 0 & 0 & 0 \end{bmatrix}}{\longleftrightarrow} \begin{bmatrix} 0 & 0 & 1 & 0 \end{bmatrix}, \right.$$
$$\left. \begin{bmatrix} 0 & 1 & 0 & 0 \end{bmatrix} \overset{\begin{bmatrix} 0 & 1 & 0 & 0 \end{bmatrix}}{\longleftrightarrow} \begin{bmatrix} 0 & 0 & 0 & 1 \end{bmatrix} \right\}$$

With these constraints $\mathcal{C}$ we can do a collapse attack like before, which will lead to the collisions (x, x) for x in $\mathbb{F}$.

But what is also possible is the permutation attack due to the symmetry in $\mathcal{C}$ *and* $\boldsymbol{O}$ under $\boldsymbol{B}$ for

$$\boldsymbol{B} = \begin{bmatrix} 0 & 1 & 0 & 0 \\ 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & 1 & 0 \end{bmatrix}. \tag{14}$$

We have $\mathcal{C}\boldsymbol{B} = \mathcal{C}$ as required in Conjeture 2.1 with $\boldsymbol{O}\boldsymbol{B} = \boldsymbol{O}$. So, if we take any $\boldsymbol{v}$ in $\mathsf{sol}(\mathcal{C})$, the chances are very high that it does not lie in the eigenspace for 1 of $\boldsymbol{B}$ so that $\boldsymbol{B}\boldsymbol{v} \neq \boldsymbol{v}$. Specifically, we can take any input (x, y) to $\mathcal{P}$ for $x \neq y$ and use $\mathcal{P}$ to compute the corresponding $\boldsymbol{v}$ in $\mathsf{sol}(\mathcal{C})$ by setting $\boldsymbol{v} = \begin{bmatrix} x & y & H(x) & H(y) \end{bmatrix}^\top$.

Then by Conjeture 2.2, we get the second preimage $\boldsymbol{I}\boldsymbol{B}\boldsymbol{v} = (y, x)$.

4.5. **A more interesting permutation attack.** Let $\mathcal{P}(x,y) = H(x) + H(y + H(x))$. The algebraic representation has two constraints:

$$\mathcal{C} = \{ [1\ 0\ 0\ 0] \mapsto [0\ 0\ 1\ 0], \quad [0\ 1\ 1\ 0] \mapsto [0\ 0\ 0\ 1] \} \qquad \boldsymbol{I} = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \end{bmatrix} \qquad \boldsymbol{O} = \begin{bmatrix} 0 & 0 & 1 & 1 \end{bmatrix}$$

The only possible non-identity basis change leaving $\mathcal{C}$ intact, exchanges these two queries. We can set up a system of 4 vector equations for $\boldsymbol{B}$ so that the first query and answer row vector is exchanged with the second when we apply $\boldsymbol{B}$. We find that this $\boldsymbol{B}$ is the unique solution to the system of equations:

$$\boldsymbol{B} = \begin{bmatrix} 0 & 1 & 1 & 0 \\ 1 & 0 & 0 & -1 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & 1 & 0 \end{bmatrix} \tag{15}$$

This $\boldsymbol{B}$ leaves both $\mathcal{C}$ and $\boldsymbol{O}$ unchanged (i.e. $\mathcal{C}\boldsymbol{B} = \mathcal{C}$ and $\boldsymbol{O}\boldsymbol{B} = \boldsymbol{O}$) so the conditions of the permutation attack are fulfilled.

An arbitrary vector from $\mathsf{sol}(\mathcal{C})$ is for example $\boldsymbol{v} = \begin{bmatrix} x & y & H(x) & H(y+H(x)) \end{bmatrix}^\top$ for some $x \neq y$ in $\mathbb{F}$. It is the execution vector for the input (x, y) to $\mathcal{P}$.

By applying $\boldsymbol{IB}$ to $\boldsymbol{v}$ we get the second preimage

$$\boldsymbol{IBv} = (y + H(x), x - H(y + H(x))). \tag{16}$$

This is not just a permutation of the two inputs, but a permutation between input and query base variables. The combined symmetry in $\mathcal{C}$ and $\boldsymbol{O}$ enable this attack. This symmetry and this attack are not obvious at first glance, which makes the abstract formulation of the permutation attack from Conjeture 2.2 valuable.

5. Merkle Dåmgard construction in Linicrypt

We want to see if the conjecture can correctly reproduce previous classifications of the PGV compression functions. Linicrypt can explain most of the previous categorizations, xcept for the Permutation attack and the 5 flawed backward attackable functions. TODO add a summary of the last section of my master's thesis that goes into detail.

Now I will try to tackle these last missing functions with the help of Conjeture 2.2. We use the variables $a, b, c, d, e, f \in \{0, 1\}$ as [2]:

$$f(h, m) = E(ch + dm, eh + fm) + ah + bm \tag{17}$$

If we see this as a Linicryp program, this is its canonical representation:

$$\boldsymbol{I} = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \end{bmatrix} \qquad \boldsymbol{O} = [a \;\; b \;\; 1]. \qquad \mathcal{C} = \{([e \;\; f \;\; 0], [c \;\; d \;\; 0], [0 \;\; 0 \;\; 1])\}$$

Now consider the Merkle Dåmgard construction with 2 calls to f. We will pass in the IV called h_0 as the first argument to the program.

$\underline{H_f^2(h_0; m_1, m_2)}$
$h_1 \coloneqq f(h_0, m_1)$
$h_2 \coloneqq f(h_1, m_2)$
return h_2

$\underline{H_f^2(h_0; m_1, m_2)}$
$y_1 \coloneqq E(ch_0 + dm_1, eh_0 + fm_1)$
$h_1 \coloneqq y_1 + ah_0 + bm_1$
$y_2 \coloneqq E(ch_1 + dm_2, eh_1 + fm_2)$
$\quad = E(cy_1 + cah_0 + cbm_1 + dm_2, ey_1 + eah_0 + ebm_1 + fm_2)$
$h_2 \coloneqq y_2 + ah_1 + bm_2$
$\quad := y_2 + ay_1 + a^2h_0 + abm_1 + bm_2$
return h_2

On the right I spelled out the Linicrypt program in the canonical basis $(h_0, m_1, m_2, y_1, y_2)$. In this case, the canonical basis is not the most pretty basis to work in for this program. If we were to add more calls to the compression functions the expressions get more and more complicated.

This is the algebraic representation of H_f^2 in this basis:

$$\boldsymbol{I} = \begin{bmatrix} 1 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 \end{bmatrix} \qquad \boldsymbol{O} = \begin{bmatrix} 1 & 0 & 0 & 0 & 0 \\ a^2 & ab & b & a & 1 \end{bmatrix}. \qquad \mathcal{C} = \left\{ \begin{matrix} ([e \;\; f \;\; 0 \;\; 0 \;\; 0], [c \;\; d \;\; 0 \;\; 0 \;\; 0], [0 \;\; 0 \;\; 0 \;\; 1 \;\; 0]) \\ ([ea \;\; eb \;\; f \;\; e \;\; 0], [ca \;\; cb \;\; d \;\; c \;\; 0], [0 \;\; 0 \;\; 0 \;\; 0 \;\; 1]) \end{matrix} \right\}$$

In this basis, it is very hard to see the originally very clean structure of the construction. This is due to two issues. First, we need to sort the basis variables in a way that mimics the structure of the construction. This would be $(h_0, m_1, y_1, m_2, y_2)$. In this basis (we switched y_1 and m_2) we have:

$$\boldsymbol{I} = \begin{bmatrix} 1 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 \end{bmatrix} \qquad \boldsymbol{O} = \begin{bmatrix} 1 & 0 & 0 & 0 & 0 \\ a^2 & ab & a & b & 1 \end{bmatrix}. \qquad \mathcal{C} = \left\{ \begin{matrix} ([e & f & 0 & 0 & 0],[c & d & 0 & 0 & 0],[0 & 0 & 1 & 0 & 0]) \\ ([ea & eb & e & f & 0],[ca & cb & c & d & 0],[0 & 0 & 0 & 0 & 1]) \end{matrix} \right\}$$

Now we can start to see the pattern that these matrices follow.

But, because we are allowed to represent the Linicrypt program in any basis, we can also just directly choose the basis $(h_0, m_1, h_1, m_2, h_2)$. Here the basis change idea really shines, because it helps to simplify the algebraic representation.

$$\boldsymbol{I} = \begin{bmatrix} 1 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 \end{bmatrix} \qquad \boldsymbol{O} = \begin{bmatrix} 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 \end{bmatrix}. \qquad \mathcal{C} = \left\{ \begin{matrix} ([e & f & 0 & 0 & 0],[c & d & 0 & 0 & 0],[-a & -b & 1 & 0 & 0]) \\ ([0 & 0 & e & f & 0],[0 & 0 & c & d & 0],[0 & 0 & -a & -b & 1]) \end{matrix} \right\}$$

The pattern is now clear, and we can easily see what the algebraic representation of H_f^n would be like for n arbitrarily high.

5.1. **Implementation of the permutation attack in SymPy.**

I used SymPy to find permutation matrices that attack these programs, the code is here [4]. I implemented only the permutation attack on H_f^2 for now. That is, we set up all the equations so that c_1 and c_2 are switched when we apply $\boldsymbol{B}$.

$$c_1\boldsymbol{B} = c_2 \quad \text{and} \quad c_2\boldsymbol{B} = c_1 \tag{18}$$

Because $\mathcal{C}$ is an **unordered** set, any solution $\boldsymbol{v}$ to $\mathcal{C}$ implies that $\boldsymbol{Bv}$ is a solution to $\mathcal{C}$ as well. If we add the constraints to $\boldsymbol{B}$ that the output matrix is untouched:

$$\boldsymbol{OB} = \boldsymbol{O} \tag{19}$$

and solve for such a $\boldsymbol{B}$ with any linear constraints solver (here I use SymPy), we can list out all the programs H_f^2 where such an attack is possible.

I tried to solve these equations by hand. It looks like it should be possible to find a nice characterization for what H_f^n are susceptible to this attack, but I just failed to solve the linear equations parametrized by a, b, c, d, e, f. The task is not to find a solution, but to find the conditions on a, b, c, d, e, f such that there is either no solution, one solution or many solutions.

There are 12 such programs. As a sanity check: The 5 programs that are marked with (P) in [5] and with (f) in [2] are contained in these 12. These are the permutation attacks where one can switch m_1 with m_2 in the input and get the same output. But the other seven additional permutation attacks can be broken down like this:

1. 3 degenerate programs (2 of type (-) and (a) and one of type (D) but it should be classified as (-))
2. 2 programs of type (D) and (b)
3. 2 programs of type (B) and (g)

Remark 5.1.1 (On the degenerate programs): One of the programs is marked as (D) in [5], but the compression function doesn't use the inputs independently, so it is essentially susceptible to the same attack as when it just uses one of the two inputs.

The permutation attack matrix $\boldsymbol{B}$ is not unique, there is a high dimensional subspace of possible matrices. This is a phenomenon I don't fully understand, but kind of what happens is that the matrix $\boldsymbol{B}$ maps one of the inputs to all its equivalent inputs. As the inputs to a degenerate program are not used independently, there is a subspace of inputs that "looks the same" to the program.

The matrix $\boldsymbol{B}$ then kind of maps one of these inputs to the rest of them. Interestingly, this is kind of a "hack", as the matrix kind of misuses some non-zero value from the solution vector to achieve this affine transformation with just a linear transformation.

I found multiple such examples during my master's thesis. I think it would be super interesting to allow for affine transformations, these would then easily capture degeneracy via this symmetry idea. But this means that one has to extend Linicrypt to work with affine combinations of its variables, not linear combinations. But it seems that essentially the same theory can be built.

Remark 5.1.2 (On the programs of type (D) and (b)): These are not marked as (P) in [5] because the authors specified an order of attacks, and (D) is higher up the list than (P).

Remark 5.1.3 (New fatal attacks on the programs of type (B) and (g)): This is what I find most interesting! The paper [2] described new 5 fatal attacks on the schemes previously classified as (B). These attacks all work in the same way, i.e. they cause two independent queries to E to collapse. The algorithmic implementation of Conjeture 2.2 found a **different** attack on two of them. This attack instead causes the two queries to happen in reversed order. By the conjecture, we get a single second-preimage to almost every possible input.

5.2. **New attack on 2 of the 5 (B) and (g) labeled schemes.**

The two compression functions of type (B) and (g) which are susceptible to a permutation attack are:

$$\begin{aligned} f_{43}(h,m) &= E(h+m,0)+h \\ f_{59}(h,m) &= E(h+m,h+m)+h \end{aligned} \tag{20}$$

As the attack is the same for both, let f be any of those two in the following. The basis change matrix with the properties for the permutation attack ($c_1\boldsymbol{B}=c_2, c_2\boldsymbol{B}=c_1, \boldsymbol{OB}=\boldsymbol{O}$) is for H_f^2:

$$\boldsymbol{B} = \begin{bmatrix} 1 & 0 & 0 & 0 & 0 \\ -1 & 0 & 1 & 1 & 0 \\ 1 & 0 & -1 & 0 & 1 \\ 0 & 1 & 1 & 0 & -1 \\ 0 & 0 & 0 & 0 & 1 \end{bmatrix} \tag{21}$$

Let (h_0, m_1, m_2) be any input to H_f^2. The conjecture says it should have a second preimage:

$$(h_0', m_1', m_2') \coloneqq (h_0, -h_0+h_1+m_2, m_1+h_1-h_2) \tag{22}$$

Note, that $h_0' = h_0$ is a side effect of putting this IV in the input and output of H_f^2.

Here we do the lengthy calculation for f_{43} to convince ourselves that it is indeed a second preimage. Everything cancels out and we get the same output.

$$
\begin{aligned}
&\underline{H^2_{f_{43}}(h_0, -h_0 + h_1 + m_2, m_1 + h_1 - h_2)} \\
&y'_1 := \quad E(h'_0 + m'_1, 0) \\
&\quad = E(h_0 - h_0 + h_1 + m_1, 0) \\
&\quad = \quad E(h_1 + m_1, 0) = y_2 \\
&h'_1 := y'_1 + h'_0 = y_2 + h_0 \\
&y'_2 := \quad E(h'_1 + m'_2, 0) \\
&\quad = E(y_2 + h_0 + m_1 + h_1 - h_2, 0) \\
&\quad = \quad E(y_2 + h_0 + m_1 - y_2, 0) \\
&\quad = \quad E(h_0 + m_1, 0) = y_1 \\
&h'_2 := \quad y'_2 + h'_1 \\
&\quad = y_1 + y_2 + h_0 = h_2 \\
&\textbf{return } h'_2
\end{aligned}
$$

Now, what are these second preimages? When are they proper second preimages, and not the same preimage again? If we solve this equation for m_1 and m_2

$$(h_0, m_1, m_2) := (h_0, -h_0 + h_1 + m_2, m_1 + h_1 - h_2), \tag{23}$$

we get exactly the collisions that [2] found. They found that all the inputs of this form

$$(h_0, h_0 + c, E(c, c) + h_0 + c), \tag{24}$$

for $c \in \mathbb{F}$ arbitrary, will collapse the queries and lead $H^2_{f_{43}}$ to output h_0. That leads to the following conclusion, which is still very puzzling to me: For the program $H^2_{f_{43}}$ every input is either:
1. In the set of collisions found by [2],
2. or it has a single second preimage generated by the permutation matrix $\boldsymbol{B}$.

This means we broke second-preimage resistance for every input now, not only for the inputs in the specific subspace described by the attack in [2]. But there is no freedom in choosing the second preimage for H^2_f.

This also hints towards some missing deeper theory that explains this. My guess is that these things are explained by something like the orbit and stabilizer from some kind of representation. And I hope that by formulating the Linicrypt theory in some way making use of the language from group actions and representations we can

- simplify the attack proofs,
- simplify the formulation of all possible attacks (i.e. optimize the conjecture),
- better understand the core of what a Linicrypt program is
- and hopefully find an easy proof for the security side.

5.2.1. *More inputs to H.*

I checked with the SymPy program that this attack holds up for dimensions up to 6 (checking all combinations of a, b, c, d, e, f takes a couple of seconds then already…). The attack is a permutation of the n constraints like this:

$$c_1 \to c_2 \to ... \to c_n \to c_1, \tag{25}$$

where the arrows represent the application of the permutation matrix $\boldsymbol{B}$.

Therefore each input $\boldsymbol{i}_1$ to $H^n_{f_{43}}$ has $n-1$ second preimages described by:

$$\begin{aligned} \boldsymbol{i}_2 &= \boldsymbol{B}^1\boldsymbol{i} \\ \boldsymbol{i}_3 &= \boldsymbol{B}^2\boldsymbol{i} \\ &... \\ \boldsymbol{i}_n &= \boldsymbol{B}^n\boldsymbol{i} = \boldsymbol{B}^0\boldsymbol{i} = \boldsymbol{i} \end{aligned} \tag{26}$$

6. Security Side

6.1. **Ideas for the proof.** Just some random notes.

General proof idea of Conjeture 2.2:

- (a), (b) or (c) ⇒ attack is done
- Reverse: Assume adversary finds collisions $\boldsymbol{v} \neq \boldsymbol{v}'$ in $\mathsf{sol}(\mathcal{C})$ under $\boldsymbol{O}$ ($\boldsymbol{O}\boldsymbol{v} = \boldsymbol{O}\boldsymbol{v}'$), then (a), (b) or (c)

We have:

- Protocol of $\mathcal{A}$ interacting with orcale H (or E or Π) with N queries
- From $\boldsymbol{v}$ and $\boldsymbol{v}$' and the protocol we can extract $T, T' : \mathcal{C} \to [N]$ when each constraint was determined.
- We define $\mathsf{finish}\ (c) = \mathsf{max}(T(c), T'(c))$ for each c in $\mathcal{C}$.
- We define $\mathcal{C}_{fix}$ to be the maximal subset such that $T = T'$ on $\mathcal{C}_{fix}$ and $\mathcal{C}_{cs} = \mathcal{C} \setminus \mathcal{C}_{fix}$

6.1.1. *Base case.* T and T' are each injective and $T(\mathcal{C}_{cs}) \cap T'(\mathcal{C}_{cs}) = \{\}$

This is the case that they have in the unique nonces paper. Then finish is injective.

6.1.2. *Case: Collapse of queries.*

6.1.2.1. *Easy case first: simultaneous collapse.*

Assume T and T' is not injective, so there is a $i \neq j$ such that $T(c_i) = T\big(c_j\big)$ and $T'(c_i) = T'\big(c_j\big)$. By definition of T and T' we have $c_i\boldsymbol{v} = c_j\boldsymbol{v}$ and $c_i\boldsymbol{v}' = c_j\boldsymbol{v}'$.

Here $c\boldsymbol{v}$ just means $(\boldsymbol{Q}\boldsymbol{v}, \boldsymbol{a}\boldsymbol{v})$ in the random oracle case.

Now I want to find an injective matrix $\boldsymbol{T}$ in $\mathbb{F}^{d' \times d}$ through which c_i and c_j collapse. And the $\mathcal{P}\boldsymbol{T}$ (meaning $(\boldsymbol{I}\boldsymbol{T}, \boldsymbol{O}\boldsymbol{T}, \mathcal{C}\boldsymbol{T})$) should then be a pseudo-algebraic representation for which we can build an attack by using $\mathcal{A}$.

The natural injective matrix $\boldsymbol{T}$ which collapses c_i and c_j is a matrix for which the columns are a basis of $\mathsf{ker}\big(c_i - c_j\big)$. This matrix is injective because the columns for a basis of the subspace. In the following we use the basis $\boldsymbol{v}_1, ..., \boldsymbol{v}_{d'}$.

The previous idea of taking $\boldsymbol{v}$ and $\boldsymbol{v}'$ as the columns is a much smaller matrix in general.

The key with this $\boldsymbol{T}$ is that we can easily translate the attack on $\mathcal{P}$ to an attack on $\mathcal{P}\boldsymbol{T}$. First, it collapses c_i and c_j:

$$c_i\boldsymbol{T} = c_i[\boldsymbol{v}_1\ \ ...\ \ \boldsymbol{v}_{d'}] = c_j[\boldsymbol{v}_1\ \ ...\ \ \boldsymbol{v}_{d'}] = c_j\boldsymbol{T} \tag{27}$$

So $\mathcal{C}\boldsymbol{T}$ has at least one constraint less, as these two have collapsed. Also, $\boldsymbol{v}$ and $\boldsymbol{v}'$ are in the image of $\boldsymbol{T}$, because its columns form a basis of $\mathsf{ker}(c_i - c_j)$.

So there are preimages $\boldsymbol{w}$ and $\boldsymbol{w}'$ to $\boldsymbol{v}$ and $\boldsymbol{v}'$ means by Lemma 1.1

$$\boldsymbol{w} \in \mathsf{sol}(\mathcal{C}\boldsymbol{T}) \iff \boldsymbol{T}\boldsymbol{w} \in \mathsf{sol}(\mathcal{C}), \tag{28}$$

that $\boldsymbol{w}$ and $\boldsymbol{w}'$ are in $\mathsf{sol}(\mathcal{C}\boldsymbol{T})$. The reason for finding these

Now we have reduced the attack $\mathcal{A}$ to an attack on $\mathcal{P}\boldsymbol{T}$:
- $\boldsymbol{w} \neq \boldsymbol{w}'$ in $\mathbb{F}^{d'}$ are solutions to $\mathcal{C}\boldsymbol{T}$
- Also $\boldsymbol{O}\boldsymbol{T}\boldsymbol{w} = \boldsymbol{O}\boldsymbol{T}\boldsymbol{w}'$
- We can apply the conjecture for the lower dimensional case $\mathcal{P}\boldsymbol{T}$

The key thing that is missing here is that $\mathcal{P}\boldsymbol{T}$ should be an algebraic representation up to basis change of some program, so that we can apply the conjecture for a lower dimension by induction.

Key issue:
- $\mathcal{C}\boldsymbol{T}$ needs to also be solvable (for the examples it always is)

Ideas:
- If $\mathcal{C}\boldsymbol{T}$ is not solvable, then it would have been hard for the attacker to collapse those two queries.
- This only works if we look at the whole set of collapsed queries. It might be hard to collapse c_i and c_j without collapsing some other queries, but easy once you do.
- My guess nr 1: The pairs of constraints that $\mathcal{A}$ could collapse need to be "independent" (to be defined).
- Guess nr 2: When they are independent there would also exist a matrix $\boldsymbol{B}$ permuting these queries. So maybe that is a line of attack.

Lets assume $\mathcal{C}\boldsymbol{T}$ is not solvable.

New idea: But the attacker has already computed the states $\boldsymbol{w}$ and $\boldsymbol{w}'$ in $\mathsf{sol}(\mathcal{C}\boldsymbol{T})$. If we can prove that it is hard to find solutions to unsolvable constraints, then we can use that to complete this step of the proof.

6.2. **Unsolvability of constraints.**

I want to reformulate the definition of solvable constraints to make it nicer to work with.

Definition 6.2.1 (Solvable constraints): Let $\mathcal{C}$ be a finite set of constraints of dimension d, and let $\mathcal{F}_0$ be a subspace of $\left(\mathbb{F}^d\right)^*$. $\mathcal{C}$ is **solvable fixing** $\mathcal{F}_0$ if there exists an ordering $(c_1, ..., c_n)$ of $\mathcal{C}$ such that the condition below is fulfilled. We define for all $i = 1, ..., n$ the spaces $\mathcal{F}_i = \mathcal{F}_{i-1} + \langle c_i \rangle$.

Solvability condition: $\boldsymbol{a}_i \notin \mathcal{F}_{i-1} + \langle \boldsymbol{Q}_i \rangle \quad$ where $(\boldsymbol{Q}_i, \boldsymbol{a}_i) \coloneqq c_i$

We call $(c_1, ..., c_n)$ a solution ordering of $\mathcal{C}$ fixing $\mathcal{F}_0$. If $\mathcal{F}_0 = \{0\}$ we just say $\mathcal{C}$ is solvable, dropping the "fixing" notation.

In the ideal cipher model, we only have to change the solvability condition. It will allow either $\begin{bmatrix}\boldsymbol{k}\\ \boldsymbol{x}\end{bmatrix}$ or $\begin{bmatrix}\boldsymbol{k}\\ \boldsymbol{y}\end{bmatrix}$ to be the query $\boldsymbol{Q}$.

Definition 6.2.2 (Deterministically solvable): Let $\mathcal{C}$ be a solvable set of constraints with solution ordering $(c_1, ..., c_n)$. $\mathcal{C}$ is **deterministically solvable fixing** $\mathcal{F}_0$ if $\left(\mathbb{F}^d\right)^* = \mathcal{F}_0 \oplus \langle \boldsymbol{a}_1, ..., \boldsymbol{a}_n \rangle$. In other words, we require $d = \dim(\mathcal{F}_0) + n$ in addition to the solvability condition.

This dimension condition forces $\langle \boldsymbol{Q}_i \rangle \in \mathcal{F}_{i-1}$, which is the condition we previously used in the definition.

Deterministically solvable is closest to the definition of collision structure in [1]. Collision structure is more complicated because it also includes splitting up the constraints into a fixed part and a deterministically solvable part. Also, deterministically solvable constraints are in a one-to-one relationship with Linicrypt programs, up to basis change. That means every Linicrypt program has a deterministically solvable set of constraints fixing $\mathcal{F}_0 = \mathsf{rowsp}(\boldsymbol{I})$. On the contrary, for every solvable set of constraints, there exists a basis such that the set of constraints is the algebraic representation of a Linicrypt program.

6.2.1. *Solvable security game.*
Now we can define a security game for finding solutions to a set of constraints $\mathcal{C}$. The adversary $\mathcal{A}$ gets access to the constraints $\mathcal{C}$ of dimension d, a description of the space $\mathcal{F}_0$ and the oracle H. Then the game randomly samples a $\boldsymbol{i}$ in $\mathbb{F}^d$ and passes it to $\mathcal{A}$. This vector represents the input to a Linicrypt program. It wins the game by outputting a $\boldsymbol{v} \in \mathsf{sol}(\mathcal{C})$ with $\boldsymbol{v} - \boldsymbol{i} \in \mathsf{ker}(\mathcal{F}_0)$.

The probability of $\mathcal{A}$ winning this game is written as $\mathrm{SOLadv}[\mathcal{A}, \mathcal{C}, \mathcal{F}_0]$.

A useful fact is that we can avoid working with $\mathcal{F}_0$ because of the following proposition.

Proposition 6.2.1.1 (Collapse of input): Let $\mathcal{C}$ be a set of constraints of dimension d and $\mathcal{F}_0$ a subspace of $\left(\mathbb{F}^d\right)^*$. We define the embedding map $\boldsymbol{L} : \mathsf{ker}(\mathcal{F}_0) \hookrightarrow \mathbb{F}^d$. Then the following are equivalent:
1. $\mathcal{C}$ is solvable fixing $\mathcal{F}_0$
2. $|\mathcal{C}\boldsymbol{L}| = |\mathcal{C}|$ and $\mathcal{C}\boldsymbol{L}$ is solvable (fixing $\{0\}$)

In Linicrypt program terms, this proposition relates a Linicrypt program with its corresponding inputless Linicrypt program where all inputs are set to 0.

Proof: We define $\mathcal{C}$, $\mathcal{F}_0$, and $\boldsymbol{L}$ as in the proposition statement.

Remark 6.2.1.1: The linear map $\boldsymbol{L}$ induces corresponding map on the dual spaces $\boldsymbol{L}^* : \left(\mathbb{F}^d\right)^* \to \left(\mathsf{ker}(\mathcal{F}_0)^*\right)$. With a bit of abuse of notation, we can extend this map $\boldsymbol{L}^*$ to act on constraints. Constraints are tuples of matrices. The rows of the matrices are in $\left(\mathbb{F}^d\right)^*$, so we can let $\boldsymbol{L}^*$ act on these structures elementwise. Then $\boldsymbol{L}^*((\boldsymbol{Q}, \boldsymbol{a})) = (\boldsymbol{Q}\boldsymbol{L}, \boldsymbol{a}\boldsymbol{L})$.

TODO: This is useful in multiple places, so it should be discussed outside this proof.

We will first prove a useful fact. Let $\mathcal{C} = (c_1, ..., c_n)$ be an ordering and $\mathcal{C}\boldsymbol{L} = (c_1\boldsymbol{L}, ..., c_n\boldsymbol{L})$ the corresponding ordering. Regardless of whether the solvability condition is fulfilled, we can define the sequence of subspaces $\mathcal{F}_i$ from the ordering of $\mathcal{C}$ and $\mathcal{F}'_i$ from the ordering of $\mathcal{C}\boldsymbol{L}$. We set $\mathcal{F}'_0 = \boldsymbol{L}^*(\mathcal{F}_0) = \{0\}$. Then we have $\mathcal{F}'_i = \boldsymbol{L}^*(\mathcal{F}_i)$ by the following inductive argument:

$$\begin{aligned}\mathcal{F}'_i &= \mathcal{F}'_{i-1} + \langle c_i \boldsymbol{L} \rangle \\ &= \boldsymbol{L}^*(\mathcal{F}_{i-1}) + \langle c_i \boldsymbol{L} \rangle \\ &= \boldsymbol{L}^*(\mathcal{F}_{i-1} + \langle c_i \rangle) \\ &= \boldsymbol{L}^*(\mathcal{F}_i)\end{aligned} \tag{29}$$

We first assume $\mathcal{C}\boldsymbol{L}$ has a solution ordering $\mathcal{C}\boldsymbol{L} = (c_1\boldsymbol{L}, ..., c_n\boldsymbol{L})$. This is a well-defined ordering because $\boldsymbol{L}$ didn't collapse any constraints by $|\mathcal{C}\boldsymbol{L}| = |\mathcal{C}|$. This gives us the ordering $\mathcal{C} = (c_1, ..., c_n)$. Assume towards a contradiction that there is an i with $\boldsymbol{a}_i \in \mathcal{F}_{i-1} + \langle \boldsymbol{Q}_i \rangle$. We can apply $\boldsymbol{L}^*$ to this equation to get

$$\boldsymbol{a}_i\boldsymbol{L} \in \mathcal{F}'_{i-1} + \langle \boldsymbol{Q}_i\boldsymbol{L} \rangle, \tag{30}$$

a contradiction with the assumed solution ordering of $\mathcal{C}\boldsymbol{L}$.

Now we prove the reverse direction and assume $\mathcal{C}$ is solvable fixing $\mathcal{F}_0$ with ordering $(c_1, ..., c_n)$. First, we show that $|\mathcal{C}| = |\mathcal{C}\boldsymbol{L}|$. Assume $c_i\boldsymbol{L} = c_j\boldsymbol{L}$ for an $i < j$. This implies in particular that $0 = (\boldsymbol{a}_i - \boldsymbol{a}_j)\boldsymbol{L}\boldsymbol{w}$ for all $\boldsymbol{w} \in \mathsf{ker}(\mathcal{F}_0)$. Therefore $\mathsf{ker}(\mathcal{F}_0) \subseteq \mathsf{ker}(\boldsymbol{a}_i - \boldsymbol{a}_j)$. It follows that $\boldsymbol{a}_i - \boldsymbol{a}_j \in \mathcal{F}_0$ and $\boldsymbol{a}_j \in \mathcal{F}_0 + \langle \boldsymbol{a}_i \rangle$. This directly contradicts with the solvability condition $\boldsymbol{a}_j \notin \mathcal{F}_{j-i} + \langle \boldsymbol{Q}_j \rangle$ because $\mathcal{F}_0 + \langle \boldsymbol{a}_i \rangle \subseteq \mathcal{F}_{j-1}$.

Now that we have $|\mathcal{C}| = |\mathcal{C}\boldsymbol{L}|$ we get a well-defined ordering $\mathcal{C}\boldsymbol{L} = (c_1\boldsymbol{L}, ..., c_n\boldsymbol{L})$. Let us assume this is not a solution ordering and we have for some i that $\boldsymbol{a}_i\boldsymbol{L}$ is contained in $\mathcal{F}'_{i-1} + \langle \boldsymbol{Q}_i\boldsymbol{L} \rangle$. It follows that $\boldsymbol{L}^*(\boldsymbol{a}_i) = \boldsymbol{L}^*(\boldsymbol{b})$ for some $\boldsymbol{b} \in \mathcal{F}_{i-1} + \langle \boldsymbol{Q}_i \rangle$. As before, we can move the equations around to get this sequence of implications:

$$\begin{aligned}&(\boldsymbol{a}_i - \boldsymbol{b})\boldsymbol{L}\boldsymbol{w} = 0 \quad \text{for any} \quad \boldsymbol{w} \in \mathsf{ker}(\mathcal{F}_0) \\ \implies &\mathsf{ker}(\mathcal{F}_0) \subseteq \mathsf{ker}(\boldsymbol{a}_i - \boldsymbol{b}) \\ \implies &\boldsymbol{a}_i - \boldsymbol{b} \in \mathcal{F}_0 \\ \implies &\boldsymbol{a}_i \in \mathcal{F}_{i-1} + \langle \boldsymbol{Q}_i \rangle\end{aligned} \tag{31}$$

The last equation would be a contradiction to the solution ordering $(c_1, ..., c_n)$. □

This proposition is useful because we can then translate statements about solvability fixing $\{0\}$ to the more general version. In some sense, we only need to understand inputless Linicrypt programs to understand all Linicrypt programs.

Theorem 6.2.1.1 (Unsolvable constraints – wrong): Let $\mathcal{C}$ be a set of constraints of dimension d which are not solvable. Every program $\mathcal{A}$ making N request to the oracle has a winning probability bounded by

$$\mathrm{SOL}\mathsf{adv}[\mathcal{A}, \mathcal{C}] < \frac{N}{|\mathbb{F}|}. \tag{32}$$

Remark 6.2.1.2: This is why I thought it could work: The key idea of why this should work is that the condition of being unsolvable is one of the form "vector is contained in subspace". This cannot

be broken by a linear map, even when it is not injective. The vector after the mapping stays in the subspace after the mapping.

UPDATE: Yes, but, if the linear map collapses the problematic constraints, then it doesn't matter that it stays in the subspace.

Remark 6.2.1.3: **Good news and bad news.** Bad news: It does not work like this. Counterexample:

$$\mathcal{C} = \{[1\ \ 0\ \ 0] \mapsto [0\ \ 0\ \ 1], [0\ \ 1\ \ 0] \mapsto [0\ \ 0\ \ 1]\} \tag{33}$$

is unsolvable. But with $\boldsymbol{L} = \begin{bmatrix} 1 & 0 \\ 1 & 0 \\ 0 & 1 \end{bmatrix}$, the set $\mathcal{C}\boldsymbol{L} = \{[1\ \ 0] \mapsto [0\ \ 1]\}$ is solvable. So solutions of $\mathcal{C}\boldsymbol{L}$ can be mapped to solutions of $\mathcal{C}$, i.e. vectors of the form $\begin{bmatrix} x \\ x \\ H(x) \end{bmatrix}$ for $x \in \mathbb{F}$.

The good news is, that it means we still have room for an NP problem.

Idea to try to save it: Call a set of constraints $\mathcal{C}$ completely unsolvable if $\mathcal{C}\boldsymbol{L}$ is unsolvable for all linear maps $\boldsymbol{L}$. Then we might be able to prove the Theorem for completely unsolvable sets. This is surprisingly similar in structure to the original conjecture.

Caveat: This means we still need at least an algorithm for determining if a set of constraints is completely unsolvable. I would guess this problem is then NP-hard.

Definition 6.2.1.1 (Completely unsolvable): A set of constraints $\mathcal{C}$ is called **completely unsolvable** if $\mathcal{C}\boldsymbol{L}$ is unsolvable for every linear map $\boldsymbol{L}$.

The relevant maps are those which collapse constraints. It is probably enough to consider only the maps $\boldsymbol{L} : \mathsf{ker}(c_i - c_j) \hookrightarrow \mathbb{F}^d$ recursively.

Theorem 6.2.1.2 (Unsolvable constraints – not yet useful): Let $\mathcal{C}$ be a set of constraints of dimension d which are completely unsolvable. Every program $\mathcal{A}$ making N request to the oracle has a winning probability bounded by

$$\mathrm{SOLadv}[\mathcal{A}, \mathcal{C}] < \frac{N}{|\mathbb{F}|}. \tag{34}$$

I want to try to refine this theorem further, because like this it is not fully useful. The issues are:

- In the proof of the conjecture, we have $\mathcal{C}\boldsymbol{L}$, which we want to be either solvable or very hard to solve
- If it is not solvable, it might still not be completely unsolvable as in the theorem
- What we want is for it to be solvable in a subspace (the one where a solution was found), or to be difficult to solve in that subspace

We refine the solvability definition.

Definition 6.2.1.2 (solvable – fixing and outside): $\mathcal{C}$ is solvable outside of a subspace W of $\mathbb{F}^d$ fixing a subspace $\mathcal{F}$ of $\left(\mathbb{F}^d\right)^*$ if there exists a linear map $f : \mathbb{F}^{d'} \to \mathbb{F}^d$ with $\mathsf{im}(f) \not\subseteq W$ s.t. $f^*(\mathcal{C})$ is solvable fixing $f^*(\mathcal{F})$.

If $W = \{0\}$ we will just say $\mathcal{C}$ is solvable fixing $\mathcal{F}$. If $\mathcal{F} = \{0\}$ we will just say $\mathcal{C}$ is solvable outside of W.

This language of being solvable outside of a subspace is useful in describing collision resistance. There we are looking for solutions $\boldsymbol{v}$ and $\boldsymbol{v}'$ of constraints, under the extra condition that $\boldsymbol{v} \neq \boldsymbol{v}'$. This condition can be encoded with this new definition.

TODO more explicit.

Now we can define a security game for finding solutions to a set of constraints $\mathcal{C}$. The adversary $\mathcal{A}$ gets access to the constraints $\mathcal{C}$ of dimension d, the subspaces $\mathcal{F}$ and W of $\left(\mathbb{F}^d\right)^*$ and $\mathbb{F}^d$ respectively. The adversary $\mathcal{A}$ also gets acces to the oracle H such that we can record its queries. Then the game randomly samples a vector $\boldsymbol{i}$ in $\mathbb{F}^d$ (representing the Linicrypt program input) and passes it to $\mathcal{A}$. It wins the game by outputting a $\boldsymbol{v} \in \mathsf{sol}(\mathcal{C})$ which fulfills $\boldsymbol{v} - \boldsymbol{i} \in \mathsf{ker}(\mathcal{F})$ and $\boldsymbol{v} \notin W$.

The probability of $\mathcal{A}$ winning this game is written as $\mathsf{SOLadv}[\mathcal{A}, \mathcal{C}, \mathcal{F}, W]$.

Theorem 6.2.1.3 (Unsolvable constraints V2): Let $\mathcal{C}$ be a set of constraints of dimension d and W a subspace of $\mathbb{F}^d$. Assume $\mathcal{C}$ is not solvable outside of W. Every program $\mathcal{A}$ making N request to the oracle has a winning probability bounded by

$$\mathsf{SOLadv}[\mathcal{A}, \mathcal{C}, \{0\}, W] < \frac{N}{|\mathbb{F}|}. \tag{35}$$

Proof Sketch: We record the queries by $\mathcal{A}$ in the function $T : \mathcal{C} \to [N]$ which maps each constraint onto the time when it was determined by a call to H. We assume $\mathcal{A}$ is successful, and outputs a $\boldsymbol{v}$ in $\mathsf{sol}(\mathcal{C})$ with $\boldsymbol{v} \notin W$.

This T might not be injective. We attempt an inductive proof over n the number of constraints in $\mathcal{C}$. So we assume the theorem holds for all sets $\mathcal{C}$ with $|\mathcal{C}| < n - 1$.

When T is injective, we can do the core step of the main proof from [1]. This is the one where we show the result of a call to H was already determined beforehand, via an equation where the left is randomly chosen, and the right is a linear combination of known values. This means that $\mathcal{A}$ was very lucky if we assume H is a random oracle. This equation can be derived from assuming $\mathcal{C}$ is not solvable outside of W. Because this means, in particular, $\mathcal{C}$ is not solvable itself. No matter what ordering of $\mathcal{C}$ we choose, for some i the negated solvability condition implies a linear equation with $\boldsymbol{a}_i\boldsymbol{v}$ on the left side, and previously determined values on the right side.

Now assume T is not injective. Then let c_i and c_j be two different constraints that are determined at the same time, i.e. $T(c_i) = T(c_j)$. Then $\boldsymbol{v}$ is in $\mathsf{ker}(c_i - c_j)$. Also, because $\boldsymbol{v} \notin W$, we know that $\mathsf{ker}(c_i - c_j)$ is not contained in W.

We can define the linear map $f : \mathsf{ker}(c_i - c_j) \hookrightarrow \mathbb{F}^d$ which is just the embedding. So this map goes from a smaller state space to the state space of our constraints $\mathcal{C}$.

This map induces a map on the dual spaces $f^* : \left(\mathbb{F}^d\right)^* \to \mathsf{ker}\left(c_i - c_j\right)^*$, i.e. a map acting on variables of a Linicrypt program, or here, the components of the constraints. So we can use it to map our constraints to a different set of constraints of smaller dimension $f^*(\mathcal{C}) = \mathcal{C}f$. Because the i'th and j'th constraint collapse under f we have $|f^*(\mathcal{C})| \leq |\mathcal{C}| - 1$

Because $\boldsymbol{v}$ is in the image of f and outside of W, the adversary $\mathcal{A}$ has thus found a solution $\boldsymbol{w}$ to $f^*(\mathcal{C})$ where $f\boldsymbol{w} = \boldsymbol{v}$ and also $\boldsymbol{w} \notin f^{-1}(W)$.

But $f^*(\mathcal{C}) \coloneqq \mathcal{C}f$ is not solvable outside of $f^{-1}(W)$.

Assume it was, i.e. there is a $g : U \to \mathsf{ker}\left(c_i - c_j\right)$ such that $\mathcal{C}fg$ is solvable and $\mathsf{im}(g) \nsubseteq W$. Then fg is a map as in the definition of $\mathcal{C}$ being solvable outside of W. We assumed this was not the case in the theorem statement.

By induction, we can apply the Theorem for $f^*(\mathcal{C})$ and $f^{-1}(W)$ to get

$$\mathrm{SOLadv}[\mathcal{A}, \mathcal{C}] \leq \mathrm{SOLadv}[\mathcal{A}, f^*(\mathcal{C})] \leq \frac{N}{|\mathbb{F}|}. \tag{36}$$

Remark 6.2.1.4: Here the actual factor on the right-hand side has to be different I think.

The base case for the induction is a set of constraints with just a single unsolvable constraint $\mathcal{C} = \{c\} = \{(\boldsymbol{Q}, \boldsymbol{a})\}$. No matter the dimensionality of c, this set is completely unsolvable. This is because $a \in \langle \boldsymbol{Q} \rangle$ implies $\boldsymbol{aL} \in \langle \boldsymbol{QL} \rangle$ for any linear map $\boldsymbol{L}$. For this singleton set any map $T : \mathcal{C} \to [N]$ is injective and we can use the original proof for that case. □

Maybe we can use a general theorem like the unsolvability theorem to replace the proofs for collision structure and second preimage characterization. The key step in the proof of unsolvability is the same as in those proofs.

Let $\mathcal{P}$ be a Linicrypt program. We can duplicate its algebraic representation such that the vectors are completely separate. One copy has all zeros on the right of the row vectors, and the other has all zeros on the left. Then we can merge the algebraic representations, by doing a union on the constraints, adding the input spaces, and concatenating the output matrix. Call this program $\mathcal{P}_{\text{join}}$. We can look at a map f that has as its image the states in $\mathbb{F}^{2d}$ where the output of $\mathcal{P}_{\text{join}}$ has both halves equal. i.e.:

$$\begin{gathered} \boldsymbol{O}_{\text{join}} = \begin{bmatrix} \boldsymbol{O}_1 \\ \boldsymbol{O}_2 \end{bmatrix}, \quad \mathcal{C}_{\text{join}} = \mathcal{C}_1 \cup \mathcal{C}_2, \quad \mathcal{F}_{\text{join}} = \mathcal{F}_1 + \mathcal{F}_2 \\ f : \mathsf{ker}(\boldsymbol{O}_1 - \boldsymbol{O}_2) \hookrightarrow \mathbb{F}^{2d} \end{gathered} \tag{37}$$

We define $\boldsymbol{P}_1$ and $\boldsymbol{P}_2$ to be the projections from F^{2d} to either the first n dimensions or the second n. So $\boldsymbol{P}_1$ and $\boldsymbol{P}_2$ recover the solutions to $\mathcal{C}$ from the first n variables, or the second n variables respectively.

Then finding a $\boldsymbol{v} \in \mathsf{sol}\left(\mathcal{C}_{\text{join}} f\right)$ with $\boldsymbol{v} \notin \mathsf{ker}(\boldsymbol{P}_1 - \boldsymbol{P}_2)$ means finding collision for $\mathcal{P}$.

Corollary 6.2.1.1 (Collisions in general): $\mathcal{C}_{\text{join}} f$ is solvable outside of $\mathsf{ker}(\boldsymbol{P}_1 - \boldsymbol{P}_2)$ is equivalent to $\mathcal{P}$ being susceptible to an easy attack against collision resistance.

Now that we have a formal definition of solutions outside of a subspace, we can try to prove it.

Proof Sketch: Assume that $\mathcal{A}$ finds a solution easily. Easy means with a higher probability than in the unsolvability theorem. Then that theorem gives us a map $g : U \to \mathsf{ker}(\boldsymbol{O}_1 - \boldsymbol{O}_2)$ such that $\mathcal{C}_{\text{join}} fg$ is solvable and $\mathsf{im}(g) \not\subseteq \mathsf{ker}(\boldsymbol{P}_1 - \boldsymbol{P}_2)$. So a solution to $\mathcal{C}_{\text{join}} fg$ is a solution to $\mathcal{C}_{\text{join}}$ when mapped by fg. Let $\boldsymbol{v}$ be such a solution to $\mathcal{C}_{\text{join}}$ with $\boldsymbol{v} \notin \mathsf{ker}(\boldsymbol{P}_1 - \boldsymbol{P}_2)$. By construction then $\boldsymbol{P}_1\boldsymbol{v}$ and $\boldsymbol{P}_2\boldsymbol{v}$ are solutions to $\mathcal{C}$ with $\boldsymbol{P}_1\boldsymbol{v} \neq \boldsymbol{P}_2\boldsymbol{v}$.

On the reverse side, assume there is such an g as above because $\mathcal{C}_{\text{join}} f$ is solvable outside of $\mathsf{ker}(\boldsymbol{P}_1 - \boldsymbol{P}_2)$. Then computing solutions to $\mathcal{C}_{\text{join}}$ takes at most $2n$ queries to H. As before, a solution outside of $\mathsf{ker}(\boldsymbol{P}_1 - \boldsymbol{P}_2)$ leads to a collision. This means we have found an attack on collision resistance. □

Remark 6.2.1.5: In this proof, in the last step, I need to actually find a solution outside of W just given the solution ordering. There we need to be a bit careful. Maybe the oracle H is not random, and actually causes all solutions to lie on the subspace W again. But for a random W the solution space is never contained in a subspace.

6.2.2. *TODOs this section.*

- Write down lots of examples to see how this works in all the special cases
- Prove that a solution ordering like the above leads to a case from the conjecture
- Pederson Hash

6.2.3. *Examples.*

In [1] the authors describe a Linicrypt program which is not collision resistant due to collapse of constraints.

$\underline{\mathcal{P}(x,y)}$
$a_1 \coloneqq H(x)$
$a_2 \coloneqq H(a_1)$
$a_3 \coloneqq H(y)$
return $a_2 - a_3$

$\underline{\text{Algebraic Representation of } \mathcal{P}(x,y)}$

$$\boldsymbol{O} \coloneqq \begin{bmatrix} 0 & 0 & 0 & 1 & -1 \end{bmatrix}$$

$$\boldsymbol{I} \coloneqq \begin{bmatrix} 1 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 \end{bmatrix}$$

$$\mathcal{C} = \{ [1\ 0\ 0\ 0\ 0] \mapsto [0\ 0\ 1\ 0\ 0],\ [0\ 0\ 1\ 0\ 0] \mapsto [0\ 0\ 0\ 1\ 0],\ [0\ 1\ 0\ 0\ 0] \mapsto [0\ 0\ 0\ 0\ 1] \}$$

We define $\boldsymbol{O}_1, \boldsymbol{O}_2, \mathcal{C}_1, \mathcal{C}_2, \boldsymbol{P}_1$ and $\boldsymbol{P}_2$ as in the corollary.

$$\begin{aligned} \mathcal{C}_{\text{join}} = \{ & [1\ 0\ 0\ 0\ 0\ 0\ 0\ 0\ 0\ 0] \mapsto [0\ 0\ 1\ 0\ 0\ 0\ 0\ 0\ 0\ 0], \\ & [0\ 0\ 1\ 0\ 0\ 0\ 0\ 0\ 0\ 0] \mapsto [0\ 0\ 0\ 1\ 0\ 0\ 0\ 0\ 0\ 0], \\ & [0\ 1\ 0\ 0\ 0\ 0\ 0\ 0\ 0\ 0] \mapsto [0\ 0\ 0\ 0\ 1\ 0\ 0\ 0\ 0\ 0], \\ & [0\ 0\ 0\ 0\ 0\ 1\ 0\ 0\ 0\ 0] \mapsto [0\ 0\ 0\ 0\ 0\ 0\ 0\ 1\ 0\ 0], \\ & [0\ 0\ 0\ 0\ 0\ 0\ 0\ 1\ 0\ 0] \mapsto [0\ 0\ 0\ 0\ 0\ 0\ 0\ 0\ 1\ 0], \\ & [0\ 0\ 0\ 0\ 0\ 0\ 1\ 0\ 0\ 0] \mapsto [0\ 0\ 0\ 0\ 0\ 0\ 0\ 0\ 0\ 1], \\ & \} \end{aligned} \tag{38}$$

Let f be the empedding of the subspace $\mathsf{ker}(\boldsymbol{O}_1 - \boldsymbol{O}_2)$ into $\mathbb{F}^d$. This linear map can be described by a matrix $\boldsymbol{M}_f$ of dimension 10×9 if we just choose a basis of $\mathsf{ker}(\boldsymbol{O}_1 - \boldsymbol{O}_2)$.

$$\boldsymbol{M}_f = \begin{bmatrix} 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 1 & 1 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 \end{bmatrix} \tag{39}$$

This matrix has been constructed by first choosing a basis of $\mathsf{ker}(O)$ and mapping it to $\mathbb{F}^{2d}$ by setting the first resp. second halve to 0. These make up 6 basis vectors. The last basis vector is chosen to be linear independent of the rest while still being in $\mathsf{ker}(\boldsymbol{P}_1 - \boldsymbol{P}_2) \subseteq \mathsf{ker}(\boldsymbol{O}_1 - \boldsymbol{O}_2)$.

This linear map leads to the partially collapsed constraints:

$$\begin{aligned} \mathcal{C}_{\text{join}}\boldsymbol{M}_f = \{&[1\ 0\ 0\ 0\ 0\ 0\ 0\ 0\ 0\ 0] \mapsto [0\ 0\ 1\ 0\ 0\ 0\ 0\ 0\ 0\ 0]\,, \\ &[0\ 0\ 1\ 0\ 0\ 0\ 0\ 0\ 0\ 0] \mapsto [0\ 0\ 0\ 1\ 0\ 0\ 0\ 0\ 0\ 1]\,, \\ &[0\ 1\ 0\ 0\ 0\ 0\ 0\ 0\ 0\ 0] \mapsto [0\ 0\ 0\ 1\ 0\ 0\ 0\ 0\ 0\ 0]\,, \\ &[0\ 0\ 0\ 0\ 1\ 0\ 0\ 0\ 0\ 0] \mapsto [0\ 0\ 0\ 0\ 0\ 0\ 0\ 1\ 0\ 0]\,, \\ &[0\ 0\ 0\ 0\ 0\ 0\ 1\ 0\ 0\ 0] \mapsto [0\ 0\ 0\ 0\ 0\ 0\ 0\ 0\ 1\ 1]\,, \\ &[0\ 0\ 0\ 0\ 0\ 1\ 0\ 0\ 0\ 0] \mapsto [0\ 0\ 0\ 0\ 0\ 0\ 0\ 0\ 1\ 0]\} \end{aligned} \tag{40}$$

These constraints are not solvable anymore, as the set of answer vectors alone is not linearly independent.

But they can be solved by further collapsing variables. We can collapse the answer vectors of lines 2 and 3, i.e. we collapse $f^*(\boldsymbol{a}_2^1)$ with $f^*(\boldsymbol{a}_3^1)$. (Here I write $\boldsymbol{a}_j^i$ for the j'th answer vector $\boldsymbol{a}_j$ of the original $\mathcal{P}$ either in the first or second half of $\mathbb{F}^d$ depending on $i \in \{1, 2\}$.) So we construct a map going to the subspace where these vectors collapse. This is $\mathsf{ker}([0\ 0\ 0\ 1\ 0\ 0\ 0\ 0\ 1] - [0\ 0\ 0\ 1\ 0\ 0\ 0\ 0\ 0]) = \mathsf{ker}([0\ 0\ 0\ 0\ 0\ 0\ 0\ 0\ 1])$.

Remark 6.2.3.1: Its interesting that $f^*(\boldsymbol{o}_1) = f^*(\boldsymbol{o}_2) = [0\ 0\ 0\ 0\ 0\ 0\ 0\ 0\ 1]$. So collapsing these two constraints is the same as setting the output to 0. In this case this makes sense, because 0 is the only output for which it is hard to find a collision.

Looking back at the original linicrypt program, this subspace are the states of the program for which the query $H(y)$ and $H(H(x))$ collapse.

Let $g : \mathsf{ker}(\boldsymbol{a}_2^1 - \boldsymbol{a}_3^1) \cap \mathsf{ker}(\boldsymbol{O}_1 - \boldsymbol{O}_2) \hookrightarrow \mathbb{F}^d$ be the embedding. Then a matrix representing this map is

$$\boldsymbol{M}_f = \begin{bmatrix} 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \end{bmatrix} \tag{41}$$

Only the last column is removed versus $\boldsymbol{M}_f$ because it lies outside of $\mathsf{ker}(\boldsymbol{a}_2^1 - \boldsymbol{a}_3^1)$. The collapsed set of constraints becomes:

$$\begin{aligned} \mathcal{C}_{\text{join}}\boldsymbol{M}_g = \{&[1\ 0\ 0\ 0\ 0\ 0\ 0\ 0] \mapsto [0\ 0\ 1\ 0\ 0\ 0\ 0\ 0], \\ &[0\ 0\ 1\ 0\ 0\ 0\ 0\ 0] \mapsto [0\ 0\ 0\ 1\ 0\ 0\ 0\ 0], \\ &[0\ 1\ 0\ 0\ 0\ 0\ 0\ 0] \mapsto [0\ 0\ 0\ 1\ 0\ 0\ 0\ 0], \\ &[0\ 0\ 0\ 0\ 1\ 0\ 0\ 0] \mapsto [0\ 0\ 0\ 0\ 0\ 0\ 1\ 0], \\ &[0\ 0\ 0\ 0\ 0\ 0\ 1\ 0] \mapsto [0\ 0\ 0\ 0\ 0\ 0\ 0\ 1], \\ &[0\ 0\ 0\ 0\ 0\ 1\ 0\ 0] \mapsto [0\ 0\ 0\ 0\ 0\ 0\ 0\ 1]\} \end{aligned} \tag{42}$$

The running example from [1] is

6.3. **Notes and ideas, in random order.**

- Second preimage resistance and collision resistance loose their relationsship for unsolvable constraints. We can find unsolvable constraints where its easy to find a second solution, if we are given a solution. But its hard to find a solution in the first place. This is due to the permutation attack; where the symmetry of the constraints leads to a symmetry in the set of solutions. This is contrary to the solvable case, where finding second preimages is harder than finding collisions. Note: For the unique nonces case [1] found that both notions are equivalent. It's interesting to see that in the more general no nonces case they split, but one is stronger that the other, and in the most general case they are unrelated.
- The set $\mathsf{sol}(\mathcal{C})$ has interesting structure. For matrices permuting constraints $\mathcal{C}\boldsymbol{B} = \mathcal{C}$ we get a well defined map $\boldsymbol{B} : \mathsf{sol}(\mathcal{C}) \to \mathsf{sol}(\mathcal{C})$.

  For general matrices $\boldsymbol{L} : \mathbb{F}^{d'} \to \mathbb{F}^d$ for arbitray d' we get a map $\boldsymbol{L} : \mathsf{sol}(\mathcal{C}\boldsymbol{L}) \to \mathsf{sol}(\mathcal{C})$.

  It looks like the set of solutions contains "components", where each component is the solution set of the "derived" set of constraints mapped linearly into $\mathbb{F}^d$. With derived constraints I mean the constraints that we can build by mapping them linearly. I think the only interesting cases are when we map them such that some constraints are collapsed and we are left with a solvable set. Then I would expect finding solutions to be easier (less constraints), and we can map them back.
- If the function H allows for fixed points, that opens another can of (intersting) worms. Because then unsolvable sets become solvable in a limited way. $\mathcal{C} = \{[1] \mapsto [1]\}$ is then solvable (fixing $\{0\}$) because $\mathsf{sol}(\mathcal{C})$ are exactly the fixed points $\{x \mid H(x) = x\}$. This set of solutions can be used to find solutions to every? unsolvable set.

E.g. $\mathcal{C} = \{[1\ 0] \mapsto [0\ 1], [0\ 1] \mapsto [1\ 0]\}$ can be collapsed via $\boldsymbol{L} = \begin{bmatrix}1\\1\end{bmatrix}$ to $\{[1] \mapsto [1]\}$. Therefore solutions to

- Maybe it is cleaner to drop the idea of input matrix $\boldsymbol{I}$ and of the fixing $\mathcal{F}_0$ space. We could model Linicrypt input as just another set of constraints. We already have random oracle constraints and ideal cipher constraints. Now we introduce input constraints $c = \boldsymbol{i}$ for $\boldsymbol{i}$ in $\left(\mathbb{F}^d\right)^*$. They restrict the valid states of $\mathcal{P}$ in the dimension $\boldsymbol{i}$. We instantiate them with a concrete input $i \in \mathbb{F}$: $\boldsymbol{iv} = i$. This is structurally similar to the random oracle constraint, which is instantiated with a concrete orcle H: $H(\boldsymbol{Qv}) = \boldsymbol{av}$.

  Benefits:
  - The collapse of the input matrix is now handled by collapse of constraints.
  - We allow extra freedom in the solution ordering which seems more general. Previously we kind of require the inputs to $\mathcal{P}$ to be fixed first. With lots of programs, it would be ok to start computing constraints without knowing all the input. This is the case for merkle damgard for example.

7. Next steps

The SymPy Linicrypt implementation has shown a lot of potential. I want to continue this path a bit more because it will help a lot with generating interesting applications of the theory. The todo-list might go like this:

- Implement the collapse attack. Currently, I tested only the SymPy equations for cycling the constraints.
- Generate a Linicrypt-driven categorization of the attacks

This is the Todo list for everything related to the security side:

- Write down the proof attempt for the security side of Conjeture 2.2
- Continue work on the security side: What can we say about unsolvable $\mathcal{C}$? Can this help to generalize the security proofs?
- Maybe (and this is a big maybe) we can then base the security proofs on weaker conditions than the Random Oracle Model. What are the exact requirements on H such that unsolvable constraints $\mathcal{C}$ are "hard" to solve?
- Start to work on the "unsolvable $\mathcal{C}$ = loops in $\mathcal{C}$" idea.

References

1. McQuoid, I., Swope, T., Rosulek, M.: Characterizing Collision and Second-Preimage Resistance in Linicrypt. In: Hofheinz, D. and Rosen, A. (eds.) TCC~2019: 17th Theory of Cryptography Conference, Part~I. pp. 451–470. Springer, Heidelberg, Germany, Nuremberg, Germany (2019)
2. Black, J., Rogaway, P., Shrimpton, T.: Black-Box Analysis of the Block-Cipher-Based Hash-Function Constructions from PGV. In: Yung, M. (ed.) Advances in Cryptology – CRYPTO~2002. pp. 320–335. Springer, Heidelberg, Germany, Santa Barbara, CA, USA (2002)
3. Hollenberg, T., Rosulek, M., Roy, L.: A Complete Characterization of Security for Linicrypt Block Cipher Modes. In: 2022 IEEE 35th Computer Security Foundations Symposium (CSF). pp. 439–454. IEEE, Haifa, Israel (2022)
4. Semmel, F.: Linicrypt SymPy Implementation, https://github.com/frederiksemmel/linicrypt
5. Preneel, B., Govaerts, R., Vandewalle, J.: Hash Functions Based on Block Ciphers: A Synthetic Approach. In: Stinson, D. R. (ed.) Advances in Cryptology – CRYPTO'93. pp. 368–378. Springer, Heidelberg, Germany, Santa Barbara, CA, USA (1994)

Zurich
Email address: semmelf@student.ethz.ch