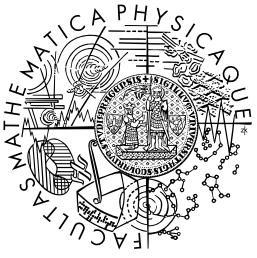


FACULTY
OF MATHEMATICS
AND PHYSICS
Charles University

MASTER THESIS

Petr Sedláček

Security of Trapdoor Permutations under Preimage Leakage

Computer Science Institute of Charles University

Supervisor of the master thesis: Mgr. Pavel Hubáček, Ph.D.

Study programme: Mathematics

Study branch: Mathematics for Information Technologies

Prague 2023

I declare that I carried out this master thesis independently, and only with the cited sources, literature and other professional sources. It has not been used to obtain another or the same degree.

I understand that my work relates to the rights and obligations under the Act No. 121/2000 Sb., the Copyright Act, as amended, in particular the fact that the Charles University has the right to conclude a license agreement on the use of this work as a school work pursuant to Section 60 subsection 1 of the Copyright Act.

In ..... date ....

Petr Sedláček

I dedicate this work to my bigger half, again.

Title: Security of Trapdoor Permutations under Preimage Leakage

Author: Petr Sedláček

Department: Computer Science Institute of Charles University

Supervisor: Mgr. Pavel Hubáček, Ph.D., Computer Science Institute of Charles University

Abstract: This thesis explores preimage leakage-resilient trapdoor permutations (PLR-TDPs) and their applications in proofs of storage replication and incompressible encodings. The thesis consists of three chapters covering the trapdoor permutations, formal definition of PLR-TDPs, and analysis of security properties of PLR-TDPs. The first chapter provides an overview of trapdoor permutations (TDPs), their definitions, and applications in proofs of storage replication. Our results are presented in the second and third chapters. The second chapter formally defines PLR-TDPs and demonstrates their use by constructing a simple incompressible encoding in the random oracle model. The third chapter focuses on the existence of PLR-TDPs. It demonstrates the strong preimage leakage-resilience of fully random TDPs in an idealized model. We are the first to provide a partial formal justification for the conjecture of the preimage leakage-resilience of practical TDPs, such as RSA or Rabin permutations.

Keywords: trapdoor permutations, preimage leakage, proof of storage replication, incompressible encodings

Contents

Introduction

Trapdoor permutations are central for public-key cryptography. In this thesis, we focus on their less known property, which is security under preimage leakage. Preimage leakage-resilience is defined by the extension of a standard one-way security experiment. An adversary tries to invert a "hard to invert" function with the knowledge of the image and a partial knowledge of the preimage. For example, let $f(x) = y$, then the adversary tries to invert f with the knowledge of y and a partial knowledge of x in a form of some short digest w capturing the *preimage leakage*. Note that standard *leakage-resilience* from cryptographic literature refers to the leakage of the secret key (see, e.g., [KR19]), whereas our leakage-resilience refers to the leakage of the preimage of a one-way (trapdoor) permutation.

In the first chapter, we present an overview of trapdoor permutations. We state their formal definition as well as definitions of related concepts. Preliminaries are followed by some known applications of trapdoor permutations. In particular, we focus on applications in *proofs of storage replication* [DGO19], which are tightly connected to the preimage leakage-resilience. The last part of the first chapter focuses on *hourglass schemes* [vDJO+12] centred around proofs of correct encryption. These schemes play a principal role in the recent developments in the field, including the aforementioned proofs of storage replication.

The second chapter focuses on the preimage leakage-resilience property. We give a formal definition of *preimage leakage-resilient trapdoor permutations* (PLR-TDPs), which, as their name suggests, are resilient to a potentially significant preimage leakage. We show that the property of preimage leakage-resilience is sufficient for the construction of schemes from the applications mentioned in the first chapter. We show this by constructing an incompressible encoding scheme as defined in [MW20] using PLR-TDPs in the random oracle (RO) model. The use of PLR-TDPs has a significant benefit over the known constructions – it leads to simple-to-analyze schemes. However, this alone is insufficient for the PLR-TDPs to be used as an assumption in proofs. It is not clear whether these primitives exist "in the wild". This issue is tackled in the last chapter.

The third and last chapter positively answers the question of whether the use of PLR-TDPs stands on solid foundations. We show that in an idealised model, a random trapdoor permutation is *preimage leakage-resilient*. A second interesting result in this chapter is that for every $\delta \in (0, 1)$, there exists a set of parameters such that a fully random trapdoor permutation of sufficient length n is secure even if $\delta \cdot n$ bits of preimage are leaked.

Our contributions

1. We give a formal definition of PLR-TDPs.
2. We show that PLR-TDPs give simple incompressible encodings in the random oracle model.
3. We show that an ideal random TDP is strongly preimage leakage-resilient.

Specifically, our result from the last item is the first step towards validating the conjecture of van Dijk et al. in [vDJO+12] about the preimage leakage-resilience of the RSA permutation (see Section 1.2.2 for further discussion).

1. Trapdoor permutations and their applications

In this chapter, we present a primitive called *trapdoor permutations* (TDPs). We start with the necessary terminology and proceed with various applications. We pay close attention to their application in *proof of storage replication* [DGO19, GLW20] and deeply connected *incompressible encodings* [MW20]. We also describe *hourglass schemes* [vDJO+12], which highly influenced all of the other applications.

1.1 Preliminaries

Notation 1.1. By Π_n, we denote the set of all permutations on $\{0,1\}^n$.

For clarity, we restate definitions of one-way permutations and trapdoor permutations based on [Gol01].

Definition 1.2 (Trapdoor permutations)**.** Let $\lambda \in \mathbb{N}$ be a security parameter. Let $\overline{I}$ be an infinite set of indices and let $G = (G^1, G^2)$ be an index-sampling algorithm, where $G^1(1^\lambda)$ denotes the first half of the output of $G(1^\lambda)$. A collection of permutations $\mathcal{F} = \{f_k : D_k \mapsto D_k\}_{k \in \overline{I}}$ is a *family of trapdoor permutations* if there exists a tuple of probabilistic polynomial-time (PPT) algorithms (G, D, F, F^{-1}) satisfying the following conditions:

Efficient evaluation: The output distribution of the index-sampling algorithm G^1 on input 1^λ is a random variable assigned values in the set $\overline{I} \cap \{0,1\}^\lambda$. The output distribution of the domain-sampling algorithm D on input $k \in \overline{I}$ is a random variable assigned values in D_k. On input $k \in \overline{I}$ and $x \in D_k$, algorithm F always outputs $f_k(x)$.

One-wayness: For every PPT algorithm $\mathcal{A}$, every polynomial p and all sufficiently large λ, it holds that

$$\Pr[\mathcal{A}(f_{G^1_\lambda}(X_\lambda), G^1_\lambda) = X_\lambda] \leq \frac{1}{p(\lambda)},$$

where G^1_λ is a random variable describing the output distribution of the algorithm G^1 on input 1^λ and X_λ is a random variable describing the output of algorithm D on input G^1_λ.

Efficient invertibility with a trapdoor: For every (k, td) in the range of G and every $x \in D_k$, it holds that $F^{-1}(td, F(k, x)) = x$.

For additional details about Definition 1.2, see [Gol01, p. 52–54, 57].

1.2 Applications of TDPs

Public-key encryption (PKE) schemes, such as RSA [RSA78] and Rabin [Rab79], had a huge impact on the cryptographic research. An important generalization was made by Andrew C. Yao [Yao82] who created the notion of trapdoor permutations as a sufficient condition for the construction of PKE schemes. Using TDPs allowed researches to avoid dependence on a concrete construction – if a new construction of PKE schemes emerges, their results are applicable without significant changes.

Trapdoor permutations are a versatile primitive for many applications in theoretical cryptography. With trapdoor permutations, it is possible to create basic cryptographic constructions, including pseudorandom generators (PRGs) and pseudorandom functions [KL20]. However, main purpose of TDPs is to use them as a building block for complex constructions. For example, TDPs can be utilised for aforementioned public-key cryptography and signatures schemes [RSA78, Rab79, Pai99, LMRS04], private information retrieval [CKGS98, KO00], and encoding schemes utilising incompressibility properties [DGO19, GLW20, MW20]. In the following, we focus on the last mentioned application, namely *replica encodings* and *incompressible encodings*.

1.2.1 Applications in proof of storage replication

An article [DGO19] (Proofs of replicated storage without timing assumptions) by Ivan Damgård, Chaya Ganesh and Claudio Orlandi was presented at the CRYPTO 2019 conference. The article introduced a novel replica encoding scheme which utilises a notion of incompressibility to prove that multiple encodings of data are stored on a remote server. Since then, subsequent research has been carried out by various groups. Tal Moran and Daniel Wichs published an article *Incompressible Encodings* [MW20]. They extracted the incompressibility property of replica encodings into a separate scheme. They showed that the replica encodings can be constructed using their incompressible encodings. In the same year, a group of scientists discovered that the original article [DGO19] had serious flaws in its security argument. In their article [GLW20], Rachit Garg, George Lu and Brent Waters established an adjusted replica encoding scheme with proof of security.

In this section, we briefly present a variant of the replica encoding by [GLW20] and highlight techniques that are used in the proof of security of the scheme. From now on, by *replica encoding scheme*, we refer exclusively to the variant by [GLW20].

The replica encoding utilises a random oracle H, trapdoor permutation F and ideal cipher T. The encoding process has three main phases that are denoted in Figure 1.1.

Decoding is similar, with the steps performed inversely to the encoding procedure. The main difference is that the decoding applies TDP F in the "easy" direction. Thus the trapdoor is not required for successful decoding.

The main gist of the security game is the following. An adversary $\mathcal{A}_1$ has access to n different replicas (encodings) of a message m, produces a small digest `state` and sends it to a second adversary $\mathcal{A}_2$. The second adversary has access to the original message m, the digest `state` created by the adversary $\mathcal{A}_1$, and it also knows the total number of replicas. Their goal is to recreate as many replicas as

Figure 1.1: Encoding process of replica encodings

First phase

- Message is divided into b blocks of the same length.

Second phase

- Every message block is XORed with an output from the random oracle H queried on a pair (ρ,i), where ρ is a randomly generated string and i is an index of the particular message block.

Third phase

- For each block, the two following steps are repeated r times:
  - The current value of the block is permuted using the ideal cipher T.
  - The "hard" direction of the trapdoor permutation F is applied, which requires the trapdoor.
- After r rounds, the encoding outputs ρ and the final values for each block.

possible using only the information provided to them.

We highlight the fact that in the third phase of the replica encoding scheme, the trapdoor permutation and the ideal cipher are queried multiple (r) times. There is an important relation between the number of rounds r and the size of the digest `state` produced by the adversary A_1 for the adversary A_2, which is discussed in Lemma 1.5.

The following lemma states conditions for the success of adversaries. Importantly, a scheme is parameterized by $s \in [0,1]$, which defines the minimal amount of compression required from the adversary A_1. The greater the s, the less compression is required from the adversary A_1.

Lemma 1.3 ([GLW20])**.** *Let $v \in \mathbb{N}$ denote the number of the correct guesses of the adversary $\mathcal{A}_2$. Let $s \in [0,1]$ be a parameter, let ℓ be the length of an encoded message and let $|\texttt{state}|$ denote the size of the digest exchanged between $\mathcal{A}_1$ and $\mathcal{A}_2$. Then the adversaries succeed if*

$$|\texttt{state}| < v \cdot s \cdot \ell$$

For a scheme to be considered secure (for a fixed parameter s), we require the probability of success of adversaries in the security game to be negligible in the security parameter λ.

One of the contributions of [GLW20] is a proof of the following theorem.

Theorem 1.4 ([GLW20])**.** *Let $\kappa \in \mathbb{N}$ be the length of the encoding of each message block and let $\lambda \in \mathbb{N}$ be the security parameter. Then the replica encoding scheme described above is secure for all*

$$s \in 1 - \frac{\omega(\log \lambda)}{\kappa}.$$

Their proof is established via a sequence of hybrids that follow two main ideas. The first is to show that the adversary $\mathcal{A}_2$ must query sequentially on at least one replica. The second focuses on exchanging the original permutation for a different one. The combination of those two approaches allows the embedding of a TDP challenge. If the adversary succeeds in breaking the scheme with non-negligible probability, then it breaks the TDP security with non-negligible probability.

For the sequence of hybrids to be correct, the probabilities of adversaries winning two consequent hybrids must be polynomially dependent. The proof contains a sequence of nine games (hybrids) in total, and it is not easy to follow.

We highlight a transition between games 5 and 6, which embeds a requirement for the number of rounds r with respect to the size of the exchanged digest `state`. The relation is described in the following lemma.

Lemma 1.5 ([GLW20, p. 25])**.** *Let ϵ be the probability of the adversary winning Game 5. Then the probability of the adversary winning Game 6 is greater than or equal to*

$$\left(\frac{\epsilon}{2}\right)\left(\frac{\epsilon 2^{r-1} - 2^{|\texttt{state}|} + 1}{\epsilon 2^{r-1}}\right).$$

For the sequence to be polynomially dependent, it must hold that r is significantly greater than $|\texttt{state}|$. Therefore, the third phase of encoding requires a considerable number of iterations, which renders the scheme unsuitable for practical use.

1.2.2 Hourglass schemes and replica encodings

Article [vDJO+12] by Marten van Dijk, Ari Juels, Alina Oprea, Ronald L. Rivest, Emil Stefanov and Nikos Triandopoulos presented protocols called *hourglass schemes* that prove the correct encryption of files at rest[1] on a server. All of the protocols focused on replica encodings (and incompressibility encodings) are greatly influenced by hourglass schemes protocols. In the following, we present the *hourglass schemes* and compare them to *replica encodings*.

The *hourglass schemes* solve the following problem. A server is supposed to store files (at rest) encrypted so that the negative impact of accidental data leakage is minimized. However, there is no obvious way for the client to verify that the files on the server are encrypted correctly. Suppose that a plaintext file (data) d should be stored in an encrypted state e. The encryption is performed on d divided into b blocks d_i, i.e., $d = \{d_i\}_{i=1}^{b} \xrightarrow{\text{Enc}} e = \{e_i\}_{i=1}^{b}$. The client can challenge the server to return an i-th encrypted block e_i. However, encryption is (generally) not an expensive operation. Even if the server stores only the plaintext d, it is easy for them to return the expected e_i by encrypting the block d_i *on the fly*. The hourglass schemes add an additional layer of complexity, where the encrypted e is transformed into g using an *hourglass function*.

Hourglass function has the following properties. It provides a resource bound on how fast the function can be computed in one direction, while it is easy to be computed in the other direction. An example is an inversion of a hash function H with a relatively short (fixed) output length. It is moderately hard to find

[1] *Files at rest* are files that are not used by any program at the given moment.

an inverse H^{-1} of a block e_i, $H^{-1}(e_i) = g_i$. On the other hand, it is easy to compute $H(g_i) = e_i$. This type of hourglass function provides a time-bound – the time it takes for an honest server to respond to a challenge is on average significantly shorter than the time it takes the dishonest server to find an inverse $H^{-1}(e_i)$. A different type of hourglass schemes is of our interest – schemes based on trapdoor permutations.

Hourglass schemes based on trapdoor permutations do not put bounds on time but rather on computational complexity. Let F be a trapdoor permutation.

- A file (data) d is encrypted by blocks into e.
- Each block e_i is encoded into g_i by the client using F^{-1}, which requires a trapdoor that is known to the client only.
- After the encoding, the trapdoor is destroyed. The $g = \{g_i\}$ and the description of F are sent to the server.

The use of trapdoor permutation guarantees that for a dishonest server, it is not feasible to obtain g_i from e_i. However, for the security of the scheme to be sound, the following assumption is used: F^{-1} is *near-incompressible*, i.e., g cannot be efficiently compressed below $|g| - O(\log |g|)$ bits.

Van Dijk et al. [vDJO+12] conjectured that this property holds for RSA permutation. They also conjecture that the assumption of *near-incompressibility of* F^{-1} can be completely avoided by applying F^{-1} multiple times together with repeated application of a (pseudo)random permutation. The last conjecture was proven correct by Rachit Garg, George Lu, and Brent Waters in [GLW20]. We discuss the relation between the number of rounds and the size of the digest in Section 1.2.1. For more details about the *hourglass schemes* and related definitions, see [vDJO+12].

Replica encodings by [DGO19, GLW20] are an adaptation of the presented hourglass scheme. Replica encodings do not require any encryption per se. Nevertheless, it is required for security purposes that the distribution of bit strings on which the trapdoor permutation is applied are selected uniformly at random. For that reason, the plaintext is XORed with an output from a RO instead of common encryption. No additional incompressibility property is assumed. Therefore, their security proof requires a considerable number of rounds, i.e., repeated use of a trapdoor permutation and an ideal cipher.

Figure 2.1: Inversion experiment PLR-TDP.InvExp$^{\eta}_{\mathcal{A},\mathcal{F}}(1^{\lambda})$

$k, td \leftarrow G(1^{\lambda})$
$x \leftarrow D(1^{\lambda})$
$y = F_k(x)$
$w \leftarrow \mathcal{A}.\text{Leak}(k, x, y)$
$x' \leftarrow \mathcal{A}.\text{Invert}(k, w, y)$
if $x = x'$ and $w \leq \eta(1^{\lambda})$ output 1, otherwise output 0

2. Preimage leakage-resilient trapdoor permutations

The example in Section 1.2.1 shows that proving the security of an incompressible scheme in the RO model is not easy. Proofs are often very involved and overly complicated. That motivates a definition of preimage leakage-resilient trapdoor permutations (PLR-TDPs). This notion allows proofs to be more straightforward by shifting the complexity to the construction of PLR-TDPs.

2.1 Definitions of PLR-TDPs

We define the following PLR-TDP inversion experiment.

Definition 2.1 (PLR-TDP Inversion Experiment)**.** For a family of trapdoor permutations $\mathcal{F}$ with the corresponding algorithms (G, D, F, F^{-1}), length parameter η, and an adversary $\mathcal{A} = (\mathcal{A}.\text{Leak}, \mathcal{A}.\text{Invert})$, we define the PLR-TDP inversion experiment in Figure 2.1.

Definition 2.2 (PLR-TDP)**.** A family of trapdoor permutations $\mathcal{F}$ is a family of *η-PLR-TDPs* if for all PPT adversaries $\mathcal{A}$ it holds that

$$\Pr[\text{PLR-TDP.InvExp}^{\eta}_{\mathcal{A},\mathcal{F}}(1^{\lambda}) = 1] \in \text{negl}(\lambda).$$

For this definition, there exists a trivial compression of any n-bit x to $n - \log(n)$ bits, i.e., $\eta(1^{\lambda}) = n - \log(n)$. Let $x = x_1x_2x_3x_4 \dots x_n$. The adversary $\mathcal{A}.\text{Leak}$ outputs $w = x_{\log(n)+1}x_{\log(n)+2} \dots x_n$, i.e., the digest w is x without the first $\log(n)$ bits. Then, $\mathcal{A}.\text{Invert}$ gets w as an input and guesses the omitted bits. The probability of the $\mathcal{A}.\text{Invert}$ correctly guessing the missing bits is $1/n$, which is non-negligible in n. Note that this trivial attack motivates the *near-incompressibility* property of van Dijk et al. dicussed in Section 1.2.2.

Figure 2.2: Compression experiment $\mathrm{IE.CompExp}^{\Phi}_{\mathcal{A}^{\mathrm{IE}},\beta}$

$m,\ \mathtt{aux}$	$\leftarrow \mathcal{A}^{\mathrm{IE}}.\mathrm{Select}(1^\lambda)$
c	$\leftarrow \Phi.\mathrm{Enc}(1^\lambda, m)$
$\widetilde{w}$	$\leftarrow \mathcal{A}^{\mathrm{IE}}.\mathrm{Compress}(c, \mathtt{aux})$
c'	$\leftarrow \mathcal{A}^{\mathrm{IE}}.\mathrm{Expand}(\widetilde{w}, \mathtt{aux})$
if $c = c'$ and $\lvert\widetilde{w}\rvert \leq \beta(1^\lambda)$ output 1, otherwise output 0	

2.2 Incompressible encodings from PLR-TDPs

In this section, we work with *incompressible encodings*. First, we recall the notion of incompressible encodings from [MW20]. Second, we construct an incompressible encoding scheme using PLR-TDPs in the RO model.

Definition 2.3 ([MW20, p. 11–12])**.** An (α, β)-*incompressible encoding scheme* Φ consists of PPT algorithms $\Phi = (\Phi.\mathrm{Enc}, \Phi.\mathrm{Dec})$. We require the following properties.

Correctness: For all $\lambda \in \mathbb{N}$ and all $m \in \{0,1\}^*$ we have

$$\Pr[\Phi.\,\mathrm{Dec}(\Phi.\,\mathrm{Enc}(1^\lambda, m) = m] - 1 \in \mathrm{negl}(\lambda).$$

α-Expansion: For all $\lambda, k \in \mathbb{N}$ and all $m \in \{0,1\}^k$ we have

$$\Pr[|\Phi.\,\mathrm{Enc}(1^\lambda, m)| \leq \alpha(\lambda, k)] = 1.$$

β-Incompressibility: Compression experiment $\mathrm{IE.CompExp}^{\Phi}_{\mathcal{A}^{\mathrm{IE}},\beta}(1^\lambda)$ is defined in Figure 2.2. For all PPT adversaries $\mathcal{A}^{\mathrm{IE}}$ we have

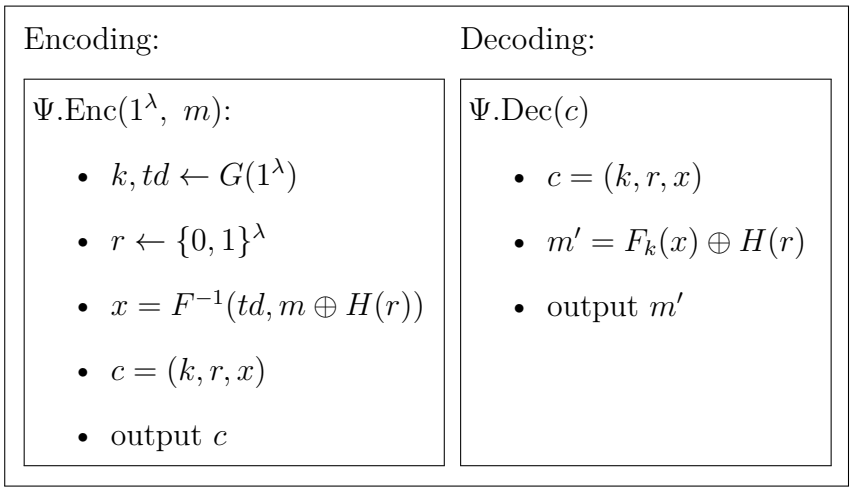
$$\Pr[\mathrm{IE.CompExp}^{\Phi}_{\mathcal{A}^{\mathrm{IE}},\beta}(1^\lambda) = 1] \in \mathrm{negl}(\lambda).$$

Given a family of PLR-TDPs $\mathcal{F}$ and a random oracle H, we construct an incompressible encoding scheme $\Psi = (\Psi.\mathrm{Enc}, \Psi.\mathrm{Dec})$ as defined in Figure 2.3.

Theorem 2.4 (PLR-TDP + RO $\Longrightarrow$ IE)**.** *For a family of η-PLR-TDPs, the encoding scheme $\Psi = (\Psi.\mathrm{Enc}, \Psi.\mathrm{Dec})$ described in Figure 2.3 is a β-incompressible encoding scheme, where $\beta(1^\lambda) = \eta(1^\lambda) - \lambda$.*

Proof. Suppose there exists an adversary $\mathcal{A}^{\mathrm{IE}}$ such that it breaks the encoding scheme Ψ (compression experiment $\mathrm{IE.CompExp}^{\Psi}_{\mathcal{A}^{\mathrm{IE}},\beta}$ outputs 1 with non-negligible probability). We want to prove that such adversary implies the existence of an adversary $\mathcal{A}^{\text{PLR-TDP}}$ capable of breaking the PLR-TDP inversion experiment $\mathrm{PLR\text{-}TDP.InvExp}^{\eta}_{\mathcal{A}^{\text{PLR-TDP}},\mathcal{F}}$ from Figure 2.1. We construct the adversary $\mathcal{A}^{\text{PLR-TDP}} = (\mathcal{A}^{\text{PLR-TDP}}.\mathrm{Leak}, \mathcal{A}^{\text{PLR-TDP}}.\mathrm{Invert})$ as follows:

Figure 2.3: Incompressible encoding scheme Ψ



$\mathcal{A}^{\text{PLR-TDP}}$.Leak$(k, x, y)$:

1. Samples $s \leftarrow \{0,1\}^\lambda$.
2. Generates r_1, r_2 using PRG with the randomness s, i.e., $(r_1, r_2) = \text{PRG}(s)$.
3. Calls $\mathcal{A}^{\text{IE}}$.Select on input (r_1) instead of running it on a truly random string. Stores its output m and auxiliary information `aux`.
4. Sets random oracle to output $m \oplus y$ on input r_2, i.e., $H(r_2) = m \oplus y$.
5. Calls $\mathcal{A}^{\text{IE}}$.Compress on input $(c = (k, r_2, x),$ `aux`$)$.
6. The $\mathcal{A}^{\text{IE}}$.Compress outputs a (short) digest $\widetilde{w}$.
7. Outputs the digest with the seed for PRG: $w = \widetilde{w}||s$.

$\mathcal{A}^{\text{PLR-TDP}}$.Invert$(k, w, y)$:

1. Interprets w as $\widetilde{w}||s$.
2. $(r_1, r_2) = \text{PRG}(s)$.
3. Calls $\mathcal{A}^{\text{IE}}$.Select on input r_1. Stores its output m and `aux`.
4. Calls $\mathcal{A}^{\text{IE}}$.Expand on input $(\widetilde{w},$ `aux`$)$.
5. The $\mathcal{A}^{\text{IE}}$.Expand outputs a codeword c'.
6. Interprets c' as (k', s', x').
7. Outputs x'.

We need to verify that the distributions of variables (inputs) for the $\mathcal{A}^{\text{IE}}$ algorithms are indistinguishable between experiments IE.CompExp$^{\Phi}_{\mathcal{A}^{\text{IE}},\beta}$ and PLR-TDP.InvExp$^{\eta}_{\mathcal{A}^{\text{PLR-TDP}},\mathcal{F}}$. First, we focus on the $\mathcal{A}^{\text{PLR-TDP}}$.Leak algorithm.

- We fix the randomness for the $\mathcal{A}^{\text{IE}}$.Select algorithm as r_1. As the algorithm does not take any input except for the security parameter 1^λ, it cannot distinguish between the experiments it takes part in.
- The $\mathcal{A}^{\text{IE}}$.Compress algorithm expects on input `aux` generated by the $\mathcal{A}^{\text{IE}}$.Select and a codeword $c = (k, r, x)$ generated by the Ψ.Enc, where key k is generated by the G algorithm, r is uniformly random and $x = F^{-1}(td, m \oplus H(r))$.
  - The key k is generated by the same algorithm G.
  - The randomness r is replaced by r_1, which was created by a PRG. If the adversary could distinguish between r (which is uniformly random) and r_1, it would contradict the pseudorandomness of the PRG.
  - The x the adversary receives on input is equal to $F^{-1}(td, y)$. Thus, we want to show that the adversary cannot distinguish between y and $y' := m \oplus H(r)$. First, we can replace r with r_1 as they are indistinguishable. Second, for any fixed m, the distribution of y' over $r \leftarrow \{0,1\}^\lambda$ is uniformly random. With the knowledge of both m and r and access to the random oracle H, the $y = m \oplus H(r)$ must hold for the y to be indistinguishable from y'. It holds from point 5 of the $\mathcal{A}^{\text{PLR-TDP}}$.Leak$(k, x, y)$ algorithm, where $r = r_1$.
  - The `aux` value is generated by the same algorithm $\mathcal{A}^{\text{IE}}$.Select.

Second, we have a look at the second algorithm $\mathcal{A}^{\text{PLR-TDP}}$.Invert.

- For the $\mathcal{A}^{\text{IE}}$.Select algorithm, the situation is the same as for the $\mathcal{A}^{\text{PLR-TDP}}$.Leak algorithm. Because of the fixed randomness r_1, the generated message m and `aux` are identical to those generated previously.
- The $\mathcal{A}^{\text{IE}}$.Expand algorithm expects on input the `aux` generated by $\mathcal{A}^{\text{IE}}$.Select and a digest $\widetilde{w}$ generated by the $\mathcal{A}^{\text{IE}}$.Compress.
  - From the case of $\mathcal{A}^{\text{PLR-TDP}}$.Leak we know that the $\mathcal{A}^{\text{IE}}$.Compress cannot distinguish between the experiments. Thus neither can $\mathcal{A}^{\text{IE}}$.Expand, as it does not have any additional information that is not available to the $\mathcal{A}^{\text{IE}}$.Compress. Namely, the digest $\widetilde{w}$ and the `aux` have the same distributions in both experiments.

For the IE.CompExp$^{\Phi}_{\mathcal{A}^{\text{IE}},\beta}$ to output 1, two conditions must be met:

1. The digest $\widetilde{w}$ is short. It must hold that $\widetilde{w} \leq \beta(1^\lambda)$
2. The codeword c is reconstructed correctly from the digest, i.e., $c = c'$.

For the PLR-TDP.InvExp$^{\eta}_{\mathcal{A}^{\text{PLR-TDP}},\mathcal{F}}$ to output 1, two similar conditions must be met:

1. The digest w is short. It must hold that $w \leq \eta(1^\lambda)$
2. The x is reconstructed correctly, i.e., $x = x'$.

Figure 2.4: Composability experiment $\mathrm{CExp}^{\Phi}_{\mathcal{A}^{\mathrm{IE}},k}(1^\lambda)$

$\{m_i\}_{i=1}^k$, aux	$\leftarrow \mathcal{A}^{\mathrm{IE}}.\mathrm{Select}(1^\lambda)$.
For $i = 1, \ldots, k$: c_i	$\leftarrow \Phi.\mathrm{Enc}(m_i)$.
w	$\leftarrow \mathcal{A}^{\mathrm{IE}}.\mathrm{Compress}(\{c_i\}_{i=1}^k, \mathtt{aux})$.
$\{c'_i\}_{i=1}^k$	$\leftarrow \mathcal{A}^{\mathrm{IE}}.\mathrm{Expand}(w, \mathtt{aux})$.
Let $\mathcal{I} = \{i : c_i = c'_i\}$. If $\mathcal{I} \neq \emptyset$ and $\lvert w\rvert \leq \sum_{i\in\mathcal{I}} \beta(1^\lambda)$ output 1, otherwise output 0.	

The output of the $\mathcal{A}^{\text{PLR-TDP}}.\mathrm{Leak}$ is a digest $w = \widetilde{w}||s$. If $|\widetilde{w}| \leq \beta(1^\lambda)$, then $|w| \leq \beta(1^\lambda) + \lambda = \eta(1^\lambda)$. Also, if $c = c'$, then it holds that $x = x'$. Because we suppose that the adversary $\mathcal{A}^{\mathrm{IE}}$ successfully breaks the β-incompressible encoding scheme Ψ with a non-negligible probability p, the adversary $\mathcal{A}^{\text{PLR-TDP}}$ breaks the underlying *PLR-TDP* with probability p. That concludes the proof. □

The fact that a single file cannot be compressed below β bits, does not imply that k files cannot be compressed below $k \cdot \beta$ bits. Moran and Wichs in [MW20] conjectured that there exist schemes such that even though one file cannot be compressed below β bits, compressing k files can be done using significantly less than $k \cdot \beta$ bits. That motivates the following definition of the composability property.

Definition 2.5 (Composability)**.** Let $k \in \mathbb{N}$. For (α, β)-incompressible encoding scheme Φ, we define the *composability experiment* $\mathrm{CExp}^{\Phi}_{\mathcal{A}^{\mathrm{IE}},k}(1^\lambda)$ with PPT adversary $\mathcal{A}^{\mathrm{IE}}$ in Figure 2.4.

We say that an incompressible encoding scheme Φ is k-composable if for every PPT adversary $\mathcal{A}^{\mathrm{IE}}$ it holds that the probability of composability experiment $\mathrm{CExp}^{\Phi}_{\mathcal{A}^{\mathrm{IE}},k}(1^\lambda)$ outputting 1 is negligible, i.e.,

$$\Pr[\mathrm{CExp}^{\Phi}_{\mathcal{A}^{\mathrm{IE}},k}(1^\lambda) = 1] \in \mathrm{negl}(\lambda).$$

Conjecture 1. We conjecture that our construction of incompressible encoding can be naturally extended to a composable incompressible encoding scheme.

3. Existence of PLR-TDPs

The natural question is whether PLR-TDPs exist under any standard assumptions. In this chapter, we show that a random trapdoor permutation is a PLR-TDP with overwhelming probability. To this end, we generalize a result from [GGKT05], who studied the one-wayness of random (trapdoor) functions and permutations.

3.1 Preliminaries

In the rest of the thesis, we work with the following ideal model of trapdoor permutations.

Definition 3.1 (Ideal trapdoor permutations)**.** Let $n \in \mathbb{N}$. Consider the set $T_n = \{\tau | \tau = (G, F, F^{-1})\}$ such that the following holds.

- $G \in \Pi_n$ is a permutation on $\{0,1\}^n$. When queried on a trapdoor string td, it produces a public key k, i.e., $G(td) = k$.
- $F : \{0,1\}^n \times \{0,1\}^n \to \{0,1\}^n$ is a function such that for each $k \in \{0,1\}^n$, $F(k, \cdot)$ is a permutation on $\{0,1\}^n$. We denote $F_k := F(k, \cdot)$.
- $F^{-1} : \{0,1\}^n \times \{0,1\}^n \to \{0,1\}^n$ satisfies that $F^{-1}(td, y) = x$, where $G(td) = k$ and $F_k(x) = y$. We denote $F_k^{-1} := F^{-1}(td, \cdot)$.

We denote such T_n as a family of trapdoor permutations of length n.

In the definition above, G represents a *key-generating algorithm*, F represents a *collection of keyed permutations*, and F^{-1} *represents inversion using the trapdoor* td.

Next, we present a version of PLR-TDPs for circuits. There are two differences between the new Definition 3.1 and Definitions 2.1 and 2.2 from the previous chapter. First, $\mathcal{A}^{\mathcal{F}}$.Invert is a circuit of a size at most S. Second, we give an explicit bound ε on the probability of success of the adversary.

Definition 3.2 (S-PLR-TDP Inversion Experiment)**.** Let $n \in \mathbb{N}$. For a family of trapdoor permutations $\mathcal{F}$ on $\{0,1\}^n$, length parameter η and an adversary $\mathcal{A}^{\mathcal{F}} = (\mathcal{A}^{\mathcal{F}}.\text{Leak}, \mathcal{A}^{\mathcal{F}}.\text{Invert})$, where $\mathcal{A}^{\mathcal{F}}$.Invert is a circuit of a size at most S, we define the inversion experiment S-PLR-TDP.$\text{InvExp}^{\eta}_{\mathcal{A}^{\mathcal{F}},\mathcal{F}}$ as it is in Figure 3.1.

Definition 3.3 ((η, S, ε)-PLR-TDP)**.** A family of trapdoor permutations $\mathcal{F}$ is a family of (η, S, ε)-*PLR-TDPs* if for all adversaries $\mathcal{A}^{\mathcal{F}} = (\mathcal{A}^{\mathcal{F}}.\text{Leak}, \mathcal{A}^{\mathcal{F}}.\text{Invert})$, where $\mathcal{A}^{\mathcal{F}}$.Invert is a circuit of a size at most S, it holds that

$$\Pr[S\text{-PLR-TDP.InvExp}^{\eta}_{\mathcal{A}^{\mathcal{F}},\mathcal{F}} = 1] \leq \varepsilon.$$

For the sake of reducing the number of parameters, by (η, S)-PLR-TDP we denote $(\eta, S, 1/S)$-PLR-TDP.

Note that (η, S)-PLR-TDP is in particular S-hard one-way function, meaning that it is hard to invert in the standard sense for circuits of size at most S. The following lemma establishes that any trapdoor permutation $\tau \in T_n$ can be described using at most $(2^n + 1)\log(2^n!)$ bits.

Figure 3.1: Inversion experiment S-PLR-TDP.InvExp$^{\eta}_{\mathcal{A}^{\mathcal{F}},\mathcal{F}}$

td is uniformly random from $\{0,1\}^n$
$k = \mathrm{G}(td)$
y is uniformly random from $\{0,1\}^n$
$x = F_k^{-1}(y)$
$w \leftarrow \mathcal{A}^{\mathcal{F}}.\mathrm{Leak}(k, x, y)$
$x' \leftarrow \mathcal{A}^{\mathcal{F}}.\mathrm{Invert}(k, w, y)$
if $x = x'$ and $w \leq \eta$ output 1, otherwise output 0

Lemma 3.4. *The number of trapdoor permutations $\tau \in T_n$ is $|T_n| = 2^n!(2^n!)^{2^n}$.*

Proof. The set T_n consists of all tuples (G, F, F^{-1}). The G is a set of all permutations on $\{0,1\}^n$. It holds that $|\{0,1\}^n| = 2^n$ thus $|G| = 2^n!$. F is a permutation in the second parameter for each key $k \in \{0,1\}^n$. There are 2^n possible keys k. Therefore $|F| = (2^n!)^{2^n}$. The oracle F^{-1} is fully determined by G and F, which concludes the proof. □

Further on, we utilise the following standard inequalities. Proofs can be found in [MN09, p. 90–95].

Lemma 3.5. *For every $n \in \mathbb{N}$, $n \geq 1$, it holds that*

$$n! \geq e\left(\frac{n}{e}\right)^n.$$

Lemma 3.6. *For every $n, k \in \mathbb{N}$, $n \geq k \geq 1$, it holds that*

$$\binom{n}{k} \leq \left(\frac{en}{k}\right)^k.$$

In particular, we rely on the following corollary.

Corollary 3.7. *Let $a, N \in \mathbb{N}$, $N \geq a \geq 1$. Then the following holds:*

$$a! \geq \left(\frac{a}{e}\right)^a \tag{3.1}$$

and

$$\binom{N}{a} \leq \left(\frac{eN}{a}\right)^a. \tag{3.2}$$

Proof. From Lemma 3.5, it follows that

$$a! \geq e\left(\frac{a}{e}\right)^a \geq \left(\frac{a}{e}\right)^a.$$

The second equation follows directly from Lemma 3.6. □

3.2 Preimage Leakage-Resilience of Random Trapdoor Permutations

In this section, we establish our main theorem on the preimage leakage-resilience of random TDPs.

Theorem 3.8. *For a sufficiently large n, a random trapdoor permutation $\tau \in T_n$ is a $(n/6, 2^{n/5})$-PLR-TDP with probability at least $1 - 2^{2^{n/2}}$.*

To prove the theorem, we utilise a number of lemmata. The most important is Lemma 3.9 which uses the following known fact. Suppose that there exists an efficient invertor for a given trapdoor permutation. Then a description of such permutation requires significantly fewer bits compared to the number of bits necessary to describe an "average" random permutation. Proof of the lemma utilises ideas from [GGKT05], where a similar claim is stated without the additional hints $\{w_{k,y}\}$. In particular, we show that the incompressibility vs compressibility relation holds even in the presence of leakage from the preimage. Proof of the theorem is near the end of this chapter.

Lemma 3.9 (TDP compressibility lemma)**.** *Let $n \in \mathbb{N}$ and let A^τ be a circuit that makes q queries to a trapdoor permutation $\tau \in T_n$, satisfying that*

$$\Pr_{k,y}[A^\tau(w_{k,y}, k, y) = x \wedge F_k(x) = y] \geq \varepsilon,$$

where $\{w_{k,y}\}_{k\in\{0,1\}^n, y\in\{0,1\}^n}$ is a sequence of hints, each of size $|w_{k,y}| \leq \eta$. Then τ can be described using at most

$$1 + 2^n \log(2^n!) + n + 2\log\binom{2^n}{a} + \log\left((2^n - a)!\right) + \eta a$$

bits, where $a = \lceil \varepsilon 2^n/(2q+1) \rceil$.

Proof. Let $N = 2^n$ and let $Q(w_{k,y}, k, y)$ denote the event that $A^\tau(w_{k,y}, k, y)$ queries either $G(td)$ or $F^{-1}(td, y')$, where y' is arbitrary and $G(td) = k$. We say that "$A^\tau(w_{k,y})$ inverts (k, y)" if $A^\tau(w_{k,y}, k, y) = x$ such that $F_k(x) = y$. There are two possibilities: either

$$\Pr_{k,y}[A^\tau(w_{k,y}) \text{ inverts } (k,y) \ \wedge \ Q(w_{k,y}, k, y)] \geq \varepsilon/2 \tag{3.3}$$

or

$$\Pr_{k,y}[A^\tau(w_{k,y}) \text{ inverts } (k,y) \ \wedge \ \overline{Q(w_{k,y}, k, y)}] \geq \varepsilon/2. \tag{3.4}$$

Consider the first case. From the *law of total probability* it follows that there exists $\hat{y}$ such that $\Pr_k[A^\tau(w_{k,\hat{y}}) \text{ inverts } (k,\hat{y}) \ \wedge \ Q(w_{k,\hat{y}}, k, \hat{y})] \geq \varepsilon/2$. We show that G can be described using a "small" number of bits. Let I be the set of at least $\varepsilon N/2$ points $(k, \hat{y})$, on which $A^\tau(w_{k,\hat{y}})$ inverts $(k, \hat{y})$ and $Q(w_{k,\hat{y}}, k, \hat{y})$ occurs. We define a set $K \subseteq I$ using the following process.

1. Let K be empty.
2. We remove the (lexicographically) first element $(k, \hat{y})$ from I and place it into K.

3. Next, we simulate the computation of $A^\tau(w_{k,\hat{y}}, k, \hat{y})$. Since $Q(w_{k,\hat{y}}, k, \hat{y})$ occurs, we know that there is a query i of the form $G(td)$ or $F^{-1}(td, y')$, where $G(td) = k$.

4. We look at all the $i - 1$ preceding queries.

   - For queries of the form $G(td')$ with answer k' we remove $(k', \hat{y})$ from I.
   - For queries of the form $F^{-1}(td', y')$ let $G(td') = k'$ and remove $(k', \hat{y})$ from I.

5. If the set K contains a pairs $(k, \hat{y})$, i.e., $|K| = a$, we stop the process. Otherwise, we repeat steps 2–5.

It is possible to finish this process because, at each step, we add one element to K and remove at most q elements from I. Since the original size of I is greater than $\varepsilon N/2$, we can perform at least $\varepsilon N/2q$ steps before the set I is empty. In other words, we are able to construct K of size at least $\lceil \varepsilon N/2q \rceil > \lceil \varepsilon N/(2q+1) \rceil = a$.

Let $K_k = \{k : (k, \hat{y}) \in K\}$. We claim that the permutation G is fully specified given A^τ, $\hat{y}$, description of K and $L := G^{-1}(K_k)$, the values of G on $\{0,1\}^n \setminus L$, the values $w_{k,\hat{y}}$ for all $k \in K_k$, and the values of F on all points. We show that it is indeed possible to invert G for any value $(k, \hat{y})$. For $(k, \hat{y}) \notin K$ the value $G^{-1}(k) = td$ is explicitly given. For all pairs $(k, \hat{y}) \in K$, the values of $G^{-1}(k)$ can be computed sequentially as follows.

Simulate the computation of $A^\tau(\hat{w}_k, k, \hat{y})$. By construction of K, we know that $A^\tau(w_{k,\hat{y}}, k, \hat{y})$ will query either:

1. G on a point $td' \notin L$,
2. G on a point $td' \in L$ for which $G(td') <_{\text{lex}} k$,
3. G on the point $td \in L$ for which $G(td) = k$
4. F^{-1} on a point (td', y'), where $G(td') \notin K_k$,
5. F^{-1} on a point (td', y') satisfying that $G(td') \in K_k,\ G(td') <_{\text{lex}} k$,
6. F^{-1} on the point (td, y') for which $G(td) = k$,
7. F on any point.

Thus, we either have enough information to continue the simulation, or we have queried on the point td for which $G(td) = k$. For clarity, we describe what happens in each of these cases. Note that we reconstruct sequentially for elements $(k, \hat{y}) \in K$ based on their lexicographical order.

1. The value for $G(td')$ is stored for $td' \in \{0,1\}^n \setminus L$. Thus, it can be immediately computed.

2. The value for $G(td')$ is not stored for $td' \in L$. Nevertheless, if it is the case that $G(td') <_{\text{lex}} k$, then the value of $G(td')$ must have been computed before – during the reconstruction. Therefore, it can be computed using the knowledge from reconstructing the values of G so far.

Figure 3.2: Number of bits required to describe τ in the first case

K	$\log \binom{N}{a}$
L	$\log \binom{N}{a}$
G on $\{0,1\}^n \setminus L$	$\log\big((N-a)!\big)$
$\{w_{k,\hat{y}}\}_{k \in K_k}$	ηa
$\hat{y}$	n
F at all points	$N \log(N!)$
first case	1

3. We are unable to compute the value $G(td)$ because it was not stored and it was not previously reconstructed. Thus, we deduce that $G(td) = k$.

4. Similarly to the first case, the value of td' is stored. Let us denote $G(td') = k'$ We know that $F^{-1}(td', \cdot) = (F_{k'})^{-1}$. The function $F_{k'}$ is stored for all points and $F^{-1}(td', \cdot)$ is fully determined by $F_{k'}$, thus $F^{-1}(td', y')$ can be computed.

5. Similarly to the second case, we do not know $G(td')$ from stored values but from the previous reconstruction. We evaluate $G(td') = k'$ and determine $F^{-1}(td', y')$ using the knowledge of $(F_{k'})^{-1}$.

6. We are unable to compute the value $G(td)$ because it was not stored and it was not previously reconstructed. Thus, we deduce that $G(td) = k$. Therefore, we can determine $F^{-1}(td, \cdot)$ using the knowledge of $(F_k)^{-1}$.

7. The function F is stored for all points. Thus, it is possible to compute it directly.

Descriptions of K and $G^{-1}(K_k)$ require $\log \binom{N}{|K|} = \log \binom{N}{a}$ bits each. The set G^{-1} on $\{0,1\}^n \setminus K_k$ requires $\log\big((N-a)!\big)$ bits, storing hints $w_{k,\hat{y}}$ for each $k \in K_k$ requires $|w| \cdot |K| = \eta a$ bits. Full description of the function F requires $N \log(N!)$ bits, t bits are needed to specify $\hat{y}$ and a single bit is needed to denote that we are in the first case. For clarity, all descriptions and their sizes are presented in Figure 3.2. Note that G and F^{-1} are fully determined by G^{-1} and F, respectively. Thus, we can completely specify τ using at most the number of bits claimed.

Next, consider the second case, i.e., when eq. (3.4) holds. From the *law of total probability*, it follows that there exists $\hat{k}$ such that $\Pr_y[A^\tau(w_{\hat{k},y})$ inverts $(\hat{k}, y) \wedge \overline{Q(w_{\hat{k},y}, \hat{k}, y)}] \geq \varepsilon/2$. Analogously to the first case, we show that $F(\hat{k}, \cdot)$ can be described using a "small" number of bits. In this case, we define I as the set of at least $\varepsilon N/2$ points $(\hat{k}, y)$ such that $A^\tau(w_{\hat{k},y})$ inverts $(\hat{k}, y)$ and event $Q(w_{\hat{k},y}, \hat{k}, y)$ does not happen. Next, we define a set $Y \subseteq I$ by the following process.

1. Let Y be empty.

2. We remove the (lexicographically) first element $(\hat{k}, y)$ from I and put it into Y.

3. We simulate the computation of $A^\tau(w_{\hat{k},y}, \hat{k}, y)$.

4. Consider the $\ell \leq q$ queries made by A of the form $\{F(\hat{k}, x_i)\}$ with corresponding answers y_i.

   - If $y \notin \{y_i\}_{i=1}^{\ell}$, we remove $\{(\hat{k}, y_i)\}_{i=1}^{\ell}$ from I.
   - If there exists j such that $y = y_j$, we remove (only) $\{(\hat{k}, y_i)\}_{i=1}^{j-1}$ from I.

5. If the size of Y is $|Y| = a$, we stop the process. Otherwise, we repeat steps 2–5.

Note that it is not possible to have empty I before Y is created. At any iteration of the process we add one element $(\hat{k}, y)$ to the set Y and remove at most q elements from I. The original size of I is at least $\varepsilon N/2$. Therefore, we can perform at least $\varepsilon N/2q \geq \lceil \varepsilon N/(2q+1) \rceil = a$ iterations.

Let X be a set satisfying that $F(\hat{k}, X) = Y$. Similarly to the first case, the permutation $F(\hat{k}, \cdot)$ is fully specified given A, sets X and Y, the values $w_{\hat{k},y}$ for $y \in Y$, the values of $F(\hat{k}, \cdot)$ on $\{0,1\}^n \setminus X$, the values of G at all points and the values of $F(k, \cdot)$ for all $k \neq \hat{k}$.

We show that the information provided is sufficient to reconstruct τ. Functions G and F_k for $k \neq \hat{k}$ are fully specified. Thus, we focus on $F_{\hat{k}}$. For all $y \in \{0,1\}^n \setminus X$, the values of $F_{\hat{k}}$ are explicitly provided. For $y \in X$ we simulate $A^\tau(w_{\hat{k},y}, \hat{k}, y)$. We know that $Q(w_{\hat{k},y}, \hat{k}, y)$ does not happen, i.e., $A^\tau(w_{\hat{k},y}, \hat{k}, y)$ does not make queries on $G(td)$) or $F^{-1}(td, y')$, where y' is arbitrary and $G(td) = \hat{k}$. All (other) queries on G can be evaluated directly because the G is stored. Similarly, we can evaluate all allowed queries (td', y') on F^{-1}. This is because $G(td') = k \neq \hat{k}$ and thus, $F^{-1}(td', \cdot)$ is fully determined by $(F_k)^{-1}$, which is stored for all $k \neq \hat{k}$. Also, we can evaluate all queries on F_k for $k \neq \hat{k}$. Therefore, we focus on queries on $F_{\hat{k}}$. There are three possible options:

1. $F_{\hat{k}}$ is queried on $x' \notin X$,

2. $F_{\hat{k}}$ is queried on $x' \in X$ such that $F_{\hat{k}}(x') <_{\text{lex}} y$,

3. $F_{\hat{k}}$ is queried on $x \in X$ such that $F_{\hat{k}}(x) = y$.

In the first case, we are able to compute the value directly. In the second case, the value was gained previously during the reconstruction. The order of the query $\leq_{\text{lex}} y$ is granted by the construction of the set Y and the sequential approach. In the third case, we are not able to evaluate the query from stored or previously computed values. Thus, we deduce that $F_{\hat{k}}(x) = y$. Therefore, F can be reconstructed for all possible values. This implies the reconstruction of F^{-1} for all possible values.

The table in Figure 3.3 presents all descriptions and their required sizes in bits.

Figure 3.3: Number of bits required to describe τ in the second case

X	$\log \binom{N}{a}$
Y	$\log \binom{N}{a}$
$\{w_{\hat{k},y}\}_{(\hat{k},y)\in Y}$	ηa
$\hat{k}$	n
G at all points	$\log(N!)$
$F(\hat{k},\cdot)$ on $\{0,1\}^n \setminus X$	$\log\big((N-a)!\big)$
$F(k,\cdot)$ for $k \neq \hat{k}$	$(N-1)\log(N!)$
second case	1

Adding all sizes together, we get the total maximal number of bits required to describe the trapdoor permutation τ, which is

$$1 + N\log(N!) + n + 2\log\binom{N}{a} + \log\big((N-a)!\big) + \eta a.$$

This concludes the proof. □

To prove Theorem 3.8, we need to upper bound the number of circuits of size at most S. For that purpose, we state the following lemmata. Lemma 3.10 comes from [GGKT05, p. 13] and provides a somewhat weaker claim. It gives an upper bound on the number of circuits of size *exactly* S. Lemma 3.11 provides the generalised claim for the circuits of size *at most* S.

Lemma 3.10 ([GGKT05] a bound on the number of circuits of size S)**.** *The number of circuits of size S having input/output length n and oracle access to a function $f\colon \{0,1\}^n \to \{0,1\}^n$ is at most*

$$2^{2S+(S+1)n\left(\log(Sn+n)+1\right)}.$$

Proof of this lemma is in [GGKT05, p. 13].

Lemma 3.11 (a bound on the number of circuits of size at most S)**.** *The number of circuits of size at most S having input/output length n and oracle access to a function $f\colon \{0,1\}^n \to \{0,1\}^n$ is at most*

$$2^{2S+(S+1)n\left(\log(Sn+n)+1\right)+\log(S)}.$$

Proof. We upper-bound the size of all possible circuits of length $1, 2, \ldots, S$. We estimate each of those by the number of circuits of length S and sum them

together.

$$\begin{aligned}\sum_{i=1}^{S} 2^{2i+(i+1)n\left(\log(in+n)+1\right)} &\leq S \cdot 2^{2S+(S+1)n\left(\log(Sn+n)+1\right)},\\ &\leq 2^{\log(S)} \cdot 2^{2S+(S+1)n\left(\log(Sn+n)+1\right)},\\ &\leq 2^{2S+(S+1)n\left(\log(Sn+n)+1\right)+\log(S)}.\end{aligned}$$

□

In the following, we give the proof of Theorem 3.8.

Proof of Theorem 3.8. From the TDP compressibility lemma (Lemma 3.9), we know that every trapdoor permutation τ that can be "efficiently inverted" can be expressed using at most $\kappa = 1 + N\log(N!) + n + 2\log\binom{N}{a} + \log\left((N-a)!\right) + \eta a$ bits. Therefore, there are at most 2^{κ} trapdoor permutations $\tau \in T_n$ that can be "efficiently inverted". We compute the fraction of all such permutations, where the number of all permutations follows from Lemma 3.4.

$$\begin{aligned}\frac{2^{\kappa}}{N!(N!)^N} &= \frac{(N!)^N 2^{n+1+\eta a}(N-a)!\binom{N}{a}^2}{N!(N!)^N},\\ &= \frac{2^{n+1+\eta a}(N-a)!\binom{N}{a}^2}{N!},\\ &= \frac{2^{n+1+\eta a}\binom{N}{a}}{a!}.\end{aligned}$$

We want to show that for a sufficiently large n, the fraction can be bounded by 2^{-a}. From the claim of the theorem, we have $\varepsilon = 2^{-n/5}$. Note that any circuit of size $S = 2^{n/5}$ makes at most $q = 2^{n/5}$ queries. Thus, $a = \lceil \varepsilon 2^n/(2q+1)\rceil = \lceil 2^{-n/5}2^n/(2^{n/5+1}+1)\rceil \geq N^{3/5}/4$. We proceed with the computation.

$$\begin{aligned}\frac{2^{n+1+\eta a}\binom{N}{a}}{a!} &\leq \left(\frac{2^{\frac{n+1}{a}} e^2 N 2^{\eta}}{a^2}\right)^a,\\ &\leq \left(\frac{2e^2 N 2^{\eta}}{a^2}\right)^a,\\ &\leq \left(\frac{32e^2 N 2^{\eta}}{N^{\frac{6}{5}}}\right)^a,\\ &\leq \left(\frac{32e^2 2^{\eta}}{N^{\frac{1}{5}}}\right)^a,\end{aligned}$$

where the first inequality follows from Corollary 3.7, specifically eq. (3.1) and eq. (3.2).

Consider now that the length of a hint is $\eta = n/6$ bits. This implies

$$\left(\frac{32e^2 2^{\eta}}{N^{\frac{1}{5}}}\right)^a = \left(\frac{32e^2 N^{\frac{1}{6}}}{N^{\frac{1}{5}}}\right)^a = \left(\frac{32e^2}{N^{\frac{1}{30}}}\right)^a = \left(\frac{32e^2}{(2^n)^{\frac{1}{30}}}\right)^a.$$

Therefore, for a sufficiently large n, it holds that

$$\frac{2^{n+1+\eta a}\binom{N}{a}}{a!} \leq \left(\frac{1}{2}\right)^a = 2^{-a} \leq 2^{-\frac{N^{3/5}}{4}}.$$

By Lemma 3.11, there are at most $2^{2S+(S+1)n\left(\log(Sn+n)+1\right)+\log(S)} = \widetilde{O}(N^{1/5})$ circuits of size at most S, where the $\widetilde{O}$ notation supresses polylogarithmic factors. We apply union bound and show that the probability (over a random choice of $\tau \in T_n$) that there exists a circuit of size at most $S = 2^{n/5}$, satisfying that the circuit inverts τ with probability greater than ε, is

$$\widetilde{O}(N^{1/5}) \cdot 2^{-a} < \widetilde{O}(N^{1/5}) \cdot 2^{-\frac{N^{3/5}}{4}} < 2^{-N^{1/2}}$$

for all sufficiently large N.

Note that we do not care about how the hint w is created. Without loss of generality, we can assume that the hint provided is the best possible, and the statement still holds. Consider now the inversion experiment from Definition 3.2. We have the hints created by $\mathcal{A}^{\mathcal{F}}$.Leak and we invert using the circuit $\mathcal{A}^{\mathcal{F}}$.Invert. Suppose the probability of the existence of a successful invertor $\mathcal{A}^{\mathcal{F}}$.Invert is greater than $2^{-N^{1/2}}$, where by "successful" we mean that the probability of inverting correctly is greater than ε. This contradicts the previous part of the proof. Thus, the probability of existence of a successful invertor is lower than $2^{-N^{1/2}}$. The theorem follows. □

Theorem 3.8 shows that for a particular set of parameters, we can have hints that are long 1/6 of the length of the trapdoor permutation. The natural question is if this can be improved. In the following theorem, we show that by adjusting the size of the circuit S and probability ε, we can have hints of length equal to any fraction (less than 1) of the length of the trapdoor permutation (for a sufficiently large n).

Theorem 3.12. *Let $k \in \mathbb{N}$, $k \geq 5$. For a sufficiently large n, a random trapdoor permutation $\tau \in T_n$ is $(n\frac{k-4}{k+1}, 2^{n/k})$-PLR-TDP with probability at least $1 - 2^{2^{n/2}}$.*

Proof. The proof is very similar to the proof of Theorem 3.8. We need to adjust the variables:

$$\begin{aligned}
S &= q = 2^{n/k},\\
\varepsilon &= 2^{-n/k},\\
\eta &= \frac{k-4}{k+1}n,\\
a &= \lceil \varepsilon 2^n/(2q+1) \rceil,\\
&= \lceil 2^{-\frac{n}{k}} 2^n/(2^{\frac{n}{k}+1}+1) \rceil,\\
&\geq N^{\frac{k-2}{k}}/4.
\end{aligned}$$

Let $k \in \mathbb{N}, k \geq 5$. We proceed with the computation as in the proof of Theorem 3.8 with the new values.

$$\begin{aligned}
\frac{2^{n+1+\eta a}\binom{N}{a}}{a!} &\leq \left(\frac{2^{\frac{n+1}{a}} e^2 N 2^{\eta}}{a^2}\right)^a, \\
&\leq \left(\frac{2e^2 N 2^{\eta}}{a^2}\right)^a, \\
&\leq \left(\frac{32e^2 N 2^{\eta}}{N^{\frac{2k-4}{k}}}\right)^a, \\
&\leq \left(\frac{32e^2 2^{\eta}}{N^{\frac{k-4}{k}}}\right)^a.
\end{aligned}$$

Consider now that the length of a hint is $\eta = n\frac{k-4}{k+1}$ bits. This implies

$$\left(\frac{32e^2 2^{\eta}}{N^{\frac{k-4}{k}}}\right)^a = \left(\frac{32e^2 N^{\frac{k-4}{k+1}}}{N^{\frac{k-4}{k}}}\right)^a = \left(\frac{32e^2}{N^{\frac{k-4}{k(k+1)}}}\right)^a = \left(\frac{32e^2}{\left(2^n\right)^{\frac{k-4}{k(k+1)}}}\right)^a.$$

Therefore, for a sufficiently large n, it holds that

$$\frac{2^{n+1+\eta a}\binom{N}{a}}{a!} \leq \left(\frac{1}{2}\right)^a = 2^{-a} \leq 2^{-\frac{N^{(k-2)/k}}{4}} \leq 2^{-\frac{N^{3/5}}{4}},$$

where the last inequality holds for all $k \geq 5$.

Note that we have arrived at the same bound $\left(2^{-\frac{N^{3/5}}{4}}\right)$ as in the proof of Theorem 3.8. Without loss of generality, we upper-bound the size of the circuit $S \leq 2^{n/5}$. The rest of the proof is identical to the proof of Theorem 3.8. □

Conclusion

The goal of our thesis is to present preimage leakage-resilient trapdoor permutations (PLR-TDPs) and their applications for cryptographic purposes with a focus on proofs of storage replication and incompressible encodings. One of the questions suggested by our thesis is whether trapdoor permutations, which have this property, exist under standard assumptions.

We introduce the reader to all necessary definitions of TDPs and highlight important applications. We use the newly defined PLR-TDPs to construct a simple incompressible encoding scheme. The advantage of using PLR-TDPs over regular TDPs is the reduced complexity of the construction and the security analysis. We also show that fully random TDPs have the preimage leakage-resilience property in a very strong sense.

In their work, van Dijk et al. conjectured that *practical* TDPs (namely RSA) are preimage leakage-resilient with respect to $n - \Omega(\log n)$ bits of leakage. Our results show that *ideal* TDPs are preimage leakage-resilient in a very strong sense with respect to $n^{\frac{k-4}{k}}$ bits of leakage. Thus, they support the conjecture that practical TDPs are preimage leakage-resilient, too. In particular, our results are the first step towards formally secure and time-efficient schemes in the future.

There are various open problems. For instance, whether the incompressible encoding scheme defined in Chapter 2 can be extended into a *composable* incompressible encoding scheme. A major open question is whether practical TDPs, such as the RSA or Rabin permutations, are preimage leakage-resilient under standard assumptions.

Bibliography

[CKGS98] Benny Chor, Eyal Kushilevitz, Oded Goldreich, and Madhu Sudan. Private information retrieval. *Journal of the ACM (JACM)*, 45(6):965–981, 1998.

[DGO19] Ivan Damgård, Chaya Ganesh, and Claudio Orlandi. Proofs of replicated storage without timing assumptions. In Alexandra Boldyreva and Daniele Micciancio, editors, *Advances in Cryptology - CRYPTO 2019 - 39th Annual International Cryptology Conference, Santa Barbara, CA, USA, August 18-22, 2019, Proceedings, Part I*, volume 11692 of *Lecture Notes in Computer Science*, pages 355–380. Springer, 2019.

[GGKT05] Rosario Gennaro, Yael Gertner, Jonathan Katz, and Luca Trevisan. Bounds on the efficiency of generic cryptographic constructions. *SIAM journal on Computing*, 35(1):217–246, 2005.

[GLW20] Rachit Garg, George Lu, and Brent Waters. New techniques in replica encodings with client setup. In Rafael Pass and Krzysztof Pietrzak, editors, *Theory of Cryptography - 18th International Conference, TCC 2020, Durham, NC, USA, November 16-19, 2020, Proceedings, Part III*, volume 12552 of *Lecture Notes in Computer Science*, pages 550–583. Springer, 2020.

[Gol01] Oded Goldreich. *Foundations of Cryptography*, volume 1. Cambridge University Press, 2001.

[KL20] Jonathan Katz and Yehuda Lindell. *Introduction to modern cryptography*. CRC press, 2020.

[KO00] Eyal Kushilevitz and Rafail Ostrovsky. One-way trapdoor permutations are sufficient for non-trivial single-server private information retrieval. In *Advances in Cryptology—EUROCRYPT 2000: International Conference on the Theory and Application of Cryptographic Techniques Bruges, Belgium, May 14–18, 2000 Proceedings 19*, pages 104–121. Springer, 2000.

[KR19] Yael Tauman Kalai and Leonid Reyzin. A survey of leakage-resilient cryptography. In Oded Goldreich, editor, *Providing Sound Foundations for Cryptography: On the Work of Shafi Goldwasser and Silvio Micali*, pages 727–794. ACM, 2019.

[LMRS04] Anna Lysyanskaya, Silvio Micali, Leonid Reyzin, and Hovav Shacham. Sequential aggregate signatures from trapdoor permutations. In *Advances in Cryptology-EUROCRYPT 2004: International Conference on the Theory and Applications of Cryptographic Techniques, Interlaken, Switzerland, May 2-6, 2004. Proceedings 23*, pages 74–90. Springer, 2004.

[MN09] Jiří Matoušek and Jaroslav Nešetřil. *Kapitoly z diskrétní matematiky*. Čtvrté, upravené a doplněné vydání. Karolinum, Praha, 2009.

[MW20] Tal Moran and Daniel Wichs. Incompressible encodings. In Daniele Micciancio and Thomas Ristenpart, editors, *Advances in Cryptology - CRYPTO 2020 - 40th Annual International Cryptology Conference, CRYPTO 2020, Santa Barbara, CA, USA, August 17-21, 2020, Proceedings, Part I*, volume 12170 of *Lecture Notes in Computer Science*, pages 494–523. Springer, 2020.

[Pai99] Pascal Paillier. Public-key cryptosystems based on composite degree residuosity classes. In *International conference on the theory and applications of cryptographic techniques*, pages 223–238. Springer, 1999.

[Rab79] Michael O. Rabin. Digitalized signatures and public-key functions as intractable as factorization. *Massachusetts Institute of Technology, Technical Memo TM-212*, 1979.

[RSA78] Ronald L. Rivest, Adi Shamir, and Leonard Adleman. A method for obtaining digital signatures and public-key cryptosystems. *Communications of the ACM*, 21(2):120–126, 1978.

[vDJO+12] Marten van Dijk, Ari Juels, Alina Oprea, Ronald L. Rivest, Emil Stefanov, and Nikos Triandopoulos. Hourglass schemes: how to prove that cloud files are encrypted. In *Proceedings of the 2012 ACM conference on Computer and communications security*, pages 265–280, 2012.

[Yao82] Andrew C. Yao. Theory and application of trapdoor functions. In *23rd Annual Symposium on Foundations of Computer Science (SFCS 1982)*, pages 80–91. IEEE, 1982.

List of Figures

List of Abbreviations

CExp	composability experiment
CompExp	compression experiment
IE	incompressible encoding
PKE	public-key encryption
PLR-TDPs	preimage leakage-resilient trapdoor permutations
PPT	probabilistic polynomial-time
PRG	pseudorandom generator
RO	random oracle
TDPs	trapdoor permutations