BERGISCHE
UNIVERSITÄT
WUPPERTAL

Doctoral Dissertation

Time Reveals The Truth - More Efficient Constructions of Timed Cryptographic Primitives

Peter Chvojka

March 25, 2021

Submitted to the
School of Electrical, Information and Media Engineering
University of Wuppertal

for the degree of
Doktor-Ingenieur (Dr.-Ing.)

The PhD thesis can be quoted as follows:

urn:nbn:de:hbz:468-20210902-111552-1
[http://nbn-resolving.de/urn/resolver.pl?urn=urn%3Anbn%3Ade%3Ahbz%3A468-20210902-111552-1

DOI: 10.25926/n2k9-cj05
[https://doi.org/10.25926/n2k9-cj05]

Peter Chvojka
Place of birth: Levice, Slovakia
Author's contact information:
chvojka@uni-wuppertal.de

Thesis Advisor:	**Prof. Dr.-Ing. Tibor Jager** University of Wuppertal, Wuppertal, Germany
Second Examiner:	**Prof. Dr. Johannes Blömer** Paderborn University, Paderborn, Germany
Thesis submitted:	March 25, 2021
Thesis defense:	June 25, 2021
Last revision:	August 24, 2021

Acknowledgements

I am deeply thankful to my supervisor Tibor Jager for giving me the opportunity to pursue a PhD in his group, for all the advice he has given to me and for all his support. It was a pleasure to work with such an inspiring and phenomenal person and I can only wish that our collaboration will continue even further.

My thanks also belong to Uli Wagner and Krzysztof Pietrzak for helping me to find the current PhD position.

I would like to thank my co-authors Daniel Slamanig and Christoph Striecks for our pleasant collaboration and joyful discussions.

An inseparable part of doctoral studies is not only hard work but also dozens of funny situations in a working environment. I would like to thanks my colleagues Pascal Bemmann, Gareth T. Davies, Denis Diemert, Jan Drees, Kai Gellert, Tobias Handirk, Saqib A. Kakvi, Lin Lyu, David Niehues and Jutta Maerten for their help, support and making the whole PhD experience more interesting and joyful. A special thanks to Rafael Kurek for the best conference experience in Brisbane and all fun we had while sharing an office.

I am thankful to the whole research group of Johannes Blömer, especially to Jan Bobolz and Fabian Eidens for all insightful discussions and for broadening my knowledge in cryptography.

Starting to live in a new country brings many challenges and since the life of a researcher continues also after the working hours, it is important to find a meaningful and at the same time fulfilling way how to spend free time. This turns out to be easier than I would expect. Since my first day in Paderborn, I have met many amazing people who cared about me all this time and give me so much that it is hard to express it only by words. I am sincerely thankful to Lewis Barhoumi, Jan Bohnenkamp, Lotte Böcker, Jonas Brand, Alex Briem, Lars Bürmann, Lukas Filges, Alina Sophie Goedeke, Lorenz Gamm, Artur Heimbuch, Dennis Laiker, Lukas Lindemann, Markus Muschiol, Justin Müller, Laura Nikolov, Alica and Matúš Rehák, Markus Reimann, Jule van Riswyck, Simon Scharrenbach, Andreas Schukow, Juraj and Sabrina Somorovsky, Mara Spiekenheuer, Florian Unruhe, Tobias Volbert, Leo Wehling, Gabriel Wiebe, Juliane Wiebe and Margarita Wiebe for all love you have given me, every second you have spent with me, all encouragement you have provided to me and every smile you have dedicated to me. You have become not only my friends but my family. You have stood next to me in every difficult situation during the last four and half years and help me to over-

come them. Thanks to you I was able to push my limits beyond my imagination and overcome my fears. You are my inspiration, motivation, the people which I admire and appreciate. You are the reason why I call Paderborn my home.

Von der Familie Kroll habe ich eine enorme Gastfreundschaft erfahren, wofür ich sehr dankbar bin.

It was a hard decision to leave the country and, more importantly, my friends and family there. Even though we have seen each other only a couple of times in the past years, you have permanently been in touch with me. My gratitude belongs to Šimon Herc, Aurel Korec, Lukáš Marčák a Marta Marčáková, Vincent Mäsiar Vargas, Veronika Pacalová, Lenka Polinová, Andrej Salaj, Daniela Salajová, and Michal Stanček for your everlasting support, great hospitality whenever I am in Slovakia, for every message and call you dedicated to me. It is a true blessing to have such friends as you are.

Zaverečné slová by som chcel venovať mojej rodine. Predovsetkým sa chcem poďakovať mojim rodičom Alene a Danielovi a mojim bratom Romanovi a Ivanovi. Milí rodičia, úprimne vám ďakujem za všetkú lásku, ktorou ma celý život zahŕňate. Napriek ťažkým finančným pomerom v našej rodine, vďaka vašej obetavosti som bol schopný spolu s mojimi bratmi úspešne ukončiť vysokoškolské štúdium. Vďaka za vašu výchovu, za všetko čo ste ma naučili, za všetok čas, ktorý ste mi venovali. Vy ste položili základy mojho vzdelania, keď ste po večeroch pri mne sedávali a učili sa so mnou. Taktiež sa chcem poďakovať za prebdené noci, ktoré ste pri mne strávili, keď som sa necítil zdravotne najlepšie, za čas, ktorý ste so mnou strávili u lekárov len aby sa mi polepšilo. Na tom akým som dnes človekom majú zásluhu aj moji dvaja bratia, ktorí vždy pri mne stáli a sú mi svojou vytrvalosťou, pokorou a dobrosrdečnosťou veľkým vzorom. Obrovská vďaka patrí mojim starým rodičom, Jánovi, Hedvige, Jánovi a Márii, ktorí ma neustále podporovali v každom rozhodnutí a predovšetkým v rozhodnutí študovať v zahraničí.

Abstract

The possibility to send information "into the future" has turned out to be surprisingly useful. It does not only allow revealing data at a required time which is needed for instance when publishing the results of a census or elections, but it helps to achieve properties that are difficult to achieve otherwise, e.g., ensuring that results of some protocol are unbiased. Currently, there are three known approaches that enable information to be kept secret until some specified future time has not passed. Historically the first approach suggested by May [May93] relies on a trusted entity which at a required time point essentially reveals the secret information. Later, the approach which requires execution of a sequential computation has been proposed by Rivest *et al.* [RSW96] who used the term Time-Lock Puzzle (TLP) to denote it. The third approach, known as Time-Lock Encryption (TLE) introduced by Liu *et al.* [LJKW18], aims to avoid both the dependency on a trusted entity and the need to perform a sequential computation. This thesis focuses on the latter two approaches with the objective of improving their practicality.

The main barrier in deploying time-lock encryption is that its constructions are not practical yet. Specifically, the only known construction of TLE requires Extractable Witness Encryption (EWE) and there are currently no efficient instantiations of this primitive. On the other hand, time-lock puzzles can be instantiated efficiently, however it is often necessary to encrypt several messages independently, which results in the wasting of computational resources.

To overcome the above-mentioned limitations, we observe that many applications allow for a trusted setup. Therefore we equip TLE and TLPs with a trusted setup and examine the possible efficiency gains. We denote TLE and TLPs with a trusted setup by *Offline* Time-Lock Encryption (OTLE) and *Timed-Release Encryption* (TRE), respectively.

Instead of focusing directly on OTLE, we introduce a new notion of *Extractable Offline Witness Encryption* (EOWE) which is extractable witness encryption with a trusted setup. OTLE can be directly built from EOWE in a similar way as TLE can be built from EWE. But since EOWE is a more general notion, it has also broader applicability than OTLE. We remark that the notion of Offline Witness Encryption (OWE) has been originally proposed by Abusalah *et al.* [AFP16], who build this primitive by combining indistinguishability obfuscation with a specific type of IND-CCA-secure Public-Key Encryption (PKE). The advantage of the resulting construction in comparison to existing constructions of witness encryp-

tion is that the most expensive computation is executed in the trusted setup which yields very efficient encryption. To build extractable OWE we proceed similarly as in [AFP16]. Concretely, we rely on extractability obfuscation in combination with a new variant of puncturable encryption [GM15] which we introduce. Additionally, by replacing extractability obfuscation with indistinguishability obfuscation in our construction, one directly obtains OWE, however with stronger security guarantees than the original construction. Moreover, we propose an efficient instantiation of a puncturable encryption scheme which results in a shorter ciphertext size than the original proposal of Abusalah *et al.*

The main idea of our timed-release encryption scheme is providing the ability to encrypt an arbitrary number of messages with respect to only one time-lock puzzle in a way that all these messages are decryptable using the solution of the puzzle. Hence, TRE achieves the so-called "solve one, get many for free" property, which has not been achievable before. We show that such an encryption scheme can be generically built from any TLP and any IND-CPA-secure PKE scheme. To further extend the capabilities and practicality of TRE, we propose a notion of *sequential* timed-release encryption (sTRE) which allows encryption of messages with respect to different time parameters. As a building block for sTRE we introduce a notion of *sequential* time-lock puzzle (sTLP). sTLP allows generation of puzzles which corresponds to an increasing sequence of time parameters. These puzzles can be solved using only one sequential computation which provides the solutions at points in time that are determined by time parameters. In this way, the task of solving puzzles can be delegated to a public server which makes TRE more economically and ecologically sustainable.

Lastly, we observe that our notion of TRE is similar to the concurrently introduced notion of Timed Public-Key Encryption (TPKE) by Katz *et al.* [KLX20] which has two types of decryption: slow decryption, with runtime determined by a time parameter, and fast decryption, with runtime independent of the time parameter. TPKE serves as a building block for non-malleable timed commitments and therefore it aims to provide non-malleability of ciphertexts. The known construction of TPKE has an inefficient encryption algorithm whose runtime is essentially the time that is necessary to solve the related time-lock puzzle and hence is proportional to the time parameter. We apply techniques used in constructing TRE to TPKE and obtain two constructions with efficient encryption. The first construction is generic with encryption time which depends logarithmically on the time parameter. The runtime of encryption in our second construction is independent of the time parameter, however, this construction is not generic.

Contents

Abstract iii

1 Introduction 1
1.1 Publication Overview 6

2 Preliminaries 7
2.1 Notation 7
2.2 NP Language 7
2.3 Cryptographic Primitives 8
2.3.1 One-Time Signatures 8
2.3.2 Public-Key Encryption 9
2.3.3 Homomorphic Encryption 10
2.3.4 Tag-Based Encryption 11
2.3.5 Functional Encryption 12
2.3.6 Obfuscation 13
2.3.7 Randomized Encoding 15
2.4 Bilinear Groups 17
2.5 Assumptions 17
2.6 The Random Oracle Model 20
2.7 The Strong Algebraic Group Model 21

3 (Extractable) Offline Witness Encryption 23
3.1 Introduction, Contributions and Related Work 23
3.2 Definitions of (Extractable) Offline Witness Encryption 28
3.3 Puncturable Tag-Based Encryption 30
3.4 Offline Time-Lock Encryption 32
3.5 Constructions 34
3.5.1 Construction of OWE 34
3.5.2 Construction of EOWE 38
3.6 Instantiation of Puncturable Encryption Scheme 43
3.6.1 PE from Kiltz's Tag-Based Encryption Scheme 44
3.7 Conclusion and Open Problems 46

4 Encryption Schemes from Time-Lock Puzzles **49**
4.1 Introduction, Contributions and Related Work 50
4.2 Time Lock-Puzzles 57
4.2.1 Instantiating TLPs from Sequential Squaring 60
4.2.2 Instantiating TLPs from Randomized Encodings 60
4.3 Sequential Time-Lock Puzzles 62
4.3.1 Instantiating Sequential TLPs from Sequential Squaring . . 64
4.3.2 Hardness of the Gap Sequential Squaring Assumption 67
4.4 (Sequential) Timed-Release Encryption 70
4.4.1 Basic TRE Construction 71
4.4.2 Sequential TRE 74
4.4.3 Properties of TRE Construction 77
4.4.4 Integrating Timed-Release Features into Functional Encryption 81
4.4.5 Applications 86
4.5 Timed Public-Key Encryption 89
4.5.1 Weak IND-CCA-secure TPKE 91
4.5.2 Generic Construction of IND-CCA-secure TPKE 91
4.5.3 Construction of IND-CCA-secure TPKE from SSSA 100
4.6 Conclusion and Open Problems 106

Bibliography **109**

1 Introduction

The topic of this dissertation can be perhaps surprisingly expressed by the words of a Roman Stoic philosopher:

> We should always allow some time to elapse, for time discloses the truth.
>
> *Seneca*

These words are almost 2000 years old, but despite that their main idea plays a crucial role in recent avenues of cryptography. Before we describe this idea in more detail, let's start at the beginning and explain how the notion of time has been introduced in cryptography.

The oldest known application of cryptography was ensuring secret communication between two or more parties. Some forms of ciphers were already used in ancient Egypt. Surprisingly, recommendations to use ciphers can be found also in books like Kamasutra, which lists the art of secret writing as one of the 64 arts that every woman should practise [Kah67]. I leave an explanation of why this should be useful for a woman, to the imagination of the reader. For centuries, designing ciphers had been indeed an art, and only very recently (around 1980), when the heuristic approach was replaced by precise mathematical proofs, the art became a science.

One of the key principles of modern cryptography is the use of formal definitions, which (among other things) require to specify a threat model. The threat model defines the power of an adversary under our consideration. One could hope to build primitives which are secure even using unlimited resources and time. Such adversaries are denoted as unbounded adversaries and primitives secure against such adversaries are referred to as unconditionally secure. It is not always possible to build unconditionally secure primitives and even if this security is feasible for some primitives, often their practicality is questionable. In order to assume more realistic adversaries, so-called computational security has been introduced, in which *efficient* adversaries are considered. The notion of efficiency has been taken from complexity theory, where efficient means algorithms whose runtime is upper bounded by some polynomial in the input length of the given algorithm. At the same time, we usually require that the algorithms of which some cryptographic

primitive consist, should be efficient as well. In other words, this requirement ensures the practical usefulness of the primitive. In this way, the notion of time has found its place in cryptography.

Let's return to ciphers. Modern terminology practically stopped using the word cipher in the late twentieth century and employed a new term - an encryption scheme. In this thesis, we also focus on building encryption schemes. There are two standard notions for an encryption scheme: private-key encryption and public-key encryption. The task of these schemes is to allow two parties to communicate secretly and hence to ensure confidentiality. The crucial difference between the two is when communicating using private-key encryption, both parties have to share the secret key in advance. On the other hand, a public-key encryption scheme makes use of two types of keys: a public key and a secret key. As the names suggest, the public key is made public and allows any party to encrypt messages which can be decrypted only using the corresponding secret key. Security for these two primitives roughly says that without the knowledge of the secret key it is impossible to obtain any information about encrypted messages. This is exactly the point where the encryption schemes considered in this work deviates from the standard notion.

Instead of using a secret key for decryption, we would like ciphertexts to be decryptable by anyone at some specified point in time in the future. The purpose and usefulness of such an encryption scheme might be at the first sight not obvious, however, there are many applications where such functionality is desirable:

1. Often there is a legal obligation that some data must be publicly available at the specified point in time. This is for instance the case for results of elections or census data.

2. An interesting application is the so-called responsible disclosure of security flaws. Security flaws have been repeatedly discovered in different applications and internet protocols [ASS+16, BSY17, PDM+18]. It is a good practice to give vendors some time to fix founded flaws and at the same time making them public is an important measure of how to avoid similar flaws in the future.

One might argue, that desired properties in the above-mentioned applications are easily achievable for example by publishing the information on a given date on your favorite website. But the true advantage of using the encryption scheme is that we do not require that an entity, which intended to send the message in the future, must be present at decryption time.

3. Ability to automatically reveal messages at a specific time in the future allows providing an unbiased lottery. One encrypts the results of a lottery

and makes ciphertext public before the betting phase starts. In this way, the results of the lottery can not be biased.

4. Another application is to deploy this type of encryption scheme in different internet protocols where several possible dishonest parties are involved. As it has been observed in [RSW96, BN00], this enables us to guarantee that either all parties learn an output of the protocol execution or no one does. This property is called *fairness* and it is not easily achievable otherwise.

Currently, there are three main approaches that have been proposed in order to enable "sending information into the future" - Timed-Release Crypto/Encryption (TRE), Time-Lock Puzzles (TLPs), and Time-Lock Encryption (TLE).

Timed-Release Crypto/Encryption. The problem of sending messages into the future was firstly discussed by May [May93] who has introduced the notion of timed-release crypto and proposed a solution based on trusted agents. The suggested idea makes use of a standard private or public-key encryption scheme in order to encrypt messages. These ciphertexts are then published and hence accessible to everyone. Corresponding secret keys are handed to so-called trusted agents that release them after some pre-determined amount of time has passed. This idea has been further developed in a plethora of other works [DOR99, CLQ05, CHKO06, CHS07, CY08] which have started to use the term timed-release encryption to denote this approach based on a trusted entity.

Time-Lock Puzzles. A few years later after introducing the idea of timed-release crypto, Rivest *et al.* [RSW96] have described the broad applicability of timed-release cryptography and at the same time have proposed a new solution using time-lock puzzles. A time-lock puzzle (TLP) allows sealing messages in such a way that one is able to obtain the original message only by executing an expensive sequential computation which takes a pre-determined amount of time even on a parallel computer. The construction of Rivest *et al.* [RSW96] is based on a sequential squaring in an RSA group and has a property that creating a puzzle is much faster than solving the given puzzle. Finding such puzzles seems to be rather difficult. Bitansky *et al.* [BGJ$^+$15] built time-lock puzzles based on randomized encodings, however, this construction is not practical. The possibility of constructing time-lock puzzles in the random oracle model was studied by Mahmoody *et al.* [MMV11] who ruled out black-box constructions of TLPs from one-way permutations and collision-resistant hash functions.

Time-Lock Encryption. To overcome the limitations of the previous two approaches, concretely the need for a trusted entity or execution of an expensive

computation, Liu *et al.* [LJKW18] came up with the idea of reusing an existing expensive computation in combination with an appropriate type of an encryption scheme. Specifically, they have shown that time-lock encryption (TLE) can be directly obtained from an Extractable Witness Encryption (EWE) scheme by defining a suitable relation. Hence, we can view TLE as a special case of the more general primitive that is extractable witness encryption, which has a plethora of other applications outside of timed cryptography. As an example of an existing computation, which provides a good estimation of time, the mining of blocks in the Bitcoin blockchain is considered. Although this solution is the most preferable, the practical constructions of (extractable) witness encryption for the above-mentioned type of relation are not known yet, and hence, it can not be used in practice at the current time.

We provide a summary of the properties that are achievable by the above-mentioned primitives in Figure 1.1. We are interested in the following aspects: if they require a trusted setup, interaction at decryption time, an expensive computation executed by a receiver, or use of heavy cryptographic primitives which are not practical yet. We indicate by a bullet point that a timed primitive currently has a given property.

	TRE	TLP	TLE
Trusted setup	•		
Interaction at decryption time	•		
Expensive computation		•	
Heavy crypto primitives			•

Figure 1.1: Comparison of timed primitives

Overview of the thesis. This thesis only focuses on the latter two of the above-mentioned approaches. We aim to improve the efficiency of both TLPs and TLE and hence broaden their applicability. Even though we use different techniques and building blocks while pursuing our goal, in both cases we apply the same approach. From the broader perspective, one could describe our approach as follows: we allow in both primitives for a trusted setup and try to optimize the remaining algorithms of the given primitive. The thesis consists of three chapters:

Chapter 2 contains preliminaries where we define the notation, cryptographic primitives, and assumptions which are used in the thesis.

Chapter 3 is dedicated to a variant of witness encryption. We have already mentioned that the limiting factor of time-lock encryption with respect to its

practicality is that there is currently only one known construction of this primitive and the proposed construction is based on witness encryption: this primitive is currently only achievable using impractical techniques. We focus on a variant of witness encryption introduced by Abusalah *et al.* [AFP16] which allows for a trusted setup. Such witness encryption with a trusted setup is called *offline* witness encryption (OWE). We propose an extractable variant of OWE (EOWE). One can observe that EOWE for an NP relation defined in the same way as in [LJKW18], directly yields time-lock encryption with trusted setup. We denote this type of TLE by *offline* time-lock encryption. Therefore we focus on building extractable offline witness encryption which has broader applicability than TLE and along the way we obtain offline time-lock encryption for free. The novelty of our construction is a generic technique that yields both OWE and EOWE achieving stronger security guarantees and smaller ciphertext size than the original proposal of Abusalah *et al.* [AFP16]. This is achieved by relying on a new primitive denoted as one-time puncturable tag-based encryption which might have further interesting applications. Moreover, we show an efficient instantiation of our one-time puncturable tag-based encryption scheme.

Chapter 4 discusses encryption schemes that are based on time-lock puzzles. As noted by Malavolta *et al.* [MT19] the practicality of time-lock puzzles is limited by the fact that most applications require several messages to be encrypted independently and solving all puzzles is usually infeasible. To mitigate this issue, they allow a trusted setup that is designed in such a way that homomorphic evaluations on puzzles can be performed. Hence, instead of solving all puzzles independently, one can at first homomorphically combine puzzles and after that solve possibly only one puzzle. This is sufficient for many applications. In this dissertation, we focus on a new approach that also relies on a trusted setup but in contrast to [MT19] we are able to decrypt many ciphertexts by solving one puzzle only. To denote this novel approach we use the term timed-release encryption and we show how we can build this primitive generically from any inherently sequential problem. Moreover, we introduce a new type of time-lock puzzle, called a sequential time-lock puzzle (sTLP). sTLP produces a set of puzzles for different hardness parameters at once in a way that it is sufficient to execute only one sequential computation in order to solve all puzzles. We use this primitive as a building block for sequential timed-release encryption which enjoys the properties of sequential time-lock puzzles and basic timed-release encryption. To instantiate an sTLP, we propose a new assumption whose hardness can be reduced to factoring in the Strong Algebraic Group Model. The last section of the chapter is dedicated to Timed Public-Key Encryption

(TPKE) which has been recently proposed by Katz *et al.* [KLX20]. We examine the relation between TRE and TPKE and propose two constructions of TPKE which are more efficient (in the sense of runtime of the encryption algorithm) than the state-of-the-art construction.

1.1 Publication Overview

The thesis is based on the following research papers.

1. **Offline Witness Encryption with Semi-Adaptive Security.** Peter Chvojka, Tibor Jager, and Saqib A. Kakvi. ACNS 20: 18th International Conference on Applied Cryptography and Network Security.

2. **Constructions of Incremental and Homomorphic Timed-Release Encryption.** Peter Chvojka, Tibor Jager, Daniel Slamanig and Christoph Striecks. Cryptology ePrint Archive, Report 2020/739. http://eprint.iacr.org/2020/739. To be submitted in spring 2021.

3. **Non-Malleable Timed Commitments** (working title). Peter Chvojka and Tibor Jager. To be submitted in spring 2021.

2 Preliminaries

2.1 Notation

We denote our security parameter as λ. For all $n \in \mathbb{N}$, we denote by 1^n the n-bit string of all ones and by $|x|$ we denote the length of the bit string x. We use $x \xleftarrow{\$} S$ to indicate that we choose x uniformly at random from a set S. We use $\mathsf{Funs}[\mathcal{X}, \mathcal{Y}]$ to denote the set of all functions from $\mathcal{X}$ to $\mathcal{Y}$. We use the notation $[n]$ to denote the set $\{1, \ldots, n\}$. For a set $\{a_1, \ldots, a_n\}$ we use notation $(a_i)_{i\in[n]}$ and in similar way we use this notation also for sets of tuples. All algorithms may be randomized. For any probabilistic polynomial-time (PPT) algorithm A, we define $x \leftarrow A(1^\lambda, (a_i)_{i\in[n]})$ as the execution of A with inputs security parameter 1^λ, $a_1, \ldots, a_n$ and fresh randomness and then assigning the output to x. Sometimes we let an algorithm run using explicit random coins r and in that case we write $x \leftarrow A(1^\lambda, (a_i)_{i\in[n]}; r)$. We write $(x_i \leftarrow A(a_i))_{i\in[n]}$ to denote running n times the algorithm A with fresh randomness on inputs $a_1, \ldots, a_n$ and assigning the output to $x_1, \ldots, x_n$. We denote by $\max(a, b)$ the function which returns the maximum element of two comparable elements a and b. We say that an algorithm is *efficient* if it is a PPT algorithm. We use $\mathsf{poly}(\cdot)$ to denote some polynomial and $\mathsf{polylog}(\cdot)$ to denote a polylogarithmic function. All adversaries are non-uniform which is usually denoted as $\mathcal{A} = \{\mathcal{A}_\lambda\}_{\lambda\in\mathbb{N}}$. In security definitions, we assume that $\mathcal{A}$ keeps a state between invocations.

In Chapter 4, we want to also capture the parallel complexity of certain computations. Therefore we use circuits to model parallel algorithms where the parallel time of an algorithm is determined by the depth of the corresponding circuit. The total running time of an algorithm is determined by the total size of the corresponding circuit and bounds the number of processors.

2.2 NP Language

Definition 2.2.1. A language $L \subseteq \{0,1\}^*$ is in $\mathtt{NP}$ if there exists a polynomial $\mathsf{poly} : \mathbb{N} \to \mathbb{N}$ and polynomial-time algorithm V (called the verifier for L) such that for every $x \in \{0,1\}^*$,

$$x \in L \iff \exists w \in \{0,1\}^{\mathsf{poly}(|x|)} \text{ s.t. } V(x, w) = 1.$$

If $x \in L$ and $w \in \{0,1\}^{\mathsf{poly}(|x|)}$ satisfy $V(x,w) = 1$, then we call w a witness for x.

2.3 Cryptographic Primitives

2.3.1 One-Time Signatures

A digital signature scheme allows one to "sign" a message in such a way that the recipient of the message is able to verify its authenticity (the fact that the message comes from a claimed sender) and integrity (the message has not been modified during a transmission). In order to sign a message, a signer has to at first generate a public verification key and a secret key. The secret key is used for signing and the public verification key is used by other parties to verify signatures. In our applications, we require rather weak security for a digital signature scheme, namely security against adversaries that are able to see a signature on one message of their choice. Such signature schemes are called one-time signatures because they are used to issue only one signature per public verification key.

Definition 2.3.1. A *one-time signature scheme* $\mathtt{OTS}$ with message space $\mathcal{M}$ consists of three efficient algorithms $\mathtt{OTS} = (\mathsf{Gen}, \mathsf{Sign}, \mathsf{Vrfy})$ such that:

- $(\mathsf{vk}_{\mathsf{OT}}, \mathsf{sk}_{\mathsf{OT}}) \leftarrow \mathsf{Gen}(1^\lambda)$ is a probabilistic algorithm that takes as input the security parameter 1^λ and outputs a public verification key $\mathsf{vk}_{\mathsf{OT}}$ and a signing key $\mathsf{sk}_{\mathsf{OT}}$.
- $\sigma \leftarrow \mathsf{Sign}(\mathsf{sk}_{\mathsf{OT}}, m)$ is a probabilistic algorithm that takes as input a signing key $\mathsf{sk}_{\mathsf{OT}}$ and a message m and outputs a signature σ.
- $m \leftarrow \mathsf{Vrfy}(\mathsf{vk}_{\mathsf{OT}}, m, \sigma)$ is a deterministic algorithm that takes as input a verification key $\mathsf{vk}_{\mathsf{OT}}$, a message m, and a signature σ and outputs either 0 (reject) or 1 (accept).

We say $\mathtt{OTS}$ is *correct* if for all $\lambda \in \mathbb{N}$, all $m \in \mathcal{M}$, it holds that

$$\Pr[\mathsf{Vrfy}(\mathsf{vk}_{\mathsf{OT}}, m, \mathsf{Sign}(\mathsf{sk}_{\mathsf{OT}}, m)) = 1 : (\mathsf{vk}_{\mathsf{OT}}, \mathsf{sk}_{\mathsf{OT}}) \leftarrow \mathsf{Gen}(1^\lambda)] = 1.$$

Definition 2.3.2. A OTS scheme $\mathtt{OTS}$ is a *strong one-time signature scheme* if for all non-uniform PPT adversaries $\mathcal{F} = \{\mathcal{F}_\lambda\}_{\lambda \in \mathbb{N}}$ there is a negligible function $\mathsf{negl}(\cdot)$ such that for all $\lambda \in \mathbb{N}$

$$\mathbf{Adv}^{\mathtt{OTS}}_{\mathcal{F}} = \Pr\left[\begin{array}{c|c} \mathsf{Vrfy}(\mathsf{vk}_{\mathsf{OT}}, m, \sigma) = 1 & (\mathsf{vk}_{\mathsf{OT}}, \mathsf{sk}_{\mathsf{OT}}) \leftarrow \mathsf{Gen}(1^\lambda) \\ \wedge (m, \sigma) \neq (m', \sigma') & (m, \sigma) \leftarrow \mathcal{F}_\lambda^{\mathsf{SIGN}(\cdot)}(\mathsf{vk}_{\mathsf{OT}}) \end{array}\right] \leq \mathsf{negl}(\lambda),$$

where $\mathsf{SIGN}(\cdot)$ is a single query oracle which on input m' returns $\sigma' \leftarrow \mathsf{Sign}(\mathsf{sk}_{\mathsf{OT}}, m')$.

2.3.2 Public-Key Encryption

A public-key encryption (PKE) scheme allows private communication among parties without a need to share any secret information at an earlier point in time. One party (the receiver) generates a so-called public key/secret key pair and makes the public key accessible to everyone. Other parties can use the public key to encrypt messages for the receiver. The resulting ciphertexts are decryptable only using the corresponding secret key. Under the assumption that the receiver has kept the secret key for itself, only the receiver and the party, which has sent the ciphertext, know its content.

Definition 2.3.3. A *public-key encryption scheme* $\mathtt{PKE}$ with message space $\mathcal{M}$ consists of three efficient algorithms $\mathtt{PKE} = (\mathsf{Gen}, \mathsf{Enc}, \mathsf{Dec})$ such that:

- $(\mathsf{pk}, \mathsf{sk}) \leftarrow \mathsf{Gen}(1^\lambda)$ is a probabilistic algorithm that takes as input the security parameter 1^λ and outputs a public/secret key pair.
- $c \leftarrow \mathsf{Enc}(\mathsf{pk}, m)$ is a probabilistic algorithm that takes as input a public key pk and a message m, and outputs a ciphertext c.
- $m \leftarrow \mathsf{Dec}(\mathsf{sk}, c)$ is a deterministic algorithm that takes as input a secret key sk and a ciphertext c and outputs $m \in \mathcal{M} \cup \{\perp\}$.

We say $\mathtt{PKE}$ is *correct* if for all $\lambda \in \mathbb{N}$ and all $m \in \mathcal{M}$, it holds that

$$\Pr[\mathsf{Dec}(\mathsf{sk}, \mathsf{Enc}(\mathsf{pk}, m)) = m : (\mathsf{pk}, \mathsf{sk}) \leftarrow \mathsf{Gen}(1^\lambda)] = 1.$$

We consider two standard security notions for public-key encryption, concretely security against a chosen plaintext attack (CPA) and chosen ciphertext attack (CCA).

Definition 2.3.4. A PKE scheme $\mathtt{PKE}$ is *IND-CPA secure* if for all non-uniform PPT adversaries $\mathcal{A} = \{\mathcal{A}_\lambda\}_{\lambda \in \mathbb{N}}$ there is a negligible function $\mathsf{negl}(\cdot)$ such that for all $\lambda \in \mathbb{N}$

$$\mathbf{Adv}_{\mathcal{A}}^{\mathtt{PKE},\mathsf{CPA}} = \left| \Pr \left[b = b' \ : \ \begin{array}{r} (\mathsf{pk}, \mathsf{sk}) \leftarrow \mathsf{Gen}(1^\lambda) \\ (m_0, m_1) \leftarrow \mathcal{A}_\lambda(\mathsf{pk}) \\ b \xleftarrow{\$} \{0,1\}; c \leftarrow \mathsf{Enc}(\mathsf{pk}, m_b) \\ b' \leftarrow \mathcal{A}_\lambda(c) \end{array} \right] - \frac{1}{2} \right| \leq \mathsf{negl}(\lambda),$$

where we require that $|m_0| = |m_1|$.

We state also an alternative definition of IND-CPA-secure public-key encryption, where an adversary submits only one message to the challenger and is given either encryption of the provided message (real message) or encryption of a random message. This variant is often denoted as real or random CPA (RoR-CPA) security and it is equivalent to Definition 2.3.4. We use this definition later in some of our proofs, which allows us to make an overall proof simpler.

$\underline{\mathsf{ExpRoR}^b_{\mathcal{A}}(\lambda)\text{:}}$
$(\mathsf{pk}, \mathsf{sk}) \leftarrow \mathsf{Gen}(1^\lambda)$
$m_0 \leftarrow \mathcal{A}_\lambda(\mathsf{pk})$
$m_1 \xleftarrow{\$} \mathcal{M}, \text{s.t. } |m_0| = |m_1|$
$c \leftarrow \mathsf{Enc}(\mathsf{pk}, m_b)$
return $b' \leftarrow \mathcal{A}_\lambda(c)$

Figure 2.1: Real or Random security experiment for PKE.

Definition 2.3.5. Consider the security experiment $\mathsf{ExpRoR}^b_{\mathcal{A}}(\lambda)$ in Figure 2.1. A PKE scheme PKE is *RoR-CPA secure* if for all non-uniform PPT adversaries $\mathcal{A} = \{\mathcal{A}_\lambda\}_{\lambda\in\mathbb{N}}$ there is a negligible function $\mathsf{negl}(\cdot)$ such that for all $\lambda \in \mathbb{N}$

$$\mathbf{Adv}^{\mathtt{PKE},\mathsf{RoR}}_{\mathcal{A}} = \left|\Pr[\mathsf{ExpRoR}^0_{\mathcal{A}}(\lambda) = 1] - \Pr[\mathsf{ExpRoR}^1_{\mathcal{A}}(\lambda) = 1]\right| \leq \mathsf{negl}(\lambda).$$

Definition 2.3.6. A PKE scheme PKE is *IND-CCA secure* if for all non-uniform PPT adversaries $\mathcal{A} = \{\mathcal{A}_\lambda\}_{\lambda\in\mathbb{N}}$ there is a negligible function $\mathsf{negl}(\cdot)$ such that for all $\lambda \in \mathbb{N}$

$$\mathbf{Adv}^{\mathtt{PKE},\mathsf{CCA}}_{\mathcal{A}} = \left|\Pr\left[b = b' \ : \begin{array}{r}(\mathsf{pk}, \mathsf{sk}) \leftarrow \mathsf{Gen}(1^\lambda)\\ (m_0, m_1) \leftarrow \mathcal{A}^{\mathsf{DEC}(\cdot)}_\lambda(\mathsf{pk})\\ b \xleftarrow{\$} \{0,1\};\ c^* \leftarrow \mathsf{Enc}(\mathsf{pk}, m_b)\\ b' \leftarrow \mathcal{A}^{\mathsf{DEC}(\cdot)}_\lambda(c^*)\end{array}\right] - \frac{1}{2}\right| \leq \mathsf{negl}(\lambda),$$

where the oracle $\mathsf{DEC}(c)$ returns $m \leftarrow \mathsf{Dec}(\mathsf{sk}, c)$ with the restriction that after giving adversary $\mathcal{A}_\lambda$ the challenge ciphertext c^*, $\mathcal{A}_\lambda$ is not allowed to query the oracle $\mathsf{DEC}(\cdot)$ for the challenge ciphertext c^*. We require that $|m_0| = |m_1|$.

2.3.3 Homomorphic Encryption

An extension of public-key encryption is homomorphic encryption (HE) which allows to evaluate some circuit C over a tuple of ciphertexts $(c_j)_{j\in[\ell]}$. The result of this evaluation is a ciphertext whose decryption is the same as evaluating the circuit on inputs $(m_j)_{j\in[\ell]}$ where m_j is decryption of c_j for all $j \in [\ell]$.

Definition 2.3.7. A *homomorphic encryption scheme* HE for a circuit class $\mathcal{C} = \{\mathcal{C}_\lambda\}_{\lambda\in\mathbb{N}}$ is a PKE scheme with one additional algorithm Eval defined as follows:

- $c \leftarrow \mathsf{Eval}(\mathsf{pk}, C, (c_j)_{j\in[\ell]})$ is a probabilistic algorithm that takes as input a public key pk, a circuit $C \in \mathcal{C}_\lambda$, and a set of ciphertexts $(c_j)_{j\in[\ell]}$ and outputs an evaluated ciphertext c.

A HE scheme is *correct*, if for all $\lambda \in \mathbb{N}$, all circuits $C \in \mathcal{C}_\lambda$, all inputs $(m_1, \ldots, m_\ell)$, all $(\mathsf{pk}, \mathsf{sk})$ in the support of $\mathsf{Gen}(1^\lambda)$, and all c_j in the support of $\mathsf{Enc}(\mathsf{pk}, m_j)$ it holds that

$$\Pr[\mathsf{Dec}(\mathsf{sk}, \mathsf{Eval}(\mathsf{pk}, C, (c_j)_{j\in[\ell]}) = C((m_j)_{j\in[\ell]})] = 1.$$

A HE scheme is *secure* if it is an IND-CPA-secure PKE scheme in the sense of Definition 2.3.4. We say that HE is *compact* if there exists a polynomial $\mathsf{poly}(\cdot)$ such that for all $\lambda \in \mathbb{N}$, all circuits $C \in \mathcal{C}_\lambda$, all inputs $(m_1, \ldots, m_\ell)$, all $(\mathsf{pk}, \mathsf{sk})$ in the support of $\mathsf{Gen}(1^\lambda)$, and all c_j in the support of $\mathsf{Enc}(\mathsf{pp}_e, m_j)$ it holds that

$$\left|\mathsf{Eval}(\mathsf{pk}, C, (c_j)_{j\in[\ell]})\right| = \mathsf{poly}(\lambda, |C((m_j)_{j\in[\ell]})|).$$

A scheme HE which is homomorphic for all polynomial-sized circuits is called fully homomorphic (FHE).

2.3.4 Tag-Based Encryption

The notion of tag-based encryption (TBE) has been introduced by MacKenzie, Reiter and Yang [MRY04], who have shown that a tag-based encryption scheme can be lifted to an IND-CCA-secure public-key encryption scheme via the so-called CHK transformation [CHK04]. Similarly to a public-key encryption scheme, a tag-based encryption scheme consists of a tuple of algorithms $(\mathsf{Gen}, \mathsf{Enc}, \mathsf{Dec})$, but encryption and decryption algorithm additionally take as input some tag t.

Definition 2.3.8. A *tag-based encryption scheme* TBE with message space $\mathcal{M}$ consists of three efficient algorithms $\mathtt{TBE} = (\mathsf{Gen}, \mathsf{Enc}, \mathsf{Dec})$ such that:

- $(\mathsf{pk}, \mathsf{sk}) \leftarrow \mathsf{Gen}(1^\lambda)$ is a probabilistic algorithm that takes as input the security parameter 1^λ and outputs a public/secret key pair.
- $c \leftarrow \mathsf{Enc}(\mathsf{pk}, m, t)$ is a probabilistic algorithm that takes as input a public key pk, a message m and a tag t, and outputs a ciphertext c.
- $m \leftarrow \mathsf{Dec}(\mathsf{sk}, t, c)$ is a deterministic algorithm that takes as input a secret key sk, a tag t, and a ciphertext c and outputs $m \in \mathcal{M} \cup \{\perp\}$.

We say TBE is *correct* if for all $\lambda \in \mathbb{N}$, all $m \in \mathcal{M}$ and all tags t, it holds that

$$\Pr[\mathsf{Dec}(\mathsf{sk}, t, \mathsf{Enc}(\mathsf{pk}, m, t)) = m : (\mathsf{pk}, \mathsf{sk}) \leftarrow \mathsf{Gen}(1^\lambda)] = 1.$$

For the security of a TBE scheme, we require a weak form of security which has been introduced by Kiltz [Kil06], who has shown that this security notion is sufficient for the CHK transformation in order to obtain an IND-CCA-secure public-key encryption scheme.

Definition 2.3.9. A TBE scheme TBE is *selective-tag weakly secure against chosen ciphertext attacks* if for all non-uniform PPT adversaries $\mathcal{A} = \{\mathcal{A}_\lambda\}_{\lambda\in\mathbb{N}}$ there is a negligible function $\mathsf{negl}(\cdot)$ such that for all $\lambda \in \mathbb{N}$

$$\mathbf{Adv}^{\mathtt{TBE}}_{\mathcal{A}} = \left| \Pr\left[b = b' \ : \begin{array}{r} t^* \leftarrow \mathcal{A}_\lambda \\ (\mathsf{pk}, \mathsf{sk}) \leftarrow \mathsf{Gen}(1^\lambda) \\ (m_0, m_1) \leftarrow \mathcal{A}_\lambda^{\mathsf{DEC}(\cdot,\cdot)}(\mathsf{pk}) \\ b \xleftarrow{\$} \{0,1\};\ c \leftarrow \mathsf{Enc}(\mathsf{pk}, m_b, t^*) \\ b' \leftarrow \mathcal{A}_\lambda^{\mathsf{DEC}(\cdot,\cdot)}(c) \end{array} \right] - \frac{1}{2} \right| \leq \mathsf{negl}(\lambda),$$

where the oracle $\mathsf{DEC}(c, t)$ returns $\mathsf{Dec}(\mathsf{sk}, t, c)$ with the restriction that $\mathcal{A}_\lambda$ is not allowed to query the oracle DEC for the target tag t^* and we require that $|m_0| = |m_1|$.

2.3.5 Functional Encryption

A functional encryption (FE) scheme [BSW11, O'N10] is a generalization of public-key encryption which allows to generate a secret key with respect to some function f. Decrypting a ciphertext using the given secret key leads to the value $f(m)$, where m is the message encrypted in the ciphertext, and security of FE guarantees that no other information about m is leaked.

Definition 2.3.10. A *functional encryption scheme* FE for a class of functions $\mathcal{F} = \{\mathcal{F}_\lambda\}_{\lambda\in\mathbb{N}}$ consists of four efficient algorithms $(\mathsf{Gen}, \mathsf{KeyGen}, \mathsf{Enc}, \mathsf{Dec})$. Let $\mathcal{X}_\lambda$ be the input space of $\mathcal{F}_\lambda$ and let $\mathcal{Y}_\lambda$ be the output space of $\mathcal{F}_\lambda$.

- $(\mathsf{pk}, \mathsf{msk}) \leftarrow \mathsf{Gen}(1^\lambda, \mathcal{F})$ is a probabilistic algorithm that takes as input the security parameter 1^λ and a class of functions $\mathcal{F}$ and outputs a public key pk and a master secret key msk.
- $\mathsf{sk}_f \leftarrow \mathsf{KeyGen}(\mathsf{msk}, f)$ is a probabilistic algorithm that takes as input a master secret key msk and a function $f \in \mathcal{F}_\lambda$ and outputs a secret key sk_f for f.

- $c \leftarrow \mathsf{Enc}(\mathsf{pk}, x)$ is a probabilistic algorithm that takes as input a public key pk and a message $x \in \mathcal{X}_\lambda$ and outputs a ciphertext c.
- $f(x) \leftarrow \mathsf{Dec}(\mathsf{sk}_f, c)$ is a deterministic algorithm that takes as input a secret key sk_f and a ciphertext c and outputs $f(x) \in \mathcal{Y}_\lambda \cup \{\perp\}$.

We say FE is *correct* if for all $\lambda \in \mathbb{N}$, for all $f : \mathcal{X}_\lambda \to \mathcal{Y}_\lambda \in \mathcal{F}_\lambda$, for all $x \in \mathcal{X}_\lambda$ it holds that

$$\Pr\left[\mathsf{Dec}(\mathsf{sk}_f, c) = f(x) \;:\; \begin{array}{r}(\mathsf{pk}, \mathsf{msk}) \leftarrow \mathsf{Gen}(1^\lambda, \mathcal{F}) \\ \mathsf{sk}_f \leftarrow \mathsf{KeyGen}(\mathsf{msk}, f) \\ c \leftarrow \mathsf{Enc}(\mathsf{pk}, x)\end{array}\right] = 1.$$

Definition 2.3.11. An FE scheme FE is *IND-CPA secure* if for all non-uniform PPT adversaries $\mathcal{A} = \{\mathcal{A}_\lambda\}_{\lambda \in \mathbb{N}}$ there is a negligible function $\mathsf{negl}(\cdot)$ such that for all $\lambda \in \mathbb{N}$

$$\mathbf{Adv}^{\mathrm{FE}}_{\mathcal{A}} = \left|\Pr\left[b = b' \;:\; \begin{array}{r}(\mathsf{pk}, \mathsf{msk}) \leftarrow \mathsf{Gen}(1^\lambda, \mathcal{F}) \\ (x_0, x_1) \leftarrow \mathcal{A}_\lambda^{\mathsf{KEYGEN}(\cdot)}(\mathsf{pk}) \\ b \leftarrow \{0,1\}, c \leftarrow \mathsf{Enc}(\mathsf{pk}, x_b) \\ b' \leftarrow \mathcal{A}_\lambda^{\mathsf{KEYGEN}(\cdot)}(c)\end{array}\right] - \frac{1}{2}\right| \leq \mathsf{negl}(\lambda),$$

where the oracle $\mathsf{KEYGEN}(f)$ returns $\mathsf{KeyGen}(\mathsf{msk}, f)$ with the restriction that $\mathcal{A}_\lambda$ only queries the oracle with functions f such that $f(x_0) = f(x_1)$.

Example: Identity-Based Encryption. A generalization of PKE is identity-based encryption (IBE) [BF01], where the public key, also called identity, can be an arbitrary string and in order to decrypt a ciphertext a secret key for the corresponding identity is required. An IBE scheme can be obtained from an FE scheme as in Definition 2.3.10 by setting the message space $\mathcal{X} := \mathcal{ID} \times \mathcal{M}$ representing pairs of identities and messages (id, m) and $\mathcal{F}$ being an equality testing functionality. A secret key $\mathsf{sk}_{f_{id^*}}$ for identity id^* is generated with respect to f_{id^*} defined as:

$$f_{id^*}((id, m)) = \begin{cases} m \text{ if } id = id^*, \\ \perp \text{ otherwise.} \end{cases}$$

2.3.6 Obfuscation

In the past years, great attention has been dedicated to the area of obfuscation, which has a plethora of exciting applications. Obfuscation is usually defined for a class of circuits $\mathcal{C}_\lambda$, which consists of circuits of size bounded by $\mathsf{poly}(\lambda)$. An

obfuscator takes as input any circuit from the class and outputs another circuit (obfuscated circuit) with the same input/output behaviour as the original circuit, however, the obfuscated circuit should be "unintelligible" in some sense. Several variants of obfuscations have been proposed, though some of them are unachievable [BGI+01]. In our applications, we rely on two weaker variants of obfuscation namely indistinguishability and extractability obfuscation.

Indistinguishability obfuscation ($i\mathcal{O}$) has been defined by Barak *et al.* [BGI+01] and its purpose is to hide implementation details of a circuit. This intuition is formalized in the following way:

Definition 2.3.12. A uniform PPT machine $i\mathcal{O}$ is called an *indistinguishability obfuscator* for a circuit class $\mathcal{C}_\lambda$ if the following conditions are met:

- *Preserving Functionality:* For all $\lambda \in \mathbb{N}, C \in \mathcal{C}_\lambda$ and $x \in \{0,1\}^\lambda$, we have

$$\Pr[C(x) = \widetilde{C}(x) : \widetilde{C} \leftarrow i\mathcal{O}(1^\lambda, C)] = 1.$$

- *Indistinguishability:* For all non-uniform PPT distinguishers $\mathcal{D} = \{\mathcal{D}_\lambda\}_{\lambda\in\mathbb{N}}$ there exists a neglible function $\mathsf{negl}(\cdot)$ such that, for all $C_0, C_1 \in \mathcal{C}_\lambda$ with $C_0(x) = C_1(x)$ for all inputs $x \in \{0,1\}^\lambda$, the following holds:

$$\mathbf{Adv}^{i\mathcal{O}}_{\mathcal{D}} = \Pr[\mathcal{D}_\lambda(i\mathcal{O}(1^\lambda, C_0)) = 1] - \Pr[\mathcal{D}_\lambda(i\mathcal{O}(1^\lambda, C_1)) = 1] \leq \mathsf{negl}(\lambda).$$

Another variant of obfuscation that we consider, is extractability obfuscation ($e\mathcal{O}$) also sometimes called differing input obfuscation. In the security definition of extractability obfuscation, we allow circuits to differ in their input/output behaviour. Whenever an adversary is able to distinguish between obfuscations of two circuits, we require that it must know the corresponding input on which the given circuits produce different outputs. We define extractability obfuscation according to Boyle, Chung and Pass [BCP14] and we consider one of their variants, specifically, we consider $e\mathcal{O}$ with distributional auxiliary input.

Definition 2.3.13. A uniform PPT machine $e\mathcal{O}$ is called an *extractability obfuscator w.r.t. distributional auxiliary input* for a circuit class $\mathcal{C}_\lambda$ if the following conditions are met:

- *Preserving Functionality:* For all $\lambda \in \mathbb{N}, C \in \mathcal{C}_\lambda$ and $x \in \{0,1\}^\lambda$, we have

$$\Pr[C(x) = \widetilde{C}(x) : \widetilde{C} \leftarrow i\mathcal{O}(1^\lambda, C)] = 1.$$

- *Extractability:* for all non-uniform PPT distinguishers $\mathcal{D} = \{\mathcal{D}_\lambda\}_{\lambda\in\mathbb{N}}$, for every polynomial $\mathsf{p}(\lambda)$ and for every efficiently samplable distribution $\mathfrak{D}$ over $\mathcal{C}_\lambda \times \mathcal{C}_\lambda \times \{0,1\}^*$ there exist a non-uniform PPT extractor $\mathcal{E} = \{\mathcal{E}_\lambda\}_{\lambda\in\mathbb{N}}$ and

polynomial $\mathsf{q}(\lambda)$ such that for every $\lambda \in \mathbb{N}$, it holds with overwhelming probability over $(C_0, C_1, \mathsf{aux}) \in_{\mathfrak{D}} \mathcal{C}_\lambda \times \mathcal{C}_\lambda \times \{0,1\}^*$ that

$$\left|\Pr[\mathcal{D}_\lambda(e\mathcal{O}(1^\lambda, C_0), C_0, C_1, \mathsf{aux}) = 1] - \Pr[\mathcal{D}_\lambda(e\mathcal{O}(1^\lambda, C_1), C_0, C_1, \mathsf{aux}) = 1]\right| \geq \frac{1}{\mathsf{p}(\lambda)} \implies \Pr[C_0(x) \neq C_1(x) : x \leftarrow \mathcal{E}_\lambda(C_0, C_1, \mathsf{aux})] \geq \frac{1}{\mathsf{q}(\lambda)}.$$

2.3.7 Randomized Encoding

A randomized encoding [IK00] scheme allows for encoding a "complex" computation specified by a function f and input x, using a representation $\widehat{f}(x)$ which is simpler to compute. The security essentially guarantees that the distribution of randomized encoding depends only on the encoded value $f(x)$ and does not reveal further information about x.

Definition 2.3.14 (Randomized Encoding [BGJ+16])**.** A *randomized encoding scheme* $\mathtt{RE}$ consists of two algorithms $\mathtt{RE} = (\mathsf{Encode}, \mathsf{Decode})$ satisfying the following requirements:

- $\widehat{M}(x) \leftarrow \mathsf{Encode}(M, x, T, 1^\lambda)$ is a probabilistic algorithm that takes as input a machine M, input x and time bound T and outputs a randomized encoding $\widehat{M}(x)$. Encode can be computed by a uniform circuit of depth $\mathsf{polylog}(T) \cdot \mathsf{poly}(|M|, |x|, \lambda)$ and total size $T \cdot \mathsf{poly}(|M|, \lambda)$.
- $y \leftarrow \mathsf{Decode}(\widehat{M}(x))$ is a deterministic algorithm that takes as input a randomized encoding $\widehat{M}(x)$ and computes an output $y \in \{0,1\}^\lambda$. Decode can be computed in (sequential) time $T \cdot \mathsf{poly}(|M|, |x|, \lambda)$.

We say that $\mathtt{RE}$ is *correct* if for every input x and machine M such that, on input x, M halts in T steps and produces a λ-bit output, it holds that $y = M(x)$ with overwhelming probability over the coins of Encode.

Definition 2.3.15 (Security [BGJ+16])**.** A randomized encoding scheme $\mathtt{RE}$ is *secure* of there exists a PPT simulator Sim satisfying: for any polynomial-size distinguisher $\mathcal{D} = \{\mathcal{D}_\lambda\}_{\lambda \in \mathbb{N}}$ and polynomials $m(\cdot), n(\cdot), T(\cdot)$, there exist a negligible function $\mathsf{negl}(\cdot)$, such that for all $\lambda \in \mathbb{N}$, machines $M \in \{0,1\}^{m(\lambda)}$, inputs $x \in \{0,1\}^{n(\lambda)}$

$$\Big|\Pr[\mathcal{D}_\lambda(\widehat{M}(x)) = 1 : \widehat{M}(x) \leftarrow \mathsf{Encode}(M, x, T(\lambda), 1^\lambda)] - \Pr[\mathcal{D}_\lambda(\widehat{S}_y) = 1 : \widehat{S}_y \leftarrow \mathsf{Sim}(y, 1^{m(\lambda)}, 1^{n(\lambda)}, T(\lambda), 1^\lambda)]\Big| \leq \mathsf{negl}(\lambda),$$

where y is the output of $M(x)$ after $T(\lambda)$ steps.

Definition 2.3.16 (Succinct Randomized Encoding [BGJ+16])**.** A *succinct randomized encoding* scheme is a randomized encoding scheme $\mathtt{RE} = (\mathsf{Encode}, \mathsf{Decode})$ in the sense of Definition 2.3.14 but it additionally fulfils that $\mathsf{Encode}(M, x, T, 1^\lambda)$ can be computed in (sequential) time $\mathsf{polylog}(T) \cdot \mathsf{poly}(|M|, |x|, \lambda)$.

We say a succinct randomized encoding scheme is secure if it is a secure RE scheme in the sense of Definition 2.3.18.

Definition 2.3.17 (Reusable Randomized Encoding [BGJ+16])**.** A reusable randomized encoding scheme consists of algorithms $\mathtt{RE} = (\mathsf{Preproc}, \mathsf{Encode}, \mathsf{Decode})$ satisfying the following requirements:

- $(\widehat{U}, K) \leftarrow \mathsf{Preproc}(m, n, T, 1^\lambda)$ is a probabilistic algorithm that takes as input bounds m, n, T on machine size, input size, and time, as well as a security parameter 1^λ. It outputs an encoded state $\widehat{U}$ and a short secret key $K \in \{0,1\}^\lambda$. $\mathsf{Preproc}$ can be computed by a uniform circuit of depth $\mathsf{polylog}(T) \cdot \mathsf{poly}(m, n, \lambda)$ and total size $T \cdot \mathsf{poly}(m, \lambda)$.
- $\widehat{M}(x) \leftarrow \mathsf{Encode}(M, x, K)$ is a probabilistic algorithm that takes as input a machine M, input x, a secret key $K \in \{0,1\}^\lambda$ and outputs a randomized encoding $\widehat{M}(x)$. Encode can be computed in sequential time $\mathsf{polylog}(T) \cdot \mathsf{poly}(m, n, \lambda)$.
- $y \leftarrow \mathsf{Decode}(\widehat{U}, \widehat{M}(x))$ is a deterministic algorithm that takes as input an encoded state $\widehat{U}$ and a randomized encoding $\widehat{M}(x)$ and computes an output $y \in \{0,1\}^\lambda$. Decode can be computed in (sequential) time $T \cdot \mathsf{poly}(m, n, \lambda)$.

We say that $\mathtt{RE}$ is *correct* if for every m, n, T, λ, n-bit input x, and m-bit machine M such that $M(x)$ halts in T steps, it holds that $y = M(x)$ with overwhelming probability over the coins of $\mathsf{Preproc}, \mathsf{Encode}$.

Definition 2.3.18 (Security [BGJ+16])**.** A reusable randomized encoding scheme $\mathtt{RE}$ is *secure* if there exists a PPT simulator Sim satisfying: for any polynomial-size distinguisher $\mathcal{D} = \{\mathcal{D}_\lambda\}_{\lambda \in \mathbb{N}}$ and polynomials $m(\cdot), n(\cdot), T(\cdot)$, there exists a negligible function $\mathsf{negl}(\cdot)$ such that for all $\lambda \in \mathbb{N}$, for all machines and inputs $(M_1, x_1), \ldots, (M_k, x_k) \in \{0,1\}^{m(\lambda)+n(\lambda)}$

$$\left| \Pr\left[\mathcal{D}_\lambda(\widehat{U}, (\widehat{M_i}(x_i))_{i\in[k]}) = 1 \ : \ \begin{array}{r} (\widehat{U}, K) \leftarrow \mathsf{Preproc}(m(\lambda), n(\lambda), T(\lambda), 1^\lambda) \\ (\widehat{M_i}(x_i) \leftarrow \mathsf{Encode}(M_i, x_i, K))_{i\in[k]} \end{array} \right] - \right.$$
$$\left. \Pr\left[\mathcal{D}_\lambda(\widehat{U}, (\widehat{S}_{y_i})_{i\in[k]}) = 1 : (\widehat{S}_{y_i})_{i\in[k]} \leftarrow \mathsf{Sim}((y_i)_{i\in[k]}, m(\lambda), n(\lambda), T(\lambda), 1^\lambda)\right] \right| \leq \mathsf{negl}(\lambda),$$

where y_i is the output of $M_i(x_i)$ after $T(\lambda)$ steps.

2.4 Bilinear Groups

Definition 2.4.1. Let $\mathbb{G}_1, \mathbb{G}_2$ and $\mathbb{G}_T$ be cyclic groups of prime order p and $g_1 \in \mathbb{G}_1, g_2 \in \mathbb{G}_2$ be generators of $\mathbb{G}_1$ and $\mathbb{G}_2$, respectively. Then a *bilinear group* is a tuple $(p, \mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T, g_1, g_2, e)$, where $e : \mathbb{G}_1 \times \mathbb{G}_2 \to \mathbb{G}_T$ is an efficiently computable map (also called pairing) fulfilling the following properties:

1. *Bilinearity:* for all $x, x' \in \mathbb{G}_1$ and $y, y' \in \mathbb{G}_2$ holds

$$e(x \cdot x', y) = e(x, y) \cdot e(x'.y) \text{ and } e(x, y \cdot y') = e(x, y) \cdot e(x.y'),$$

2. *Non-degeneracy:* $g_T = e(g_1, g_2)$ is a generator of $\mathbb{G}_T$.

Notice that bilinearity implies the following: for all $a, b \in \mathbb{Z}_p$ holds

$$e(g_1^a, g_2^b) = e(g_1, g_2)^{ab} = e(g_1^b, g_2^a).$$

If $\mathbb{G}_1 = \mathbb{G}_2$, we say that the pairing is symmetric, otherwise we say that the pairing is asymmetric.

2.5 Assumptions

In this section, we define some standard assumptions which are later used as a basis for the cryptographic primitives of our interest. We introduce the Decisional Diffie-Hellman assumption only for purpose of explanation of the decision linear assumption.

Let GrpGen denote a polynomial-time group generation algorithm which on input 1^λ outputs a description of a cyclic group $\mathbb{G}$ of order p, where p is λ-bit prime, together with its generator $g \in \mathbb{G}$.

Definition 2.5.1 (Decisional Diffie-Hellman Assumption (DDH))**.** The *Decisional Diffie-Hellman assumption* holds relative to GrpGen if for all non-uniform PPT adversaries $\mathcal{A} = \{\mathcal{A}_\lambda\}_{\lambda \in \mathbb{N}}$ there exists a negligible function $\mathsf{negl}(\cdot)$ such that for all $\lambda \in \mathbb{N}$

$$\mathbf{Adv}_{\mathcal{A}}^{\mathsf{DDH}} = \left| \Pr \left[b = b' \ : \begin{array}{r} (p, \mathbb{G}, g) \leftarrow \mathsf{GrpGen}(1^\lambda) \\ x, y \xleftarrow{\$} \mathbb{Z}_p, b \xleftarrow{\$} \{0,1\} \\ \text{if } b = 0 : z := x \cdot y \\ \text{if } b = 1 : z \xleftarrow{\$} \mathbb{Z}_p \\ b' \leftarrow \mathcal{A}_\lambda(p, \mathbb{G}, g, g^x, g^y, g^z) \end{array} \right] - \frac{1}{2} \right| \leq \mathsf{negl}(\lambda).$$

We now define the decision linear assumption [BBS04] in a so-called *gap group* [OP01]. A group generation algorithm $\mathsf{GrpGen}(1^\lambda)$ of a gap group on input 1^λ, outputs additionally to the description of cyclic group $\mathbb{G}$ of order p and a generator g also the description of a Diffie-Hellman verification function DDHvf. DDHvf takes as input a tuple $(g_1, g_1^x, g_1^y, g_1^z)$ and outputs 1 if and only if $x \cdot y = z \mod p$ and 0 otherwise. That means that the Decisional Diffie-Hellman problem is easy in gap groups. We denote by $\mathbb{G}^*$ the set of generators of the group $\mathbb{G}$, concretely $\mathbb{G}^* := \mathbb{G} \setminus \{1\}$, where 1 is the neutral element of $\mathbb{G}$

An example of a gap group is a symmetric bilinear group, hence $\mathbb{G}_1 = \mathbb{G}_2$. If a pairing is symmetric, then the DDH problem in $\mathbb{G}_1$ is indeed easy. To see this, notice that given a DDH tuple $(g_1, g_1^x, g_1^y, g_1^z)$, we can efficiently test if $z = x \cdot y$ in $\mathbb{Z}_p$ by executing the following check:

$$e(g_1^x, g_1^y) = e(g_1^z, g_1).$$

This is the case if and only if $e(g_1, g_1)^{xy} = e(g_1, g_1)^z$, which holds if and only if $x \cdot y = z \mod p$.

$\mathsf{ExpDLIN}^b_{\mathcal{A}}(\lambda)$:

$(p, \mathbb{G}, g, \mathsf{DDHvf}) \leftarrow \mathsf{GrpGen}(1^\lambda)$
$g_1, g_2 \xleftarrow{\$} \mathbb{G}^*, x, y \xleftarrow{\$} \mathbb{Z}_p, b \xleftarrow{\$} \{0,1\}$
if $b = 0 : z := x + y$
if $b = 1 : z \xleftarrow{\$} \mathbb{Z}_p$
$b' \leftarrow \mathcal{A}_\lambda(p, \mathbb{G}, \mathsf{DDHvf}, g_1, g_2, g, g_1^x, g_2^y, g^z)$
return b'

Figure 2.2: Security experiment for the DLIN assumption.

Definition 2.5.2 (Decision Linear Assumption (DLIN))**.** Consider the security experiment $\mathsf{ExpDLIN}^b_{\mathcal{A}}(\lambda)$ in Figure 2.2. The *decision linear assumption* holds relative to GrpGen if for all non-uniform PPT adversaries $\mathcal{A} = \{\mathcal{A}_\lambda\}_{\lambda \in \mathbb{N}}$ there exists a negligible function $\mathsf{negl}(\cdot)$ such that for all $\lambda \in \mathbb{N}$

$$\mathbf{Adv}^{\mathsf{DLin}}_{\mathcal{A}} = \left|\Pr[\mathsf{ExpDLIN}^0_{\mathcal{A}}(\lambda) = 1] - \Pr[\mathsf{ExpDLIN}^1_{\mathcal{A}}(\lambda) = 1]\right| \leq \mathsf{negl}(\lambda).$$

Next, we recall a definition of the sequential squaring assumption which was implicitly introduced by Rivest *et al.* [RSW96]. In this assumption we consider the parallel running time of an adversary and therefore we model the adversary as a circuit. Let p be an odd prime number. We say that p is a strong prime, if $p = 2p' + 1$ for some prime number p'. Let GenMod be a probabilistic polynomial-time algorithm which on input 1^λ outputs two λ-bit strong primes p and q, and

modulus N that is the product of p and q. Let $\varphi(\cdot)$ denote Euler's totient function. We denote by $\mathbb{QR}_N$ the cyclic group of quadratic residues which has order $|\mathbb{QR}_N| = \frac{\varphi(N)}{4} = \frac{(p-1)(q-1)}{4}$. We remark that it is possible to efficiently sample a random element x from $\mathbb{QR}_N$ by sampling $r \xleftarrow{\$} \mathbb{Z}_N^*$ and setting $x := r^2 \bmod N$.

$\underline{\mathsf{ExpSS}^b_{\mathcal{A}}(\lambda)\text{:}}$
$(p, q, N) \leftarrow \mathsf{GenMod}(1^\lambda)$
$x \xleftarrow{\$} \mathbb{QR}_N$
if $b = 0 : y := x^{2^{T(\lambda)}} \bmod N$
if $b = 1 : y \xleftarrow{\$} \mathbb{QR}_N$
return $b' \leftarrow \mathcal{A}_\lambda(N, T(\lambda), x, y)$

Figure 2.3: Security experiment for the sequential squaring assumption.

Definition 2.5.3 (Sequential Squaring Assumption (SS))**.** Consider the security experiment $\mathsf{ExpSS}^b_{\mathcal{A}}(\lambda)$ in Figure 2.3. The *sequential squaring assumption* with gap $0 < \epsilon < 1$ holds relative to GenMod if there exists a polynomial $\tilde{T}(\cdot)$ such that for all polynomials $T(\cdot) \geq \tilde{T}(\cdot)$ and for every non-uniform polynomial-size adversary $\mathcal{A} = \{\mathcal{A}_\lambda\}_{\lambda\in\mathbb{N}}$, where the depth of $\mathcal{A}_\lambda$ is at most $T^\epsilon(\lambda)$, there exists a negligible function $\mathsf{negl}(\cdot)$ such that for all $\lambda \in \mathbb{N}$

$$\mathbf{Adv}^{\mathsf{SS}}_{\mathcal{A}} = \left|\Pr[\mathsf{ExpSS}^0_{\mathcal{A}}(\lambda) = 1] - \Pr[\mathsf{ExpSS}^1_{\mathcal{A}}(\lambda) = 1]\right| \leq \mathsf{negl}(\lambda).$$

Some works [MT19, KLX20] also consider a stronger version of the SS assumption, the so-called strong sequential squaring assumption.

$\underline{\mathsf{ExpSSS}^b_{\mathcal{A}}(\lambda)\text{:}}$
$(p, q, N) \leftarrow \mathsf{GenMod}(1^\lambda)$
$\mathsf{st} \leftarrow \mathcal{A}_{1,\lambda}(N, T(\lambda))$
$x \xleftarrow{\$} \mathbb{QR}_N$
if $b = 0 : y := x^{2^{T(\lambda)}} \bmod N$
if $b = 1 : y \xleftarrow{\$} \mathbb{QR}_N$
return $b' \leftarrow \mathcal{A}_{2,\lambda}(x, y, \mathsf{st})$

Figure 2.4: Security experiment for the strong sequential squaring assumption.

Definition 2.5.4 (Strong Sequential Squaring Assumption (SSS))**.** Consider the security experiment $\mathsf{ExpSSS}^b_{\mathcal{A}}(\lambda)$ in Figure 2.4. The *strong sequential squaring*

assumption with gap $0 < \epsilon < 1$ holds relative to GenMod if there exists a polynomial $\tilde{T}(\cdot)$ such that for all polynomials $T(\cdot) \geq \tilde{T}(\cdot)$ and for every non-uniform polynomial-size adversary $\mathcal{A} = \{(\mathcal{A}_{1,\lambda}, \mathcal{A}_{2,\lambda})\}_{\lambda\in\mathbb{N}}$, where the depth of $\mathcal{A}_{2,\lambda}$ is at most $T^{\epsilon}(\lambda)$, there exists a negligible function $\mathsf{negl}(\cdot)$ such that for all $\lambda \in \mathbb{N}$

$$\mathbf{Adv}^{\mathsf{SSS}}_{\mathcal{A}} = \left|\Pr[\mathsf{ExpSSS}^0_{\mathcal{A}}(\lambda) = 1] - \Pr[\mathsf{ExpSSS}^1_{\mathcal{A}}(\lambda) = 1]\right| \leq \mathsf{negl}(\lambda).$$

Definition 2.5.5 (The Factoring Assumption)**.** The *factoring assumption* holds relative to GenMod if for every non-uniform PPT adversary $\mathcal{A} = \{\mathcal{A}_\lambda\}_{\lambda\in\mathbb{N}}$ there exists a negligible function $\mathsf{negl}(\cdot)$ such that for all $\lambda \in \mathbb{N}$

$$\mathbf{Adv}^{\mathtt{Factor}}_{\mathcal{A}} = \Pr\left[N = p'q' \;:\; \begin{array}{r}(p,q,N) \leftarrow \mathsf{GenMod}(1^\lambda)\\ p', q' \leftarrow \mathcal{A}_\lambda(N),\\ \text{such that } p', q' \in \mathbb{N}; p', q' > 1\end{array}\right] \leq \mathsf{negl}(\lambda).$$

2.6 The Random Oracle Model

The Random Oracle Model (ROM) is an idealized model which was introduced by Bellare and Rogaway [BR93]. It is used as a means to prove the security of cryptographic primitives which use as a building block a cryptographic hash function, whenever the proof for the given primitive using standard security notions of hash functions is not known.

In the ROM, the given hash function $H : \mathcal{X} \to \mathcal{Y}$ (where the set $\mathcal{Y}$ is finite) is treated as a random function to which everyone (i.e., adversaries and honest parties) has only so-called oracle access. Concretely, it means that when analysing security in the ROM the hash function H is replaced by a random function $\mathcal{O} \in \mathsf{Funs}[\mathcal{X}, \mathcal{Y}]$ (recall that $\mathsf{Funs}[\mathcal{X}, \mathcal{Y}]$ denotes the set of all functions from $\mathcal{X}$ to $\mathcal{Y}$) and everyone is allowed to evaluate $\mathcal{O}$ on arbitrary inputs.

There are three properties of the random oracle model which are particularly useful in security proofs:

1. The value of $H(x)$ is uniform.

   This allows to build up a random oracle on the fly which is known as *lazy sampling*. The main idea is to think about a random oracle as a possibly infinite lookup table T. On a query x, it is checked if x is in T. If this is the case, the value $\mathsf{T}[x]$ is returned. Otherwise, a uniform random value y is sampled from $\mathcal{Y}$, which is then returned and stored in the table as $\mathsf{T}[x] := y$.

2. The reduction can see all queries made by an adversary to a random oracle and hence learn these queries. This property is called *extractability*.

3. The value of $H(x)$ can be set by the reduction, as long as this value is uniform. This property is called *programmability*.

For more insightful discussion about the ROM see [KL14, KM15].

2.7 The Strong Algebraic Group Model

The Strong Algebraic Group Model (SAGM) was introduced by Katz *et al.* [KLX20] as a variant of the Algebraic Group Model (AGM) [FKL18] and makes it possible to consider time-sensitive assumptions, such as (Gap) Sequential Squaring. In the SAGM the running time of an algebraic algorithm is defined by the number of its algebraic steps with respect to some cyclic group $\mathbb{G}$ and algorithms are given access to actual group elements. In our case, $\mathbb{G}$ corresponds to the group $\mathbb{QR}_N$ of quadratic residues modulo N, for some N output by GenMod. Therefore, we use multiplication to denote the group operation. In the strong AGM, all algorithms are treated as strongly algebraic.

Definition 2.7.1 (Strongly algebraic algorithm [KLX20])**.** An algorithm $\mathcal{A}$ over $\mathbb{G}$ is called *strongly algebraic*, if in each (algebraic) step $\mathcal{A}$ does arbitrary local computation and then outputs[1] one or more tuples of the following form:

1. $(y, y_1, y_2) \in \mathbb{G}^3$, where $y = y_1 y_2$ and y_1, y_2 were either previously received by $\mathcal{A}$ or output by $\mathcal{A}$ in previous steps;

2. $(y, y_1) \in \mathbb{G}^2$, where $y = y_1^{-1}$ and y_1 was either provided as input to $\mathcal{A}$ or was output by $\mathcal{A}$ in some previous step.

The running time of $\mathcal{A}$ is the number of algebraic steps it takes.

Parallel computation is captured in the SAGM by allowing an algorithm to output multiple group elements at once (in one step, which counts as one query), but it is required that all of them can be represented by a sequence of previously output group elements. There are two ways how the running time of an adversary is measured: the number of group operations and the standard running time in some underlying computational model. Hence the running time is defined as a pair (t_1, t_2), which is understood in the following sense: running in time t_2 in the underlying computational model and using t_1 algebraic steps. For a more detailed description of SAGM and its relation to AGM and the Generic Group Model (GGM) [Sho97] we refer to the original paper by Katz *et al.* [KLX20].

[1] Here it is required that $\mathcal{A}$ outputs group elements at intermediate steps of its computation. The final output of $\mathcal{A}$ can be distinguished by requiring $\mathcal{A}$ to output a special indicator when generating its final output.

3 (Extractable) Offline Witness Encryption

Author's Contribution. The content of this chapter is joint work with Tibor Jager and Saqib A. Kakvi [CJK20]. The author has made the following contributions:

- formally defining semi-adaptive security of extractable offline witness encryption;
- discussing the constructions of extractable offline witness encryption and one-time puncturable tag-based encryption;
- proving security of the proposed construction of extractable offline witness encryption;
- formally defining semi-adaptive security of offline witness encryption;
- constructing an offline witness encryption scheme and proving its security.

For completeness, we include also sections about instantiation of one-time puncturable encryption.

Additionally, Section 3.4 is written by the author.

3.1 Introduction, Contributions and Related Work

Witness encryption (WE) [GGSW13] can be seen as a generalization of standard public-key encryption, where instead of encrypting messages and decrypting ciphertexts using a public/secret key pair, an instance/witness pair (x, w) of some NP language L is used. In more detail, witness encryption is defined with respect to a language L and allows to encrypt a message using an instance x which might or might not be in L. The knowledge of a witness w that x is indeed in the language enables decryption of a ciphertext to obtain the original message. We note that there might exist several witnesses for x and decryption should be successful for any of them. However, if there is no such a witness and hence x is not in L, then a produced ciphertext should hide the content of the message. As one

might expect, witness encryption is closely related to proof systems. Faonio *et al.* [FNV17] have shown equivalence of predictable arguments of knowledge and an extractable variant of witness encryption. Similarly, 1-bit laconic arguments can be built from WE and they also imply a variant of WE [BISW18]. The construction of WE from 1 group element laconic arguments has been presented in [BIOW20]. Since the seminal paper of Garg, Gentry, Sahai and Waters [GGSW13], witness encryption has turned out to be a versatile building block for cryptography with a plethora of applications.

Applications of WE. One of the applications of witness encryption is "non-interactive" payment for solving some (possibly well known) problem, e.g., the Riemann hypothesis. Anyone could deposit a prize for solving the given problem in a bank, encrypt the access details using the problem and publish the ciphertext with the bank details in a way that they are freely accessible. The first person with a valid solution to the problem can decrypt the ciphertext and hence gain access to the prize. An advantage of this approach is that we are able to access the prize without the entity that deposited the prize being present. The broad applicability of witness encryption has been presented already by Garg *et al.* [GGSW13], who gave constructions of a public-key, identity-based and attribute-based encryption schemes from witness encryption. There are many other applications of witness encryption and its variants in secure computation [BL18, CDG$^+$17, GLS15, KNY14], constructions of new primitives [BMSZ16, BH15, BGI14, GKP$^+$13a, LJKW18] or indeed novel constructions of known primitives [AJN$^+$16, BGI$^+$17, FNV17].

Variants and Extensions of Witness Encryption. There are several interesting variants and extensions of WE.

- *Functional* Witness Encryption (FWE) additionally allows to encrypt messages with respect to some function F. Decryption using a valid witness w reveals the value $F(m, w)$ instead of revealing the message m. This generalization of WE has been introduced by Boyle et al. [BCP14] and is equivalent to extractability obfuscation.

- *Reusable* Witness Encryption (RWE) introduced by Zhandry[Zha16] is key-encapsulation scheme with a setup which outputs parameters and master decryption key (allowing CCA-type security). Parameters are used by both encryption algorithm and decryption algorithm. This type of witness encryption yields attribute-based encryption with short ciphertexts.

- *Extractable* Witness Encryption (EWE) offers stronger security than standard WE and hence broadens its applicability. The notion of EWE was

proposed in [GKP+13a]. EWE guarantees that anyone who is able to distinguish encryptions of two messages under a statement x must "know" a corresponding witness w. As we have already mentioned this variant is equivalent to predictable arguments of knowledge [FNV17] and has applications such as time-lock encryption [LJKW18], secret sharing for NP [KNY14], running Turing machines on encrypted data [GKP+13a], asymmetric password-based cryptography [BH15], and functional signatures [BGI14].

- *Witness Selector* [BL18] guarantees that hiding property holds not only if there does not exist a witness for an instance but even if it is computationally hard to find a witness. Therefore from the perspective of security, a witness selector scheme guarantees stronger security than WE but weaker security than EWE.

- *Offline* Witness Encryption (OWE) was proposed by Abusalah *et al.* [AFP16] in order to improve the efficiency of the encryption in WE. Similarly to RWE, OWE has a setup algorithm that outputs public encryption and public decryption parameters. The encryption can be usually realized with tools of classical public-key cryptography, however, decryption is typically more expensive and requires for example an obfuscation. Therefore OWE is especially useful in scenarios when there is a disparity in the powers of the encryptor and decryptor, such as in the case of Asymmetric Password-Based Encryption [BH15].

Contributions. We introduce a new notion of *Extractable Offline* Witness Encryption (EOWE) which is an offline variant of extractable witness encryption proposed by Goldwasser *et al.* [GKP+13a]. The known constructions of EWE are based either on multilinear maps [BS02, BS03, FHHL18, GGH12, GGH13] or extractability obfuscation, hence these constructions are not practical or even feasible yet. Specifically, the constructions of EWE based on extractability obfuscation require obfuscating a circuit that internally checks if the given instance/witness pair is valid. If this is the case, the circuit outputs a decrypted message. In particular, the encryption algorithm in this solution is impractical, because it requires obfuscating a circuit and moreover the ciphertext is the obfuscated circuit. As a stepping stone to efficiency improvements of the encryption algorithm, we allow for trusted setup in a similar manner as [AFP16]. We introduce a novel generic technique to construct extractable offline witness encryption. This technique is flexible enough that it can be adjusted to obtain standard offline witness encryption. Moreover, our approach leads to stronger security and reduced ciphertext overhead compared to the original proposal of Abusalah *et al.* [AFP16]. Concretely, our construction is the first one achieving adaptive chosen-message security. The prior work only

provided selective security, where an adversary has to commit to the "challenge" messages m_0, m_1 even before seeing the public parameters of the scheme.

The basic idea is to combine public-key encryption with obfuscation. At the first sight, one could think that any combination of IND-CCA-secure public-key encryption with iO should lead to Offline Witness Encryption. Unfortunately, this straightforward approach does not seem to work and a more delicate method is needed. The construction of Abusalah *et al.* [AFP16] uses Naor-Yung [NY90] style double encryption which requires encrypting a message twice using IND-CPA-secure public-key encryption and a statistical simulation-sound zero-knowledge proof (SSS-NIZK). Even though Abusalah *et al.* propose an efficient instantiation of SSS-NIZK using a Groth-Sahai-like pairing-based proof system [GS08], the proof consists of a rather large number of group elements and it grows linearly with the size of the message. In this thesis, we rely on a novel approach. We show that puncturable tag-based encryption [GM15] can be used to obtain (extractable) offline witness encryption which has a more efficient instantiation that requires only one ciphertext, no zero-knowledge proofs, and achieves stronger security guarantees. The improved security and efficiency extend the applicability of offline witness encryption schemes in cryptography.

We propose a new variant of puncturable tag-based encryption which we denote as one-time puncturable encryption and show that this variant is sufficient for our construction. Moreover, we provide an efficient instantiation of this primitive based on the tag-based public-key encryption of Kiltz [Kil06].

Applications of Semi-Adaptive Offline Witness Encryption. Even though our constructions do not achieve full adaptive security, there are interesting applications where semi-adaptive security suffices. One example is the already discussed non-interactive payment for solving some problem. In this scenario, the problem is fixed once and for all. Another application is the more recent primitive of Time-Lock Encryption (TLE) introduced by Liu *et al.* [LJKW18]. Because we discuss TLE in more detail later in Section 3.4, we provide here only a short overview. TLE has several interesting applications including, but not limited to: responsible disclosure, pre-distribution of digital media, sealing of auction tenders, and publication of grades. As we explain later, the instance in the case of TLE is release time, which in many scenarios might be fixed at the time of executing the setup. Therefore, the use of semi-adaptive EOWE is reasonable in this case as well. As OWE and EOWE are relatively new primitives, one can expect more applications to follow in the near future.

Related Work. The notion of offline witness encryption was firstly considered by Abusalah, Fuchsbauer and Pietrzak [AFP16], who constructed it by combin-

ing indistinguishability obfuscation with Naor-Yung style IND-CCA-secure PKE. This work also introduced Offline Functional Witness Encryption (OFWE), where encryption is applied to a message and some function F from a supported function family: given a witness w for the corresponding instance x the value $F(m, w)$ is returned. Similarly, one could define also Extractable Offline Functional Witness Encryption (EOFWE). Both constructions of Abusalah *et al.* achieve only selective security which is weaker security than our semi-adaptive security and at the same time, they rely on more complex primitives like simulation-sound NIZKs. Therefore the ciphertext size of our scheme is smaller. We note that our constructions of OWE and EOWE can be simply adjusted to the functional setting of OFWE and EOFWE, however, this adjustment means that the constructions are only selectively secure. We discuss this in more detail later. Moreover, we believe that their construction can be adapted to yield an EOWE and EOFWE, by replacing the indistinguishability obfuscator with an extractability obfuscator.

Recently, two new works on OWE of Pal and Dutta have appeared [PD20, PD19]. The work [PD20] provides constructions for all four primitives - OWE, EOWE, OFWE, and EOFWE. It achieves semi-adaptive security for OWE and EOWE and shorter ciphertext size than our work by using private-key encryption, however, it relies on iO and eO, respectively, and puncturable witness PRF which itself requires iO. Therefore, the mentioned constructions are less efficient than ours in the sense of runtime of the setup and the encryption algorithm. The proposed OFWE and EOFWE achieve only selective security and suffer the same efficiency limitations. The work [PD19] considers OWE and OFWE. The constructions are based on randomized encodings in the CRS model, extractable witness pseudorandom functions (PRFs), and rely on sub-exponential assumptions. Both constructions achieve only selective security. We remark that known constructions of extractable witness PRFs are based on multilinear maps which are comparable to iO.

Goldwasser *et al.* [GKP$^+$13a] have defined extractable WE and construct it from multilinear maps. Boyle, Chung and Pass [BCP14] consider extractable functional witness encryption and they show its equivalence with extractability obfuscation. We remark that Boyle *et al.* use the term functional WE instead of extractable FWE, however, the term extractable functional witness encryption is more appropriate for their definition.

Reusable Witness Encryption (RWE) has been proposed by Zhandry [Zha16] which similarly to OWE has a setup algorithm. But RWE is defined as a key encapsulation mechanism, the setup outputs one type of parameters which are used by both encryption and decryption algorithm, and additionally, it outputs a master decryption key.

3.2 Definitions of (Extractable) Offline Witness Encryption

Before stating our definition of Extractable Offline Witness Encryption and explaining the intuition behind it, we recap the notions of Witness Encryption and Offline Witness Encryption. Witness Encryption for an NP language L is a pair of algorithms $(\mathsf{Enc}, \mathsf{Dec})$. The encryption algorithm takes an instance x (not necessarily from L) and a message m as input and outputs a ciphertext c. The decryption algorithm takes as input a ciphertext c and a witness w and outputs a message m if w is a witness for the instance x. However, if the instance x is not in the language, then informally no efficient adversary is able to get any information about the encrypted message from the ciphertext. *Offline* Witness Encryption has been proposed by Abusalah, Fuchsbauer and Pietrzak [AFP16] in order to improve the efficiency of witness encryption. OWE has an additional algorithm Setup in which the most costly computation should be executed. This allows to improve the efficiency of the encryption and therefore this variant of WE is useful in scenarios where there is a discrepancy in computational power between encryptor and decryptor, as is quite often the case. The setup algorithm produces two types of public parameters: one type used by the encryption algorithm and the other one by the decryption algorithm.

Next, we recall the definition of OWE.

Definition 3.2.1. An *offline witness encryption scheme* OWE for an NP language L (with corresponding relation $\mathcal{R}$) with message space $\mathcal{M}$ is defined as a triple of efficient algorithms $\mathtt{OWE} = (\mathsf{Setup}, \mathsf{Enc}, \mathsf{Dec})$:

- $(\mathsf{pp}_e, \mathsf{pp}_d) \leftarrow \mathsf{Setup}(1^\lambda)$ is a probabilistic algorithm that takes as input the security parameter 1^λ and outputs parameters pp_e for encryption and parameters pp_d for decryption.
- $c \leftarrow \mathsf{Enc}(\mathsf{pp}_e, x, m)$ is a probabilistic algorithm that takes as input the encryption parameters pp_e, an instance x, and a message $m \in \mathcal{M}$ and outputs a ciphertext c.
- $m/\perp \leftarrow \mathsf{Dec}(\mathsf{pp}_d, c, w)$ is a deterministic algorithm that takes as input the decryption parameters pp_d, a ciphertext c, and a witness w and outputs $m \in \mathcal{M}$ if $(x, w) \in \mathcal{R}$ and $\perp$ otherwise.

We say OWE is *correct* if for all $\lambda \in N$, for all messages $m \in \mathcal{M}$, for all $x \in L$ and for all w such that $(x, w) \in \mathcal{R}$, we have

$$\Pr[\mathsf{Dec}(\mathsf{pp}_d, \mathsf{Enc}(\mathsf{pp}_e, x, m), w) = m : (\mathsf{pp}_e, \mathsf{pp}_d) \leftarrow \mathsf{Setup}(1^\lambda)] = 1.$$

$\mathsf{ExpOWE}_{\mathcal{A}}(\lambda)$:

$x \leftarrow \mathcal{A}_\lambda$
$(\mathsf{pp}_e, \mathsf{pp}_d) \leftarrow \mathsf{Setup}(1^\lambda)$
$(m_0, m_1) \leftarrow \mathcal{A}_\lambda(\mathsf{pp}_e, \mathsf{pp}_d)$
$b \xleftarrow{\$} \{0,1\}; c^* \leftarrow \mathsf{Enc}(\mathsf{pp}_e, x, m_b)$
$b' \leftarrow \mathcal{A}_\lambda(c^*)$
return $(b' = b \wedge x \notin L)$

Figure 3.1: Security experiment for OWE.

The security of OWE is defined using the experiment in Figure 3.1. This experiment defines security in the semi-adaptive setting, where the adversary must commit to the instance, but not the messages, before seeing parameters.

Definition 3.2.2. Consider the security experiment $\mathsf{ExpOWE}_{\mathcal{A}}(\lambda)$ in Figure 3.1. An offline witness encryption scheme $\mathtt{OWE}$ for a language L with corresponding relation $\mathcal{R}$ is *semi-adaptively secure*, if for all non-uniform PPT adversaries $\mathcal{A} = \{\mathcal{A}_\lambda\}_{\lambda \in \mathbb{N}}$ (where we require that $\mathcal{A}$'s output satisfies $|m_0| = |m_1|$) there is a negligible function $\mathsf{negl}(\cdot)$ such that for all $\lambda \in \mathbb{N}$

$$\mathbf{Adv}^{\mathtt{OWE}}_{\mathcal{A}} = \left| \Pr[\mathsf{ExpOWE}_{\mathcal{A}}(\lambda) = 1] - \frac{1}{2} \right| \leq \mathsf{negl}(\lambda).$$

Finally, we discuss *Extractable* Offline Witness Encryption (EOWE). Syntactically EOWE is the same as OWE but from the security perspective, we require that if the adversary can distinguish between encryptions of two messages of its choice under some instance x, then it must "know" a corresponding witness w. Our definition of extractability is similar to the definition of extractable WE introduced by Goldwasser *et al.* [GKP+13a]. To make the above intuition precise we define the following experiment:

$\mathsf{ExpEOWE}_{\mathcal{A}}(\lambda, x)$:

$(\mathsf{pp}_e, \mathsf{pp}_d) \leftarrow \mathsf{Setup}(1^\lambda)$
$(m_0, m_1) \leftarrow \mathcal{A}_\lambda(\mathsf{pp}_e, \mathsf{pp}_d, x)$
$b \xleftarrow{\$} \{0,1\}; c^* \leftarrow \mathsf{Enc}(\mathsf{pp}_e, x, m_b)$
$b' \leftarrow \mathcal{A}_\lambda(c^*)$
return $b' = b$

Figure 3.2: Security experiment for EOWE.

Definition 3.2.3. Consider the security experiment $\mathsf{ExpEOWE}_{\mathcal{A}}(\lambda, x)$ in Figure 3.2. An extractable offline witness encryption scheme $\mathtt{EOWE}$ for a language L with corresponding relation $\mathcal{R}$ is *semi-adaptively secure*, if for all non-uniform PPT adversaries $\mathcal{A} = \{\mathcal{A}_\lambda\}_{\lambda\in\mathbb{N}}$ (where we require that $\mathcal{A}$'s output satisfies $|m_0| = |m_1|$), for every polynomial $\mathsf{p}(\lambda)$ there exist a non-uniform PPT extractor $\mathcal{E} = \{\mathcal{E}_\lambda\}_{\lambda\in\mathbb{N}}$ and a polynomial $\mathsf{q}(\lambda)$ such that for every $\lambda \in \mathbb{N}$ and for all $x \in \{0,1\}^*$, it holds that

$$\left|\Pr[\mathsf{ExpEOWE}_{\mathcal{A}}(\lambda, x) = 1] - \frac{1}{2}\right| \geq \frac{1}{\mathsf{p}(\lambda)}$$
$$\implies \Pr[(x, w) \in \mathcal{R} : w \leftarrow \mathcal{E}_\lambda(x)] \geq \frac{1}{\mathsf{q}(\lambda)}.$$

Our constructions of OWE and EOWE cleverly combine only two building blocks. One of them is obfuscation, specifically indistinguishability or extractability obfuscation. The details regarding obfuscation can be found in Section 2.3.6. The second primitive is puncturable tag-based encryption. Since we require a novel security notion for puncturable tag-based encryption which is weaker than the standard security notions considered in other works, we discuss this primitive in the next section. Later, we also show how to efficiently instantiate it from a concrete tag-based encryption scheme.

3.3 Puncturable Tag-Based Encryption

Puncturable Encryption (PE) extends the capabilities of classical public-key encryption. In addition to the algorithms $(\mathsf{Gen}, \mathsf{Enc}, \mathsf{Dec})$, it provides a puncturing algorithm Punct, which is capable of adjusting the secret key in a way that some ciphertext become "undecryptable". This concept was firstly discussed by Anderson [And97], however, the formal definition of PE was later given by Green and Miers [GM15] who also construct this primitive. In our work, we focus on the tag-based variant of puncturable encryption in which the encryption and the decryption algorithm also take some tag t as input. The puncturing of the secret key is as well done with respect to a tag t. The resulting punctured secret key is able to decrypt all ciphertexts which have not been encrypted with the tag on which the key was punctured. This property is sometimes referred to as "all-but-one" decryption. A standard puncturable tag-based encryption allows repeated puncturing on several different tags. However, as we show for our construction it is sufficient to puncture on one tag only. We denote such PE as *one-time* puncturable encryption. Syntactically, we differ from the standard definition by allowing an alternative decryption algorithm PDec which uses the punctured secret key for decrypting ciphertexts. In this way, the original secret key and punctured secret key can have different forms.

Definition 3.3.1. A *one-time puncturable tag-based encryption scheme* $\mathtt{PE}$ for message space $\mathcal{M}$ and tag space $\mathcal{T}$ is defined as a tuple of efficient algorithms $\mathtt{PE} = (\mathsf{Gen}, \mathsf{Punct}, \mathsf{Enc}, \mathsf{Dec}, \mathsf{PDec})$:

- $(\mathsf{pk}, \mathsf{sk}) \leftarrow \mathsf{Gen}(1^\lambda)$ is a probabilistic algorithm that takes as input the security parameter 1^λ and outputs public key pk and an unpunctured secret key sk.
- $\mathsf{sk}_{t^*} \leftarrow \mathsf{Punct}(\mathsf{sk}, t^*)$ is a probabilistic algorithm that takes as input an unpunctured secret key sk and a single tag $t^* \in \mathcal{T}$ and outputs a secret key punctured at t^* denoted by sk_{t^*}.
- $c \leftarrow \mathsf{Enc}(\mathsf{pk}, m, t)$ is a probabilistic algorithm that takes as input the public key pk, a message $m \in \mathcal{M}$, and a tag $t \in \mathcal{T}$ and outputs a ciphertext c.
- $m/\perp \leftarrow \mathsf{Dec}(\mathsf{sk}, c, t)$ is a deterministic algorithm that takes as input the unpunctured secret key sk, a ciphertext c, and a tag $t \in \mathcal{T}$ and outputs $m \in \mathcal{M}$ or $\perp$.
- $m/\perp \leftarrow \mathsf{PDec}(\mathsf{sk}_{t^*}, m, t)$ is a deterministic algorithm which takes as input a punctured secret key sk_{t^*}, a ciphertext c, and a tag $t \in \mathcal{T}$ with $t \neq t^*$ and outputs $m \in \mathcal{M}$ or $\perp$.

We say $\mathtt{PE}$ is *correct* if for all $\lambda \in \mathbb{N}$, for all key pairs $(\mathsf{pk}, \mathsf{sk}) \leftarrow \mathsf{Gen}(1^\lambda)$, all messages $m \in \mathcal{M}$ and for all tags $t, t' \in \mathcal{T}$, where $t' \neq t$, we have

$$\Pr[\mathsf{Dec}(\mathsf{sk}, \mathsf{Enc}(\mathsf{pk}, m, t), t) = m] = 1 \wedge \Pr[\mathsf{Dec}(\mathsf{sk}, \mathsf{Enc}(\mathsf{pk}, m, t), t') = \perp] = 1.$$

and further for any tag t^*, all punctured keys $\mathsf{sk}_{t^*} \xleftarrow{\$} \mathsf{Punct}(\mathsf{sk}, t^*)$ and all tags $t \neq t^*$, we have

$$\Pr[\mathsf{PDec}(\mathsf{sk}_{t^*}, \mathsf{Enc}(\mathsf{pk}, m, t), t) = m] = 1 \wedge \Pr[\mathsf{PDec}(\mathsf{sk}_{t^*}, \mathsf{Enc}(\mathsf{pk}, m, t^*), t) = \perp] = 1.$$

For our construction, we require relatively weak security, namely *selective indistinguishability from random*. Our security definition deviates from the standard definition of Green and Miers [GM15] by not giving the adversary access to a decryption oracle since we are puncturing only on one tag. We define the security of PE via the experiment in Figure 3.3.

Definition 3.3.2. Consider the security experiment $\mathsf{ExpPE}_{\mathcal{A}}(\lambda)$ in Figure 3.3. We say a one-time puncturable tag-based encryption scheme $\mathtt{PE}$ is *secure*, if for all non-uniform PPT adversaries $\mathcal{A} = \{\mathcal{A}_\lambda\}_{\lambda \in \mathbb{N}}$ (where we require that $\mathcal{A}$'s output satisfies $|m_0| = |m_1|$) there is a negligible function $\mathsf{negl}(\cdot)$ such that for all $\lambda \in \mathbb{N}$

$$\mathbf{Adv}^{\mathtt{PE}}_{\mathcal{A}} = \left| \Pr[\mathsf{ExpPE}_{\mathcal{A}}(\lambda) = 1] - \frac{1}{2} \right| \leq \mathsf{negl}(\lambda).$$

$\mathsf{ExpPE}_{\mathcal{A}}(\lambda)$:

$t^* \leftarrow \mathcal{A}_\lambda, b \xleftarrow{\$} \{0,1\}$
$(\mathsf{pk}, \mathsf{sk}) \leftarrow \mathsf{Gen}(1^\lambda)$
$\mathsf{sk}_{t^*} \leftarrow \mathsf{Punct}(\mathsf{sk}, t^*)$
$(m_0, m_1) \leftarrow \mathcal{A}_\lambda(\mathsf{pk}, \mathsf{sk}_{t^*})$
$c \leftarrow \mathsf{Enc}(\mathsf{pk}, m_b, t^*)$
$b' \leftarrow \mathcal{A}_\lambda(c)$
return $b = b'$

Figure 3.3: Security experiment for PE.

3.4 Offline Time-Lock Encryption

Before stating technical details of our constructions, we explain how extractable offline witness encryption can be used to build a variant of time-lock encryption. Time-lock encryption [LJKW18], as a means for sending messages into the future, has numerous advantages compared to other approaches which either rely on trusted agents or execution of expensive computations. The construction proposed by Liu *et al.* [LJKW18] make use of extractable witness encryption. Because we want to rely on offline EWE, we adjust the definition of TLE by introducing an additional setup algorithm. We denote such TLE as *offline* time-lock encryption (OTLE). Before defining OTLE we state the definition of *computational reference clock* introduced in [LJKW18].

Definition 3.4.1 (Computational Reference Clock)**.** A *computational reference clock* is a stateful probabilistic machine $\mathcal{C}(1^\lambda)$ that takes as input the security parameter 1^λ and outputs an infinite sequence $w_1, w_2, \ldots$ in the following way. The initial state of $\mathcal{C}$ is w_0. It runs a probabilistic algorithm $f_{\mathcal{C}}$ which computes $w_T := f_{\mathcal{C}}(w_{T-1})$ and outputs w_T.

To abbreviate the process of executing the clock T times in a row, starting from initial state and outputting the state w_T of $\mathcal{C}$, we use notation $w_T \leftarrow \mathcal{C}(1^\lambda, T)$.

Definition 3.4.2. We say that language L with corresponding relation $\mathcal{R}$ is associated to $\mathcal{C}$, if L is an NP language and for all $x \leq T$, it holds that

$$\Pr\left[(1^x, w_T) \in \mathcal{R} : w_T \leftarrow \mathcal{C}(1^\lambda, T)\right] = 1.$$

Definition 3.4.3. A computational reference clock $\mathcal{C}$ with associated NP language L (and corresponding relation $\mathcal{R}$) is *secure* if for all non-uniform PPT adversaries

$\mathcal{A} = \{\mathcal{A}_\lambda\}_{\lambda\in\mathbb{N}}$, all $T \in \mathbb{N}$, there is a negligible function $\mathsf{negl}(\cdot)$ such that for all $\lambda \in \mathbb{N}$

$$\mathbf{Adv}^{\mathcal{C}}_{\mathcal{A}} = \left|\Pr\left[(1^T, w_T) \in \mathcal{R} \ : \ w_T \leftarrow \mathcal{A}^{\mathcal{C}}_\lambda(T)\right] - \frac{1}{2}\right| \leq \mathsf{negl}(\lambda),$$

where $\mathcal{C}$ returns the value $w := f_{\mathcal{C}}(w)$ and the initial state of the $\mathcal{C}$ is $w := w_0$. The adversary $\mathcal{A}$ is allowed to make at most $T-1$ queries to $\mathcal{C}$ in total.

As an instantiation of a computational reference clock Liu *et al.* [LJKW18] suggested to use the Bitcoin blockchain. Now, we state the definition of offline time-lock encryption.

Definition 3.4.4 (Offline Time-Lock Encryption)**.** An offline time-lock encryption shceme (OTLE) for computational reference clock $\mathcal{C}(1^\lambda)$ with message space $\mathcal{M}$ is a tuple of efficient algorithms $\mathtt{OTLE} = (\mathsf{Setup}, \mathsf{Enc}, \mathsf{Dec})$ such that

- $(\mathsf{pp}_e, \mathsf{pp}_d) \leftarrow \mathsf{Setup}(1^\lambda)$ is a probabilistic algorithm which takes as input the security parameter 1^λ and outputs parameters pp_e for encryption and parameters pp_d for decryption.
- $c \leftarrow \mathsf{Enc}(\mathsf{pp}_e, T, m)$ is a probabilistic algorithm which takes as input the encryption parameters pp_e, an integer $T \in \mathbb{N}$, and a message m and outputs a ciphertext c.
- $m/\perp \leftarrow \mathsf{Dec}(\mathsf{pp}_d, c, w)$ is a deterministic algorithm which takes as input the decryption parameters pp_d, a ciphertext c, and $w \in \{0,1\}^*$ and outputs $m \in \mathcal{M}$ or $\perp$.

We say $\mathtt{OTLE}$ is *correct* if for all $\lambda, T, T' \in \mathbb{N}$, where $T' \geq T$ and all $m \in \mathcal{M}$ we have

$$\Pr[\mathsf{Dec}(\mathsf{pp}_d, \mathsf{Enc}(\mathsf{pp}_e, T, m), \mathcal{C}(1^\lambda, T')) = m : (\mathsf{pp}_e, \mathsf{pp}_d) \leftarrow \mathsf{Setup}(1^\lambda)] = 1.$$

Definition 3.4.5. We say that $\mathtt{OTLE}$ is *secure* if for all non-uniform PPT adversaries $\mathcal{A}$, for all $T \in \mathbb{N}$, there is a negligible function $\mathsf{negl}(\cdot)$ such that for all $\lambda \in \mathbb{N}$

$$\mathbf{Adv}^{\mathtt{OTLE}}_{\mathcal{A}} = \left|\Pr\left[b = b' \ : \ \begin{array}{r}(\mathsf{pp}_e, \mathsf{pp}_d) \leftarrow \mathsf{Setup}(1^\lambda)\\ (m_0, m_1) \leftarrow \mathcal{A}^{\mathcal{C}}_\lambda(\mathsf{pp}_e, \mathsf{pp}_d, T)\\ b \xleftarrow{\$} \{0,1\};\ c \leftarrow \mathsf{Enc}(\mathsf{pp}_e, T, m_b)\\ b' \leftarrow \mathcal{A}^{\mathcal{C}}_\lambda(c)\end{array}\right] - \frac{1}{2}\right| \leq \mathsf{negl}(\lambda),$$

where $\mathcal{C}$ returns the value $w := f_{\mathcal{C}}(w)$ and the initial state of the $\mathcal{C}$ is $w := w_0$. The adversary $\mathcal{A} = \{\mathcal{A}_\lambda\}_{\lambda\in\mathbb{N}}$ is allowed to make at most $T-1$ queries to $\mathcal{C}$ in total and we require that $|m_0| = |m_1|$.

Construction of OTLE. To construct an offline time-lock encryption scheme we can directly use an extractable offline witness encryption scheme $\texttt{EOWE} = (\texttt{EOWE.Setup}, \texttt{EOWE.Enc}, \texttt{EOWE.Dec})$. Let $\mathcal{C}$ be a secure computational reference clock and let L be the NP language associated with $\mathcal{C}$. Let $\texttt{EOWE} = (\texttt{EOWE.Setup}, \texttt{EOWE.Enc}, \texttt{EOWE.Dec})$ be an extractable offline witness encryption for L. Then $\texttt{OTLE} = (\mathsf{Setup}, \mathsf{Enc}, \mathsf{Dec})$ defined as

- $\mathsf{Setup}(1^\lambda) := \texttt{OWE.Setup}(1^\lambda)$
- $\mathsf{Enc}(\mathsf{pp}_e, T, m) := \texttt{OWE.Enc}(\mathsf{pp}_e, 1^T, m)$
- $\mathsf{Dec}(\mathsf{pp}_d, c, w) := \texttt{OWE.Dec}(\mathsf{pp}_d, c, w)$

is a secure offline time-lock encryption scheme. The proof is straightforward adaptation of the proof of Theorem 1 from [LJKW18].

3.5 Constructions

In this section, we provide constructions and security proofs of our offline witness encryption and extractable offline witness encryption. Both constructions follow the same template and make use of one-time puncturable tag-based encryption together with obfuscation (see Section 2.3.6). We start by presenting the construction of OWE. After that, we give the construction with the proof for EOWE.

3.5.1 Construction of OWE

Let $\texttt{PE} = (\texttt{PE.KeyGen}, \texttt{PE.Enc}, \texttt{PE.Punct}, \texttt{PE.Dec})$ be a one-time puncturable encryption and $i\mathcal{O}$ an indistinguishablity obfuscator for a circuit class $\mathcal{C}_\lambda$. Our construction of offline witness encryption $\texttt{OWE} = (\mathsf{Setup}, \mathsf{Enc}, \mathsf{Dec})$ for a language L is given in Figure 3.4. We assume that the decryption circuit is padded to the maximum length of sizes of all circuits appearing in the security proof, hence, all circuits have the same size.

Theorem 3.5.1. *Let* $\texttt{PE} = (\texttt{PE.KeyGen}, \texttt{PE.Enc}, \texttt{PE.Punct}, \texttt{PE.Dec})$ *be a secure one-time puncturable tag-based encryption and* $i\mathcal{O}$ *be a secure indistinguishability obfuscator. Then* $(\mathsf{Setup}, \mathsf{Enc}, \mathsf{Dec})$ *defined in Figure 3.4 is a semi-adaptively secure offline witness encryption.*

Proof. Correctness of the scheme is implied by the correctness of the puncturable encryption scheme and the indistinguishability obfuscator. To prove security we define a series of games $\mathsf{G}_0 - \mathsf{G}_1$ which are computationally indistinguishable. The individual games differ in how we realize our setup and the decryption circuit. For $i \in \{0, 1\}$ we denote by $\mathsf{G}_i = 1$ the event that the adversary $\mathcal{A} = \{\mathcal{A}_\lambda\}_{\lambda \in \mathbb{N}}$ outputs in G_i values x and b' such that $b' = b \wedge x \notin L$.

$\underline{\mathsf{Setup}(1^\lambda)}$
$(\mathsf{sk}, \mathsf{pk}) \leftarrow \mathtt{PE.Gen}(1^\lambda)$
$\widetilde{C}_{\mathsf{sk}} \leftarrow i\mathcal{O}(1^\lambda, C_{\mathsf{sk}})$
$\mathsf{pp}_e := \mathsf{pk}$, $\mathsf{pp}_d := \widetilde{C}_{\mathsf{sk}}$
return $(\mathsf{pp}_e, \mathsf{pp}_d)$

$\underline{\mathsf{Enc}(\mathsf{pp}_e, x, m)}$
$c_{pe} \leftarrow \mathtt{PE.Enc}(\mathsf{pp}_e, m, x)$
return $c \leftarrow (c_{pe}, x)$

$\underline{C_{\mathsf{sk}}(c, w)}$
Parse c as (c_{pe}, t)
if $R(t, w) = 1$
  $m \leftarrow \mathtt{PE.Dec}(\mathsf{sk}, c_{pe}, t)$
  return m
return $\perp$

$\underline{\mathsf{Dec}(\mathsf{pp}_d, c, w)}$
return $m \leftarrow \widetilde{C}_{\mathsf{sk}}(c, w)$

Figure 3.4: Construction of OWE

Game 0. Game G_0 (Figure 3.5) corresponds to the original security experiment, where we use the Setup, Enc, and C_{sk} directly from our construction.

$\underline{\mathsf{G}_0(1^\lambda)}$
$x \xleftarrow{\$} \mathcal{A}_\lambda$
$(\mathsf{pp}_e, \mathsf{pp}_d) \leftarrow \mathsf{Setup}(1^\lambda)$
$(m_0, m_1) \leftarrow \mathcal{A}_\lambda(\mathsf{pp}_e, \mathsf{pp}_d)$
$b \xleftarrow{\$} \{0, 1\}$
$c^* \leftarrow \mathsf{Enc}(\mathsf{pp}_e, x, m_b)$
$b' \leftarrow \mathcal{A}_\lambda(c^*)$
return $(b' = b \wedge x \notin L)$

$\underline{C_{\mathsf{sk}}(c, w)}$
Parse c as (c_{pe}, t)
if $R(t, w) = 1$
  $m \leftarrow \mathtt{PE.Dec}(\mathsf{sk}, c_{pe}, t)$
  return m
return $\perp$

Figure 3.5: Game G_0

We now define in Figure 3.6 an alternative setup algorithm Setup'. This algorithm differs from Setup in that we additionally puncture the secret key sk on the challenge tag x.

$\underline{\mathsf{Setup}'(1^\lambda, x)}$
$(\mathsf{sk}, \mathsf{pk}) \leftarrow \mathtt{PE.Gen}(1^\lambda)$
$\mathsf{sk}^* \leftarrow \mathtt{PE.Punct}(\mathsf{sk}, x)$
$\widetilde{C}_{\mathsf{sk}^*, x} \leftarrow i\mathcal{O}(1^\lambda, C_{\mathsf{sk}^*, x})$
$\mathsf{pp}_e := \mathsf{pk}$, $\mathsf{pp}_d := \widetilde{C}_{(\mathsf{sk}^*, x)}$
return $(\mathsf{pp}_e, \mathsf{pp}_d)$

Figure 3.6: Alternative setup

Game 1. In G_1 (Figure 3.7) we run our alternative setup algorithm Setup', which punctures the secret key sk on the tag x. The decryption circuit uses the punctured key sk^* and returns $\perp$ if target tag x is equal to tag t of the ciphertext.

$\underline{\mathsf{G}_1(1^\lambda)}$	$\underline{C_{\mathsf{sk}^*,x}(c,w)}$
$x \leftarrow \mathcal{A}_\lambda$	Parse c as (c_{pe}, t)
$\boxed{(\mathsf{pp}_e, \mathsf{pp}_d) \leftarrow \mathsf{Setup}'(1^\lambda, x)}$	if $R(t,w) = 1$
$(m_0, m_1) \leftarrow \mathcal{A}_\lambda(\mathsf{pp}_e, \mathsf{pp}_d)$	if $x = t$
$b \xleftarrow{\$} \{0,1\}$	return $\perp$
$c^* \leftarrow \mathsf{Enc}(\mathsf{pp}_e, x, m_b)$	$\boxed{m \leftarrow \mathsf{PE.PDec}(\mathsf{sk}^*, c_{pe}, t)}$
$b' \leftarrow \mathcal{A}_\lambda(c^*)$	return m
return $(b' = b \wedge x \notin L)$	return $\perp$

Figure 3.7: Game G_1

Lemma 3.5.1. *From any PPT adversary $\mathcal{A} = \{\mathcal{A}_\lambda\}_{\lambda\in\mathbb{N}}$ we can construct a PPT distinguisher $\mathcal{D} = \{\mathcal{D}_\lambda\}_{\lambda\in\mathbb{N}}$ such that*

$$\mathbf{Adv}^{i\mathcal{O}}_{\mathcal{D}} = \left|\Pr[\mathsf{G}_0 = 1] - \Pr[\mathsf{G}_1 = 1]\right|.$$

To prove the lemma we construct a distinguisher $\mathcal{D} = \{\mathcal{D}_\lambda\}_{\lambda\in\mathbb{N}}$ which breaks the security of $i\mathcal{O}$ as follows.

The distinguisher $\mathcal{D}_\lambda$:

1. Runs $x \leftarrow \mathcal{A}_\lambda$.
2. Generates $(\mathsf{sk}, \mathsf{pk}) \leftarrow \mathsf{PE.Gen}(1^\lambda)$.
3. Punctures the key $\mathsf{sk}^* \leftarrow \mathsf{PE.Punct}(\mathsf{sk}, x)$.
4. Constructs $C_0 := C_{\mathsf{sk}}, C_1 := C_{\mathsf{sk}^*,x}$.
5. Submits C_0, C_1 to the $i\mathcal{O}$ challenger and obtains $\widetilde{C}$ as a reply.
6. Runs $(m_0, m_1) \leftarrow \mathcal{A}_\lambda(\mathsf{pp}_e, \mathsf{pp}_d)$ where $\mathsf{pp}_e := \mathsf{pk}$, $\mathsf{pp}_d := \widetilde{C}$.
7. Picks $b \xleftarrow{\$} \{0,1\}$ uniformly at random and computes $c^* \leftarrow \mathsf{Enc}(\mathsf{pk}, x, m_b)$.
8. Runs $b' \leftarrow \mathcal{A}_\lambda(c^*)$.
9. Returns the truth value of $b = b'$.

If $\widetilde{C} \leftarrow i\mathcal{O}(1^\lambda, C_0)$, then $\mathcal{D}$ simulates G_0, otherwise it simulates G_1. Moreover, both circuits have the same input/output behaviour. Circuit $C_{\mathsf{sk}^*,x}$ can potentially differ from C_{sk} only on inputs where $t = x$ and in this case $C_{\mathsf{sk}^*,x}$ outputs $\perp$. However, $\mathcal{A}$ has to provide $x \notin L$ and that means for $t = x$, that $R(t, w) = 0$ and hence C_{sk} outputs $\perp$ too. Thus, it must hold

$$|\Pr[\mathsf{G}_0 = 1] - \Pr[\mathsf{G}_1 = 1]| = |\Pr[\mathcal{D}(i\mathcal{O}(C_0)) = 1] - \Pr[\mathcal{D}(i\mathcal{O}(C_1)) = 1]|\,.$$

Hence, we can conclude that $|\Pr[\mathsf{G}_0 = 1] - \Pr[\mathsf{G}_1 = 1]| = \mathbf{Adv}^{i\mathcal{O}}_{\mathcal{D}}$ as required.

Now we show that we can construct an adversary against the security of the PE scheme.

Lemma 3.5.2. *From any PPT adversary $\mathcal{A} = \{\mathcal{A}_\lambda\}_{\lambda\in\mathbb{N}}$ we can construct a PPT adversary $\mathcal{B} = \{\mathcal{B}_\lambda\}_{\lambda\in\mathbb{N}}$ such that*

$$\mathbf{Adv}^{\mathtt{PE}}_{\mathcal{B}} = \left|\Pr[\mathsf{G}_1 = 1] - \frac{1}{2}\right|.$$

We construct an adversary $\mathcal{B} = \{\mathcal{B}_\lambda\}_{\lambda\in\mathbb{N}}$ which breaks the security of the puncturable encryption scheme.

The adversary $\mathcal{B}_\lambda$:

1. Runs $x \leftarrow \mathcal{A}_\lambda$ and sends x to its challenger.
2. Receives public key pk and punctured secret key sk^* from the challenger.
3. Constructs obfuscation of the circuit $C_{\mathsf{sk}^*,x}$ and runs $(m_0, m_1) \leftarrow \mathcal{A}_\lambda(\mathsf{pp}_e, \mathsf{pp}_d)$ where $\mathsf{pp}_e := \mathsf{pk}$, $\mathsf{pp}_d := \widetilde{C}_{\mathsf{sk}^*,x}$.
4. Outputs (m_0, m_1) to the challenger and obtains ciphertext c^* as a response.
5. Returns $b' \leftarrow \mathcal{A}_\lambda(c^*)$.

It is easy to see that $\mathcal{B}$ perfectly simulates G_1. Hence, we have

$$\mathbf{Adv}^{\mathtt{PE}}_{\mathcal{B}} = \left|\Pr[\mathsf{G}_1 = 1] - \frac{1}{2}\right|.$$

Combining Lemmas 3.5.1 and 3.5.2 we obtain following:

$$\begin{aligned}
\mathbf{Adv}^{\mathtt{OWE}}_{\mathcal{A}} &= \left|\Pr[\mathsf{G}_0 = 1] - \frac{1}{2}\right| \\
&\leq |\Pr[\mathsf{G}_0 = 1] - \Pr[\mathsf{G}_1 = 1]| + \left|\Pr[\mathsf{G}_1 = 1] - \frac{1}{2}\right| \\
&= \mathbf{Adv}^{i\mathcal{O}}_{\mathcal{D}} + \mathbf{Adv}^{\mathtt{PE}}_{\mathcal{B}},
\end{aligned}$$

which concludes our proof. □

3.5.2 Construction of EOWE

Our construction of EOWE is similar to the construction of OWE. It also relies on puncturable encryption, but the principal difference is that in EOWE we use $e\mathcal{O}$ instead of $i\mathcal{O}$. Let $\mathtt{PE} = (\mathtt{PE.KeyGen}, \mathtt{PE.Enc}, \mathtt{PE.Punct}, \mathtt{PE.Dec})$ be a puncturable encryption and $e\mathcal{O}$ an extractability obfuscator for a circuit class $\mathcal{C}_\lambda$. Our construction of extractable offline witness encryption $\mathtt{EOWE} = (\mathsf{Setup}, \mathsf{Enc}, \mathsf{Dec})$ for a language L is given in Figure 3.8. We assume that the decryption circuit is padded to the maximum length of sizes of all circuits appearing in the security proof, hence, all circuits have the same size.

$\underline{\mathsf{Setup}(1^\lambda)}$	$\underline{C_{\mathsf{sk}}(c, w)}$
$(\mathsf{sk}, \mathsf{pk}) \leftarrow \mathtt{PE.Gen}(1^\lambda)$	Parse c as (c_{pe}, t)
$\widetilde{C}_{\mathsf{sk}} \leftarrow e\mathcal{O}(1^\lambda, C_{\mathsf{sk}})$	if $R(t, w) = 1$
$\mathsf{pp}_e := \mathsf{pk},\ \mathsf{pp}_d := \widetilde{C}_{\mathsf{sk}}$	$\quad m \leftarrow \mathtt{PE.Dec}(\mathsf{sk}, c_{pe}, t)$
return $(\mathsf{pp}_e, \mathsf{pp}_d)$	$\quad$ return m
	return $\perp$
$\underline{\mathsf{Enc}(\mathsf{pp}_e, x, m)}$	$\underline{\mathsf{Dec}(\mathsf{pp}_d, c, w)}$
$c_{pe} \leftarrow \mathtt{PE.Enc}(\mathsf{pp}_e, m, x)$	return $m \leftarrow \widetilde{C}_{\mathsf{sk}}(c, w)$
return $c := (c_{pe}, x)$	

Figure 3.8: Construction of EOFWE

Theorem 3.5.2. *Let* $\mathtt{PE} = (\mathtt{PE.Gen}, \mathtt{PE.Enc}, \mathtt{PE.Punct}, \mathtt{PE.Dec})$ *be a secure one-time puncturable tag-based encryption and* $e\mathcal{O}$ *be a secure extractability obfuscator. Then* $(\mathsf{Setup}, \mathsf{Enc}, \mathsf{Dec})$ *defined in Figure 3.8 is semi-adaptively secure extractable offline witness encryption.*

Proof. Correctness of the scheme is implied by the correctness of the puncturable encryption scheme and the extractability obfuscator. To prove security we define a series of games $\mathsf{G}_0 - \mathsf{G}_1$. The individual games differ in how we realize our setup algorithm and decryption circuit. For $i \in \{0, 1\}$ we denote by $\mathsf{G}_i(1^\lambda, x) = 1$ the event that the adversary $\mathcal{A} = \{\mathcal{A}_\lambda\}_{\lambda \in \mathbb{N}}$ outputs b' in game $\mathsf{G}_i(1^\lambda, x)$ such that $b' = b$.

Game 0. Game G_0 (Figure 3.9) corresponds to the original security experiment, where we use the Setup, Enc, and C_{sk} directly from our construction.

Next, we define an alternative setup algorithm Setup' in Figure 3.10. This algorithm differs from Setup in that we additionally puncture the secret key sk on the challenge tag x.

$\underline{\mathsf{G}_0(1^\lambda, x)}$	$\underline{C_{\mathsf{sk}}(c, w)}$
$(\mathsf{pp}_e, \mathsf{pp}_d) \leftarrow \mathsf{Setup}(1^\lambda)$	Parse c as (c_{pe}, t)
$(m_0, m_1) \leftarrow \mathcal{A}_\lambda(\mathsf{pp}_e, \mathsf{pp}_d, x)$	if $R(t, w) = 1$
$b \xleftarrow{\$} \{0, 1\}$	$m \leftarrow \mathtt{PE}.\mathsf{Dec}(\mathsf{sk}, c_{pe}, t)$
$c^* \leftarrow \mathsf{Enc}(\mathsf{pp}_e, x, m_b)$	return m
$b' \leftarrow \mathcal{A}_\lambda(c^*)$	return $\perp$
return $b' = b$	

Figure 3.9: Game G_0

$\underline{\mathsf{Setup}'(1^\lambda, x)}$
$(\mathsf{sk}, \mathsf{pk}) \leftarrow \mathtt{PE}.\mathsf{Gen}(1^\lambda)$
$\mathsf{sk}^* \leftarrow \mathtt{PE}.\mathsf{Punct}(\mathsf{sk}, x)$
$\widetilde{C}_{\mathsf{sk}^*,x} \leftarrow e\mathcal{O}(1^\lambda, C_{\mathsf{sk}^*,x})$
$\mathsf{pp}_e := \mathsf{pk},\ \mathsf{pp}_d := \widetilde{C}_{(\mathsf{sk}^*,x)}$
return $(\mathsf{pp}_e, \mathsf{pp}_d)$

Figure 3.10: Alternative setup

Game 1. In G_1 (Figure 3.11) we now run our alternative setup algorithm Setup', which punctures the secret key sk on the tag x. The decryption circuit now uses the punctured key sk^*.

$\underline{\mathsf{G}_2(1^\lambda, x)}$	$\underline{C_{\mathsf{sk}^*,x}(c, w)}$
$\boxed{(\mathsf{pp}_e, \mathsf{pp}_d) \leftarrow \mathsf{Setup}'(1^\lambda, x)}$	Parse c as (c_{pe}, t)
$(m_0, m_1) \leftarrow \mathcal{A}_\lambda(\mathsf{pp}_e, \mathsf{pp}_d, x)$	if $R(t, w) = 1$
$b \xleftarrow{\$} \{0, 1\}$	if $x = t$
$c^* \leftarrow \mathsf{Enc}(\mathsf{pp}_e, x, m_b)$	return $\perp$
$b' \leftarrow \mathcal{A}_\lambda(c^*)$	$\boxed{m \leftarrow \mathtt{PE}.\mathsf{PDec}(\mathsf{sk}^*, c_{pe}, t)}$
return $b' = b$	return m
	return $\perp$

Figure 3.11: Game G_1

Lemma 3.5.3. *For every PPT adversary $\mathcal{A} = \{\mathcal{A}_\lambda\}_{\lambda\in\mathbb{N}}$, for every polynomial $\mathsf{p}(\lambda)$ there exist a PPT extractor $\mathcal{E} = \{\mathcal{E}_\lambda\}_{\lambda\in\mathbb{N}}$ and polynomial $\mathsf{q}(\lambda)$ such that for*

every $\lambda \in \mathbb{N}$ *and for all* $x \in \{0,1\}^*$ *holds*

$$\Big|\Pr[\mathsf{G}_0(1^\lambda, x) = 1] - \Pr[\mathsf{G}_1(1^\lambda, x) = 1]\Big| \geq \frac{1}{\mathsf{p}(\lambda)}$$
$$\implies \Pr[(x, w) \in \mathcal{R} : w \leftarrow \mathcal{E}_\lambda(x)] \geq \frac{1}{\mathsf{q}(\lambda)}.$$

Assume there exist an adversary $\mathcal{A} = \{\mathcal{A}_\lambda\}_{\lambda\in\mathbb{N}}$ and polynomial $\mathsf{p}(\lambda)$ for which holds

$$\Big|\Pr[\mathsf{G}_0(1^\lambda, x) = 1] - \Pr[\mathsf{G}_1(1^\lambda, x) = 1]\Big| \geq \frac{1}{\mathsf{p}(\lambda)}.$$

The games differ only in the circuit that is obfuscated. Thus, we can use these to construct a distinguisher $\mathcal{D} = \{\mathcal{D}_\lambda\}_{\lambda\in\mathbb{N}}$ to distinguish the obfuscations of these circuits. First, we show how to construct an efficient sampler for our circuits.

The circuit sampler $S(1^\lambda, x)$:

1. Generates $(\mathsf{sk}, \mathsf{pk}) \leftarrow \mathtt{PE.Gen}(1^\lambda)$ and runs $\mathsf{sk}^* \leftarrow \mathtt{PE.Punct}(\mathsf{sk}, x)$.
2. Constructs $C_{\mathsf{sk}}, C_{\mathsf{sk}^*,x}$.
3. Sets $\mathsf{aux} := (\mathsf{pk}, x)$.
4. Returns $(C_{\mathsf{sk}}, C_{\mathsf{sk},x}, \mathsf{aux})$.

Now we construct a distinguisher $\mathcal{D} = \{\mathcal{D}_\lambda\}_{\lambda\in\mathbb{N}}$ as follows.

The distinguisher $\mathcal{D}_\lambda(\widetilde{C}, C_{\mathsf{sk}}, C_{\mathsf{sk}^*,x}, \mathsf{aux})$:

1. Parses aux as (pk, x).
2. Runs $(m_0, m_1) \leftarrow \mathcal{A}_\lambda(\mathsf{pk}, \widetilde{C}, x)$.
3. Randomly picks $b \xleftarrow{\$} \{0,1\}$ and computes $c^* \xleftarrow{\$} \mathsf{Enc}(\mathsf{pk}, x, m_b)$.
4. Runs $b' \leftarrow \mathcal{A}_\lambda(c^*)$.
5. Returns the truth value of $b = b'$.

If $\widetilde{C}$ is an obfuscation of C_{sk}, $\mathcal{D}$ perfectly simulates G_0, whereas if $\widetilde{C}$ is an obfuscation of $C_{\mathsf{sk}^*,x}$, it perfectly simulates G_1. Hence, $\mathcal{D}$ can distinguish the obfuscated circuits with probability at least $\frac{1}{\mathsf{p}(\lambda)}$. The security of $e\mathcal{O}$ guarantees that there is an extractor $\mathcal{E}$ which extracts an input on which the given circuits differ with probability at least $\frac{1}{\mathsf{q}(\lambda)}$. Notice that the decryption circuits produce different outputs only in the case where tag t is equal to our challenge x and $R(t, w) = 1$. This is guaranteed by correctness of puncturable tag-based encryption scheme. Hence, the extractor $\mathcal{E}$ returns $(c, w) = ((c_{pe}, x), w)$ such that $R(x, w) = 1$,

which means we obtain a valid witness for our instance x with probability at least $\frac{1}{\mathsf{q}(\lambda)}$. Therefore, we obtain required extractor $\mathcal{E}$ from Lemma 3.5.3.

Now we show that we can construct an adversary against the security of the PE scheme.

Lemma 3.5.4. *From any PPT adversary $\mathcal{A} = \{\mathcal{A}_\lambda\}_{\lambda\in\mathbb{N}}$, for any $x \in \{0,1\}^*$ we can construct a PPT adversary $\mathcal{B} = \{\mathcal{B}_\lambda\}_{\lambda\in\mathbb{N}}$ such that*

$$\mathbf{Adv}^{\mathtt{PE}}_{\mathcal{B}} = \left|\Pr[\mathsf{G}_1(1^\lambda, x) = 1] - \frac{1}{2}\right|.$$

This lemma is proven in the same way as Lemma 3.5.2. For any $x \in \{0,1\}^*$, we construct an adversary $\mathcal{B} = \{\mathcal{B}_\lambda\}_{\lambda\in\mathbb{N}}$ which breaks the security of the puncturable encryption scheme.

The adversary $\mathcal{B}_\lambda$:

1. Sends x to its challenger.
2. Receives public key pk and punctured secret key sk^* from the challenger.
3. Constructs obfuscation of the circuit $C_{\mathsf{sk}^*,x}$ and runs $(m_0, m_1) \leftarrow \mathcal{A}_\lambda(\mathsf{pp}_e, \mathsf{pp}_d)$ where $\mathsf{pp}_e := \mathsf{pk}$, $\mathsf{pp}_d := \widetilde{C}_{\mathsf{sk}^*,x}$.
4. Outputs (m_0, m_1) to the challenger and obtains ciphertext c^* as a response.
5. Returns $b' \leftarrow \mathcal{A}_\lambda(c^*)$.

It is easy to see that $\mathcal{B}$ perfectly simulates G_1. Hence, we have

$$\mathbf{Adv}^{\mathtt{PE}}_{\mathcal{B}} = \left|\Pr[\mathsf{G}_1(1^\lambda, x) = 1] - \frac{1}{2}\right|.$$

Combining Lemmas 3.5.3 and 3.5.4 we obtain following:

$$\begin{aligned}
\left|\Pr[\mathsf{ExpEOWE}_{\mathcal{A}}(1^\lambda, x) = 1] - \frac{1}{2}\right| &= \left|\Pr[\mathsf{G}_0 = 1] - \frac{1}{2}\right| \\
= \left|\Pr[\mathsf{G}_0(1^\lambda, x) = 1] - \Pr[\mathsf{G}_1(1^\lambda, x) = 1] + \Pr[\mathsf{G}_1(1^\lambda, x) = 1] - \frac{1}{2}\right| \\
\leq \left|\Pr[\mathsf{G}_0(1^\lambda, x) = 1] - \Pr[\mathsf{G}_1(1^\lambda, x) = 1]\right| + \left|\Pr[\mathsf{G}_1(1^\lambda, x) = 1] - \frac{1}{2}\right| \\
= \left|\Pr[\mathsf{G}_0(1^\lambda, x) = 1] - \Pr[\mathsf{G}_1(1^\lambda, x) = 1]\right| + \mathbf{Adv}^{\mathtt{PE}}_{\mathcal{B}}
\end{aligned}$$

Assuming that

$$\left|\Pr[\mathsf{ExpEOWE}_{\mathcal{A}}(1^\lambda, x) = 1] - \frac{1}{2}\right| \geq \frac{1}{\mathsf{p}(\lambda)}$$

we obtain:

$$\Big|\Pr[\mathsf{G}_0(1^\lambda, x) = 1] - \Pr[\mathsf{G}_1(1^\lambda, x) = 1]\Big| + \mathbf{Adv}^{\mathtt{PE}}_{\mathcal{B}} \geq \frac{1}{\mathsf{p}(\lambda)}$$
$$\iff$$
$$\Big|\Pr[\mathsf{G}_0(1^\lambda, x) = 1] - \Pr[\mathsf{G}_1(1^\lambda, x) = 1]\Big| \geq \frac{1}{\mathsf{p}(\lambda)} - \mathbf{Adv}^{\mathtt{PE}}_{\mathcal{B}}.$$

Because $\mathbf{Adv}^{\mathtt{PE}}_{\mathcal{B}}$ is a negligible function, there exists a polynomial $\mathsf{p}'(\lambda)$ such that $\frac{1}{\mathsf{p}(\lambda)} - \mathbf{Adv}^{\mathtt{PE}}_{\mathcal{B}} \geq \frac{1}{\mathsf{p}'(\lambda)}$. By Lemma 3.5.3, this implies the existence of an extractor $\mathcal{E}$ which outputs a witness w such that $(x, w) \in \mathcal{R}$ with probability at least $\frac{1}{\mathsf{q}(\lambda)}$ for some polynomial $\mathsf{q}(\lambda)$, which concludes our proof. □

Encrypting large messages. In order to encrypt large messages using our (extractable) offline witness encryption, a message must be split into blocks of appropriate size and after that, each block must be encrypted as a separate message. Hence, if a message consists of n blocks, we must produce n ciphertexts. One can avoid this by using (E)OWE as a Key Encapsulation Mechanism (KEM) and encrypt a random key κ for a block cipher. Then we can encrypt an arbitrary message using the block cipher with the key κ, which is our Data Encapsulation Mechanism (DEM). This results in a final ciphertext size of one OWE ciphertext and n DEM ciphertext blocks.

(Extractable) Offline Functional Witness Encryption. Abusalah *et al.* [AFP16] have constructed Offline Functional Witness Encryption (OFWE) which additionally encrypts a function F from some supported function family with a message. Anyone who knows a witness w for a given instance x learns the value $F(m, w)$. Security of OFWE guarantees indistinguishability of encryptions of $(x, (m_0, F_0))$ and $(x, (m_1, F_1))$ as long as either $x \notin L$ or for all w such that $(x, w) \in \mathcal{R}$ holds that $F_0(m_0, w_0) = F_1(m_1, w_1)$. Similarly, extractable OFWE can also be defined which guarantees that if the adversary can distinguish encryptions of $(x, (m_0, F_0))$ and $(x, (m_1, F_1))$, then it must know w such that $(x, w) \in \mathcal{R}$ and at the same time $F_0(m_0, w) \neq F_1(m_1, w)$. Our generic constructions, can be used to build these two primitives by puncturing on the tuple $(x, (m_0, F_0), (m_1, F_1))$. However, this requires to know all these values in advance and therefore we are able to prove only selective security. Proofs for OFWE and EOFWE are direct adaptations of the corresponding proofs for OWE and EOWE, respectively.

Next, we discuss an instantiation of our scheme and compare it with other constructions. Mainly we focus on instantiating a one-time puncturable tag-based encryption scheme because all other schemes are based on obfuscation and more-

over there has not been so much progress in building practical obfuscation, as compared to that of encryption schemes.

3.6 Instantiation of Puncturable Encryption Scheme

As one might expect, standard puncturable encryption schemes [DGJ+18, GM15, GHJL17], that allows repeatedly puncturing of the secret key, are not very efficient and therefore their deployment in our framework would harm the overall efficiency of our constructions. Since tag-based encryption schemes require similar "all-but-one" decryption in order to prove security, it is reasonable to assume that tag-based encryption schemes can be turned into one-time puncturable encryption schemes. And that is exactly the approach which we take. Concretely, we show that tag-based encryption due to Kiltz [Kil06] can be modified to yield one-time PE. The advantage of this scheme is its efficiency with ciphertext size only 5 group elements. Hence the total ciphertext size of our constructions for short messages is the size of the instance $|x|$ plus 5 group elements. In the case of long messages consisting of n blocks, the ciphertext size is $|x|$ plus $5n$ group elements.

The OWE scheme of Abusalah *et al.* [AFP16] can be instantiated with ElGamal encryption and a type of Groth-Sahai's NIZK proof which leads to ciphertext size of 32 group elements plus $|x|$. For messages consisting of n blocks the construction in [AFP16] requires ciphertext of size $24 + 8n$ group elements plus $|x|$. We believe that one can adjust the construction by replacing $i\mathcal{O}$ with $e\mathcal{O}$ and obtain EOWE. However, both of these constructions would achieve only selective security.

There are two other constructions of OWE [PD20, PD19] which have appeared only recently. In [PD19] only the constructions of OWE and OFWE are given and it is not clear if they can be adjusted to the extractable setting. Both constructions are based on Naor-Yung style encryption in a similar manner as [AFP16], which requires encrypting the given message twice using IND-CPA-secure PKE. But instead of a non-interactive zero-knowledge proof, the extractable witness PRF is used, whose known constructions rely on multilinear maps. Additionally, the constructions require a randomized encoding scheme with sub-exponential security. The resulting schemes achieve only selective security and by instantiating IND-CPA-secure PKE with ElGamal Encryption one can obtain the ciphertext of size $|x| + |y| + 4$ group elements, where y is the output of extractable witness PRF and $|y|$ can be bounded by a constant that is independent of the PKE scheme. For messages consisting of n blocks the ciphertext size is $|x| + |y|$ plus $4n$ group elements.

The second work [PD19] constructs OWE, EOWE, OFWE and EOFWE. All constructions follow the same template and rely either on $i\mathcal{O}$ or $e\mathcal{O}$ which are combined with private-key encryption, a pseudorandom generator (PRG), and a

puncturable witness PRF (pWPRF). We note that currently known constructions of pWPRFs are based on obfuscation. The ciphertext size even for long messages is $|x|+|r|+|c|$ where r is the output of the PRG and c is the ciphertext of the private-key encryption scheme. The main disadvantage of this approach is expensive encryption which requires executing the evaluation algorithm of a pWPRF which in other words means running an obfuscated circuit.

3.6.1 PE from Kiltz's Tag-Based Encryption Scheme

In this section, we describe an efficient one-time puncturable tag-based encryption scheme based on the tag-based encryption scheme of Kiltz [Kil06]. The scheme is given in Figure 3.12. Specifically, the $\mathsf{Gen}, \mathsf{Enc}$ algorithms of both schemes are identical. Dec algorithm of Kiltz's scheme contains an implicit test if a ciphertext is consistent with a tag t. In our scheme, we execute this test explicitly to ensure the required correctness properties. Hence, our Dec is roughly the alternative way of decryption that is described in the correctness section of Kiltz's tag-based encryption. The only thing left is to specify Punct and PDec algorithms. PDec is similar to the original decryption algorithm proposed by Kiltz but instead of sampling values s_1, s_2 uniformly at random, we construct them in a specific way. That allows us to decrypt messages without explicitly using the values of the original secret key. To enforce correctness properties of a puncturable tag-based encryption scheme, we have to include a consistency check of the validity of a given ciphertext, however, this time without the knowledge of the secret key. This is possible using DDHvf. The construction of values s_1, s_2 is defined in the PDec algorithm.

Similarly to Kiltz's tag-based encryption, the security of our scheme is based on the decision linear assumption (DLin) in a gap group. We remark that our proof is an adaptation of Kiltz's original proof. As we have already mentioned, as a concrete instantiation we can assume a symmetric bilinear group that naturally provides an oracle to decide the DDH problem.

Theorem 3.6.1. *Assume the* DLin *assumption holds. Then* $(\mathsf{Gen}, \mathsf{Punct}, \mathsf{Enc}, \mathsf{Dec}, \mathsf{PDec})$ *defined in Figure 3.12 is a secure one-time puncturable encryption scheme.*

Proof. Correctness properties of the scheme can be verified by simple calculations.

To prove security we define two games G_0 and G_1 and show that these are computationally indistinguishable. For $i \in \{0, 1\}$ we denote by $\mathsf{G}_i = 1$ the event that the adversary $\mathcal{A} = \{\mathcal{A}_\lambda\}_{\lambda\in\mathbb{N}}$ outputs b' in game G_i such that $b = b'$.

Game 0. Game G_0 corresponds to the original security experiment, where we use the Enc algorithm directly from our construction to produce the challenge

$\underline{\mathsf{Gen}(1^\lambda)}$
$g_1 \xleftarrow{\$} \mathbb{G}^*, x_1, x_2 \xleftarrow{\$} \mathbb{Z}_p^*, y_1, y_2 \xleftarrow{\$} \mathbb{Z}_p$
Pick $g_2, g \in \mathbb{G}$ s.t. $g_1^{x_1} = g_2^{x_2} = g$
$u_1 := g_1^{y_1}, u_2 := g_2^{y_2}$
$\mathsf{pk} := (g_1, g_2, u_1, u_2, g), \mathsf{sk} := (x_1, x_2, y_1, y_2)$
return $(\mathsf{pk}, \mathsf{sk})$

$\underline{\mathsf{Enc}(\mathsf{pk}, m, t)}$
$r_1, r_2 \xleftarrow{\$} \mathbb{Z}_p$
$C_1 := g_1^{r_1}, C_2 := g_2^{r_2}$
$D_1 := g^{t \cdot r_1} u_1^{r_1}, D_2 = g^{t \cdot r_2} u_2^{r_2}$
$K := g^{r_1 + r_2}$
$E := K \cdot m$
return $c = (C_1, C_2, D_1, D_2, E)$

$\underline{\mathsf{Dec}(\mathsf{sk}, c, t)}$
Parse c as (C_1, C_2, D_1, D_2, E)
if $(C_1^{t \cdot x_1 + y_1} \neq D_1) \vee (C_2^{t \cdot x_2 + y_2} \neq D_2)$
  retrurn $\perp$
$K := C_1^{x_1} \cdot C_2^{x_2}$
return $m := E \cdot K^{-1}$

$\underline{\mathsf{PDec}(\mathsf{sk}_{t^*}, c, t)}$
Parse c as (C_1, C_2, D_1, D_2, E)
if $\mathsf{DDHvf}(g_1, g^t \cdot u_1, C_1, D_1) = 0$
$\vee \mathsf{DDHvf}(g_2, g^t \cdot u_2, C_2, D_2) = 0$
  return $\perp$
$\Gamma := C_1^{s_1} \cdot C_2^{s_2}, \Delta := D_1 \cdot D_2$
$K := (\Gamma / \Delta)^{1/(t^* - t)}$
return $m := E \cdot K^{-1}$

$\underline{\mathsf{Punct}(\mathsf{sk}, t^*)}$
Parse sk as (x_1, x_2, y_1, y_2)
$s_1 := y_1 + t^* \cdot x_1$
$s_2 := y_2 + t^* \cdot x_2$
return $\mathsf{sk}_{t^*} := (s_1, s_2)$

Figure 3.12: Construction of one-time puncturable tag-based encryption

ciphertext.

Game 1. In G_1 the values C_1, C_2, D_1, D_2 of the challenge ciphertext are computed exactly as in G_0 but we compute value E by sampling $z \xleftarrow{\$} \mathbb{Z}_p$ uniformly at random and computing $E := m \cdot g^z$.

Lemma 3.6.1. *From any PPT adversary $\mathcal{A} = \{\mathcal{A}_\lambda\}_{\lambda \in \mathbb{N}}$ we can construct a PPT adversary $\mathcal{B} = \{\mathcal{B}_\lambda\}_{\lambda \in \mathbb{N}}$ such that*

$$\mathbf{Adv}_{\mathcal{B}}^{\mathsf{DLin}} = \left| \Pr[\mathsf{G}_0 = 1] - \Pr[\mathsf{G}_1 = 1] \right|.$$

To prove this lemma we construct an adversary $\mathcal{B}$ as follows.
The adversary $\mathcal{B}_\lambda(p, \mathbb{G}, \mathsf{DDHvf}, g_1, g_2, g, g_1^x, g_2^y, g^z)$:

1. Runs $\mathcal{A}_\lambda$ to obtain t^*.
2. Randomly samples values $s_1, s_2 \xleftarrow{\$} \mathbb{Z}_p$ and sets $\mathsf{sk}_{t^*} := (s_1, s_2)$.
3. Computes $u_1 := g^{-t^*} \cdot g_1^{s_1}, u_2 := g^{-t^*} \cdot g_2^{s_2}$, set $\mathsf{pk} := (g_1, g_2, u_1, u_2, g)$.

4. Runs $(m_0, m_1) \leftarrow \mathcal{A}_\lambda(\mathsf{pk}, \mathsf{sk}_{t^*})$, samples $b \xleftarrow{\$} \{0,1\}$, and computes the challenge ciphertext as $c = (g_1^x, g_2^y, (g_1^x)^{s_1}, (g_2^y)^{s_2}, m_b \cdot g^z)$.

5. Runs $b' \leftarrow \mathcal{A}_\lambda(c)$.

6. Returns the truth value of $b = b'$.

Notice that pk and sk_{t^*} are distributed correctly. Moreover, if $z = x + y$, then c is indeed a well formed encryption of m_b under pk and hence $\mathcal{B}$ simulates G_0 perfectly. On the other hand if $z \xleftarrow{\$} \mathbb{Z}_p$, then $\mathcal{B}$ simulates G_1 perfectly. Therefore we can conclude that

$$|\Pr[\mathsf{G}_0 = 1] - \Pr[\mathsf{G}_1 = 1]| = \mathbf{Adv}^{\mathsf{DLin}}_{\mathcal{B}}$$

as required.

Lemma 3.6.2. $\Pr[\mathsf{G}_1 = 1] = 1/2$.

Because z is chosen uniformly at random from $\mathbb{Z}_p$, the element g^z is a uniform element of $\mathbb{G}$. Therefore also $m_b \cdot g^z$ is uniformly distributed element which is independent of m_b and hence also independent of b. The other components of ciphertext are clearly independent of m_b. Thus, the adversary gets no information about b and can do no better than guessing. Hence it has a success probability of exactly $\frac{1}{2}$.

By combining Lemmas 3.6.1 and 3.6.2 we obtain following:

$$\begin{aligned}\mathbf{Adv}^{\mathsf{PE}}_{\mathcal{A}} &= \left|\Pr[\mathsf{G}_0 = 1] - \frac{1}{2}\right| = \left|\Pr[\mathsf{G}_0 = 1] - \Pr[\mathsf{G}_1 = 1] + \Pr[\mathsf{G}_1 = 1] - \frac{1}{2}\right| \\ &= \mathbf{Adv}^{\mathsf{DLin}}_{\mathcal{B}},\end{aligned}$$

which concludes the proof. □

3.7 Conclusion and Open Problems

We have constructed the first extractable offline witness encryption scheme which can be used for building a variant of time-lock encryption with a trusted setup. The security achieved by our construction requires that time is fixed in advance before running the setup, however, this seems to be sufficient for many applications. All currently known constructions of an (extractable) offline witness encryption scheme either achieve selective security, where an adversary has to provide an instance and challenge messages before seeing public parameters, or semi-adaptive security, where messages are provided adaptively by an adversary after seeing public parameters and an instance must be known beforehand. Therefore an interesting

open problem is to construct (E)OWE scheme with *full-adaptive* security. Another common denominator of known constructions is that they rely on indistinguishability/extractability obfuscation. It would be interesting to provide some evidence that this is indeed necessary or give a construction that relies on some weaker primitives. One could also focus on more efficient constructions of E(OWE) for some concrete "non-trivial" languages instead of arbitrary languages. Concerning OFWE and EOFWE, all current constructions are only selectively secure. Hence, it seems that novel techniques are needed to achieve even semi-adaptive security for these two primitives. A different option would be to focus purely on OTLE and examine new ways of how to build it.

4 Encryption Schemes from Time-Lock Puzzles

The content of this chapter is based on two independent works which are closely related. Both works focus on encryption schemes which require executing some amount of sequential computation in order to decrypt ciphertexts. At first, we study so-called Timed-Release Encryption (TRE) that we originally introduced in [CJSS20]. After that we discuss the notion of Timed Public-Key Encryption (TPKE) which has been introduced in concurrent work of Katz *et al.* [KLX20].

Author's Contribution. The content of Sections 4.2-4.4 is joint work with Tibor Jager, Daniel Slamanig and Christoph Striecks [CJSS20]. The author has made the following contributions:

- formally defining syntax and security of sequential time-lock puzzles;
- defining the gap sequential squaring assumption and showing its hardness in the Strong Algebraic Group Model;
- constructing and proving the security of sequential time-lock puzzle from the gap sequential squaring assumption in the Random Oracle Model;
- formally defining syntax and security of sequential timed-release encryption;
- proving the security of the basic construction of timed-release encryption;
- constructing a generic sequential timed-release encryption scheme and proving its security;
- noting that the constructions of timed-release encryption and sequential timed-release encryption allow public verifiability and that they can be adjusted to support homomorphic encryption;
- all discussion about practical applications of (sequential) timed-release encryption.

For completeness, we also include sections about integrating functional features into timed-release encryption, discussions about variants of time-lock puzzles, optimal amortized costs of sequential timed-release encryption, and combining sequential timed-release encryption with public servers.

The content of Section 4.5 is joint work with Tibor Jager. At the time of writing the thesis, this work is not finished yet. The author has made the following contributions:

- realizing that the construction of timed-release encryption can be adjusted to yield timed public-key encryption achieving weak IND-CCA security;
- defining timed-release encryption secure against pre-processing;
- defining time-lock puzzles secure against pre-processing;
- proving that the basic timed-release encryption scheme is secure against pre-processing if the underlying time-lock puzzle is secure against pre-processing;
- discussing the generic construction of timed public-key encryption and proving its security;
- constructing timed public-key encryption from the strong sequential squaring assumption and proving its security.

4.1 Introduction, Contributions and Related Work

In this chapter, we focus on the approach which requires performing some kind of sequential computation in order to decrypt a message. This idea was already suggested in 1996 by Rivest *et al.* in [RSW96], who used the term time-lock puzzle (TLP) to denote it. The proposed solution relies on a private-key encryption scheme $(\mathsf{Enc}, \mathsf{Dec})$ and sequential squaring in $\mathbb{Z}_N$ where N is the product of two large primes p and q. Specifically, the puzzle Z is produced in the following way:

- generate randomly a modulus N,
- pick a hardness parameter $T \in \mathbb{N}$ and two values $x, k \xleftarrow{\$} \mathbb{Z}_N$,
- compute values $y := x^{2^T} \bmod N$ and $c_1 := y + k \bmod N$,
- encrypt a message m using k, via $c_2 := \mathsf{Enc}(k, m)$,
- output $Z := (N, T, x, c_1, c_2)$.

To solve the puzzle $Z := (N, T, x, c_1, c_2)$, the following steps are executed:

- compute value $y := x^{2^T} \bmod N$,
- recompute the secret key $k := c_1 - y \bmod N$,
- decrypt the ciphertext $m := \mathsf{Dec}(k, c_2)$.

It is straightforward to see that value y can be computed as the sequence of 2^T repeated squarings - $x^2, (x^2)^2 = x^{2^2}, (x^{2^2})^2 = x^{2^3}, \ldots x^{2^T} \bmod N$. However, the crucial observation is that the knowledge of primes p and q allows the computation of the value y much faster than by sequential squaring and therefore the creation of the puzzle is much more efficient than solving it. Concretely, knowing the factorization of N and thus the order $\varphi(N)$ of the multiplicative group $\mathbb{Z}_N^*$, one can compute $x^{2^T} \bmod N$ by computing at first $t := 2^T \bmod \varphi(N)$. Then it is sufficient to compute $y := x^t \bmod N$. This discrepancy between runtimes of generating and solving the puzzle instance is an important property of TLPs when the required amount of time until the puzzle should be solved is very large. However, constructing time-lock puzzles that have this property seems to be difficult. Until now, there exist only two constructions that have this property. The first one is the already-mentioned proposal of Rivest *et al.* and the second one was suggested by Bitansky *et al.* [BGJ+16] and relies on succinct randomized encodings. The known constructions of succinct randomized encodings are based on indistinguishability obfuscation and therefore this construction is not practical yet. Moreover, Mahmoody *et al.* [MMV11] have shown that it is impossible to construct time-lock puzzles from one-way permutations and collision-resistant hash functions in the random oracle model in black-box way.

Applications of TLPs. Time-lock puzzles have found their place in different applications such as sealed-bid auctions [RSW96], fair contract signing [BN00], zero-knowledge arguments [DN00], and non-malleable commitments [LPS17]. Most of the applications, however, require creating several time-lock puzzles, meaning that it is necessary to execute multiple expensive computations usually in parallel. This is considered to be one of the main drawbacks in deploying TLPs.

Variants and Extensions of Time-Lock Puzzles. In recent years several variants and extensions of TLPs have been proposed to either overcome the difficulty in building them, or improve their practicality, or strengthen their security guarantees.

- *Weak* Time-Lock Puzzles (wTLPs) were proposed by Bitansky *et al.* [BGJ+16] and this variant relaxes the requirement on the time which is needed for generating a puzzle. Concretely, it allows an efficient parallel computation in the generation of the puzzle. wTLPs can be built from randomized encodings.

- Time-Lock Puzzles *with Pre-processing* (ppTLPs) have an additional algorithm which allows executing solution-independent pre-processing whose runtime can be proportional to the hardness parameter T. Similarly to wTLPs, ppTLPs have been introduced by Bitansky *et al.* [BGJ$^+$16] who have constructed them from reusable randomized encodings which in turn can be based on the Learning With Errors (LWE) assumption.

- *Homomorphic* Time-Lock Puzzles (HTLPs) are the first proposal which aims to improve on the applicability of TLPs and overcome the already-mentioned limitation when an application requires solving possibly a large number of puzzles. This variant was introduced in [MT19] by Malavolta and Thyagarajan(MT19 henceforth) and allows for a trusted setup which produces public parameters. The main advantage of the setup is that it allows homomorphic evaluation of some circuit C over a tuple of puzzles $(Z_i)_{i\in[n]}$ which results in a new puzzle Z. The solution s of the resulting puzzle Z has a property that it is equal to the value $C((s_i)_{i\in[n]})$ where $(s_i)_{i\in[n]}$ are corresponding solutions of the puzzles $(Z_i)_{i\in[n]}$. Therefore if an application requires the execution of some computation on solutions of puzzles, then instead of solving all puzzles, it is sufficient to firstly combine all puzzles, and then to solve possibly only one puzzle. MT19 has proposed three constructions for HTLPs - linearly homomorphic TLP (LHTLP) which allows only additions, multiplicatively homomorphic (MHTLP) supports only multiplications, and fully homomorphic (FHTLP) which supports both operations at the same time. All of these constructions are based on the (strong) sequential squaring assumption and the first two are essentially relying on Paillier encryption [Pai99], however, the third one is based on sub-exponentially hard indistinguishability obfuscation. We remark that MT19 provides also constructions that support aggregation of puzzles which might have different hardness parameters. The resulting puzzle is then solvable in time which corresponds to the largest hardness parameter among all aggregated puzzles. An improved construction of a fully homomorphic time-lock puzzle has been later given by Brakerski *et al.* in [BDGM19]. The construction combines a multi-key fully-homomorphic encryption (MK-FHE) scheme [LTV12] with LHTLP. Basically, MK-FHE is used to encrypt a message with a fresh key and an LHTLP to lock the respective MK-FHE secret keys. To ensure compatibility of both primitives, Brakerski *et al.* require an MK-FHE scheme with a linear decryption algorithm.

- *Timed Public-Key Encryption* (TPKE) was proposed by Katz, Loss and Xu [KLX20] and compared to TLPs it has a key generation algorithm which outputs a public/secret key pair and supports two types of decryption - slow

decryption, which requires the execution of an expensive sequential computation in order to decrypt a ciphertext and fast decryption, which is efficient and makes use of the secret key. This primitive is used in [KLX20] as a building block for a non-malleable timed commitment. A timed commitment was originally introduced by Boneh and Naor [BN00] and it allows to prove that a commitment is well-formed, moreover one can prove that the given value is indeed the committed value and additionally it has a force open algorithm which is able to open commitment by executing a sequential computation.

Contributions. The work in this chapter discusses two aspects of time-lock puzzles: broadening their applicability and achieving some form of non-malleability. At first, we aim for improving the practicality of time-lock puzzles and in particular, we try to address the problem that in many scenarios several independent puzzles have to be solved in parallel. Therefore our approach can be seen as an alternative to the approach of MT19. However, it is more convenient to have an approach that supports the decryption of many messages by solving only one puzzle, the so-called *"solve one, get many for free"* property. Therefore we propose a new primitive denoted as Timed-Release Encryption (TRE) which similarly to HTLP has a trusted setup. We stress that we require from TRE very different functionality compared to TRE schemes based on trusted agents. Specifically, we require that the setup of TRE outputs public encryption and public decryption parameters and it must run in time which is only logarithmic in the hardness parameter T. Messages are then encrypted using the encryption algorithm which takes as input public encryption parameters and its runtime is independent of T. One is able to decrypt all messages encrypted using the same public parameters by performing an expensive sequential computation which is determined by the corresponding public decryption parameters. The runtime of decryption should be determined by the hardness parameter T. Clearly, this primitive provides the desirable *"solve one, get many for free"* property.

We show how to construct such a TRE scheme in a generic way, without relying on the algebraic structure of the underlying sequential problem. Our construction is rather simple and combines only two primitives - a public-key encryption scheme (PKE) and a time-lock puzzle. However, we should stress that our definition of a TLP slightly deviates from the original definition given by Rivest *et al.* [RSW96]. In this thesis, we abstract from the fact that TLPs are used to encrypt messages. Hence, for us a TLP provides core functionality of a puzzle that needs a certain amount of time to be solved, without considering any messages. To build TRE, we generate a puzzle Z with corresponding solution s and use this solution as random coins for generating a public/secret key pair of PKE. The public key and puzzle Z are the output of the trusted setup and hence they represent our public

encryption and public decryption parameters, respectively. Messages are then encrypted by simply running the encryption algorithm of a PKE scheme using a public key. To decrypt a ciphertext it is sufficient to solve the puzzle Z once and then use the solution s to compute the corresponding secret key. Note that when a PKE scheme in our construction is a partially homomorphic encryption scheme, e.g., ElGamal [ElG84], or a fully homomorphic encryption scheme, e.g., BGV [BGV12], this immediately yields (fully) homomorphic TRE (HTRE). Therefore, in this way we are able to obtain the *"solve one, get many for free"* property for both, the result of a homomorphic evaluation of many ciphertexts, but also if we want to decrypt all ciphertexts individually. Moreover, our HTRE scheme fulfils the basic definition of HTLPs from MT19, where the time required to solve the puzzles starts with the generation of the parameters. Hence, our construction can be viewed as the first generic construction for basic HTLPs. The alternative variant of HTLPs proposed in MT19 is HTLPs with a reusable setup, where the parameters generated by setup can be used to produce many puzzles in a way that for every single puzzle the time only starts to run from the point where the puzzle is generated (this characteristic is also inherited by [BDGM19]). However, there is no need to use such an HTLP in any of the applications discussed in [MT19], on the contrary, in some applications, it is even more desirable when the runtime of the puzzle is counted from the point of running the setup algorithm.

To further mitigate the negative impacts of executing an expensive computation from an economic and an ecological perspective, we propose a variant of TLP which takes as input an increasing sequence of hardness parameters $(T_i)_{i\in[n]}$ and produces a set of puzzles and solutions $(Z_i, s_i)_{i\in[n]}$. The crucial property which we require is that the knowledge of the solution s_{i-1} of the puzzle Z_{i-1} enables solving the puzzle Z_i in time which is determined by hardness $T_i - T_{i-1}$. This simply means, there is no need to start from the beginning to solve the following puzzles. We call a TLP with this property a *sequential* Time-Lock Puzzle (sTLP) and we use it as a building block for sequential Timed-Release Encryption (sTRE) that fulfils a similar property. The advantage of sTRE is the possibility of outsourcing the task of solving puzzles to an external entity that can perform an expensive computation instead of letting each receiver perform this wasteful computation. We note that providing a useful security definition for an sTLP, which is sufficient for building an sTRE scheme, is not so trivial. We provide a detailed explanation of why this is the case later when we formally define sequential TLPs. To instantiate an sTLP we introduce the gap sequential squaring assumption, which similarly to other gap assumptions [OP01] states that computational variant of sequential squaring assumption is hard even given an oracle which takes as input a hardness parameter T' and a value y' and outputs 1 if and only if $y' = x^{2^{T'}} \bmod N$. As evidence for the hardness of this assumption, we provide an analysis in the strong algebraic group

model (SAGM) and in particular, we show that our assumption holds as long as factoring is hard. We observe that our construction of (sequential) timed-release encryption has the interesting property of public verifiability. Specifically, given a candidate solution with respect to public decryption parameters, one can efficiently check if the given solution is correct, by generating a public/secret key pair of the underlying PKE scheme using the solution and comparing if the generated public key matches the public encryption parameters of (s)TRE. This property is particularly useful when the task of solving puzzles is outsourced.

Finally, we discuss the possibilities of achieving the non-malleability of time-locked messages. One can observe that our definition of TRE has many similarities with the concurrently introduced notion of timed public-key encryption. The security of TPKE is defined in a similar manner as IND-CCA security of PKE and it requires that the time needed for decryption of some ciphertext starts to run from the point when the ciphertext is created. We observe that if one relaxes this requirement and allows that the time required for decryption starts with the generation of the public/secret key pair, then our TRE construction can be adjusted to yield a TPKE scheme. Recall that the setup of our TRE computes a public/secret key pair by running the generation algorithm of some PKE scheme using the solution of a puzzle Z as random coins. Hence, if the setup additionally outputs the secret key, this can be used for the fast decryption of TPKE. In some scenarios, this relaxed security notion does not need to be sufficient and therefore we propose two constructions that satisfy the original security notion and are more efficient than the construction of Katz *et al.* [KLX20] (to the best of our knowledge this is currently the only known construction). The mentioned inefficiency stems from the fact that encryption of a single message takes roughly time T, where T is the hardness parameter. Therefore encryption and slow decryption have similar running times which is not practical for most applications. Our first construction is generic and allows us to encrypt messages exponentially faster than running the slow decryption algorithm. We remark that the construction in [KLX20] is not generic and is based on the strong sequential squaring assumption. Our second construction relies on the strong sequential squaring problem too however the runtime of encryption is independent of the hardness parameter.

Applications of (s)TRE. The crucial observation made in MT19 is that many applications admit a trusted setup. In this way, an HTLP can be used even in scenarios that require a rather large number of time-locked messages. Specifically, MT19 present applications to e-voting, multi-party coin flipping as well as multi-party contract signing, or more recently verifiable timed signatures [TBM+20], again yielding several further interesting applications. As we explain in detail later, our timed-release encryption can be used as a replacement of HTLPs in all

of these applications which results in some efficiency improvements and therefore can be viewed as a preferable solution. The main disadvantage of HTLPs compared to TRE is that one must wait until all puzzles of interest are available, then execute the homomorphic evaluation to produce a resulting puzzle: only after that can the resulting puzzle be solved. For instance, if we consider an example of e-voting where the voting phase takes some fixed time T, then only after the voting phase is over can one start with solving a puzzle which takes additional time T. On the other hand, with TRE we can start executing the sequential computation immediately after the setup is performed and hence the results can be available at the end of the voting phase. Moreover, we are able to use TRE also in applications that require to have access to all time-locked messages in full and not only to the result of a homomorphic evaluation. As a novel application, we combine (s)TRE with Functional Encryption which allows us to time-lock functional keys.

Related Work. In addition to the already-mentioned works regarding TLPs, HTLPs, TPKE, and timed commitments, there are several concurrent works that have recently appeared. The work of Ephraim *et al.* [EFKP20] focuses on concurrent non-malleable time-lock puzzles. The proposed construction in [EFKP20] is efficient however it relies on the auxiliary-input random oracle model. The basic idea of their construction is as follows. The random oracle is evaluated on the string $s||r$ where s is the solution and r is some randomness. The output of the random oracle is used as random coins for the puzzle generation algorithm of any TLP. Moreover, they introduce also so-called public verifiable time-lock puzzles, which have the property that everyone is able to efficiently verify if the given solution is a valid solution of the puzzle and even the case that a puzzle does not have any solution is efficiently verifiable. In this way, publicly verifiable TLPs can be seen as an intermediate notion between standard TLPs and timed commitments. We remark that our construction of TRE attains a similar notion of public verifiability.

Non-malleable TLPs have also been constructed in the plain model by Dachman *et al.* [DKP20], however, the construction is not very efficient. The main objective of the work is a construction of non-malleable codes and non-malleable TLPs are obtained only as a by-product.

Burdges *et al.* [BD20] have introduced the notion of Delay Encryption (DE). Delay Encryption is defined as a Key Encapsulation Mechanism (KEM), however, this choice has no consequences for comparison with our TRE. Similarly to TRE, DE has a trusted setup algorithm that takes as input the hardness parameter T. However, the runtime of the setup of a DE scheme can be polynomial in the hardness parameter, while in TRE we only allow setup algorithms with a runtime that is logarithmic in the hardness parameter. DE provides efficient encapsulation

and decapsulation algorithms (their runtime is independent of T) with respect to arbitrary identities (similarly to Identity-Based Encryption) and a slow extraction algorithm whose runtime is determined by the hardness parameter. The construction of Burdges *et al.* [BD20] is based on isogenies.

Finally, we discuss *Multi-Instance* Time-Lock Puzzles (MITLP) proposed by Abadi and Kiayias [AK21] which are similar to our notion of sequential timed-release encryption. MITLP allows to generated puzzles with respect to consecutive multiples of one hardness parameter T in a way that all puzzles can be solved sequentially without a need to perform parallel computation. Concretely, a multi-instance time-lock puzzle has a setup algorithm that takes as input the hardness parameter T and some positive integer n. The setup outputs a public/secret key pair. Both the public and the secret key are used to generate n puzzles which correspond to hardness parameters $T, 2T, \ldots, nT$. One can solve all puzzles sequentially using the public key. Abadi *et al.* construct this primitive generically from any time-lock puzzle and a commitment scheme and the construction also provides public verifiability. The main idea is chaining the generated puzzles where the message which is supposed to be decrypted as the last is encrypted first and then the information which is needed for decrypting that message is included in the ciphertext which should be decrypted as penultimate. This process is repeated until all messages are encrypted. This, however, requires that all messages of interest must be known at the time when MITLP is generated. This is in contrast to our sTRE which allows encrypting an arbitrary number of messages with respect to different hardness parameters at any time. Moreover, we support an arbitrary increasing sequence of hardness parameters as opposed to only consecutive multiples of one hardness parameter. In MITLP, the time needed for decryption is counted from the point of generating ciphertexts where in sTRE the time is counted from the point of running the setup. Additionally, sTRE uses only public parameters for encryption.

4.2 Time Lock-Puzzles

In this section, we state a new definition for time-lock puzzles and explain how it relates to the old definition. Afterwards, we give a brief overview of other variants of time-lock puzzles and discuss possible instantiations of time-lock puzzles that fulfil our new definition.

Definition 4.2.1. A time-lock puzzle is pair of algorithms $\mathtt{TLP} = (\mathsf{Gen}, \mathsf{Solve})$ with the following syntax.

- $(Z, s) \leftarrow \mathsf{Gen}(1^\lambda, T)$ is a probabilistic algorithm that takes as input the security parameter 1^λ and a hardness parameter $T \in \mathbb{N}$ and outputs a puzzle Z

$\underline{\mathsf{ExpTLP}^b_{\mathcal{A}}(\lambda)\text{:}}$
$(Z, s) \leftarrow \mathsf{Gen}(1^\lambda, T(\lambda))$
if $b = 0 : c := s$
if $b = 1 : c \xleftarrow{\$} S$
return $b' \leftarrow \mathcal{A}_\lambda(Z, c)$

Figure 4.1: Security experiment for time-lock puzzles.

together with the unique solution s of the puzzle. We require that Gen runs in time at most $\mathsf{poly}(\log T, \lambda)$ for some polynomial poly.

- $s \leftarrow \mathsf{Solve}(Z)$ is a deterministic algorithm that takes as input a puzzle Z and outputs a solution $s \in S$, where S is a finite set. We require that Solve runs in time at most $T \cdot \mathsf{poly}(\lambda)$. There is also a lower bound on the running time, which is part of the security definition.

We say $\mathtt{TLP}$ is *correct* if for all $\lambda, T \in \mathbb{N}$ it holds

$$\Pr[s = s' : (Z, s) \leftarrow \mathsf{Gen}(1^\lambda, T), s' \leftarrow \mathsf{Solve}(Z)] = 1.$$

For security we require that the solution of a TLP is indistinguishable from random, unless the adversary has sufficient running time to solve the puzzle.

Definition 4.2.2. Consider the security experiment $\mathsf{ExpTLP}^b_{\mathcal{A}}(1^\lambda)$ in Figure 4.1. We say that a time-lock puzzle $\mathtt{TLP}$ is *secure with gap* $0 < \epsilon < 1$, if there exists a polynomial $\tilde{T}(\cdot)$ such that for all polynomials $T(\cdot) \geq \tilde{T}(\cdot)$ and every polynomial-size adversary $\mathcal{A} = \{\mathcal{A}_\lambda\}_{\lambda \in \mathbb{N}}$, where the depth of $\mathcal{A}_\lambda$ is bounded from above by $T^\epsilon(\lambda)$, there exists a negligible function $\mathsf{negl}(\cdot)$ such that for all $\lambda \in \mathbb{N}$ it holds

$$\mathbf{Adv}^{\mathtt{TLP}}_{\mathcal{A}} = \left|\Pr\left[\mathsf{ExpTLP}^0_{\mathcal{A}}(\lambda) = 1\right] - \Pr\left[\mathsf{ExpTLP}^1_{\mathcal{A}}(\lambda) = 1\right]\right| \leq \mathsf{negl}(\lambda).$$

Relation to prior definitions.

Syntactically, a time-lock puzzle as defined by Rivest *et al.* [RSW96], Bitansky *et al.* [BGJ+16] and Malavolta and Thyagarajan [MT19] has the algorithm Gen which receives s as an additional input and outputs a puzzle Z. This immediately yields a timed-release encryption (TRE) scheme by viewing s as a message that is encrypted. Our definition enables a slightly simpler generic construction of (homomorphic) TRE. Intuitively, our new definitions relates to the prior one in a similar way like a key encapsulation mechanism relates to an encryption scheme. Concretely, let $\mathtt{TLP} = (\mathsf{Gen}, \mathsf{Solve})$ be a puzzle according to our new definition. Then we obtain a puzzle $\mathtt{TLP}' = (\mathsf{Gen}', \mathsf{Solve}')$ of the old form as follows:

- $\mathsf{Gen}'(1^\lambda, T, m)$ computes $(Z, s) \leftarrow \mathsf{Gen}(1^\lambda, T)$ outputs $Z' = (Z, m \oplus s)$.
- $\mathsf{Solve}'(Z' = (Z, c))$ computes $s \leftarrow \mathsf{Solve}(Z)$ and outputs $c \oplus s$.

Regarding security, the original definition of Bitansky *et al.* [BGJ+16] requires that no adversary is able to tell for any pair of solutions, which solution is contained in the given puzzle, unless the adversary has sufficient running time to solve the puzzle.

Other Variants of TLPs.

Bitansky *et al.* [BGJ+16] have proposed two weaker notions of time-lock puzzle. The first one relaxes the runtime requirement for Gen algorithm and instead of requiring a runtime of $\mathsf{poly}(\log T, \lambda)$ it allows the execution of Gen in "fast parallel time". This type of puzzle is denoted as a *weak* TLP. The second one is denoted as TLP with pre-precessing and allows for an (expensive) setup. Next, we define these two variants, however, we adjust the original definitions to match our definition for TLPs.

Definition 4.2.3 (Weak Time-Lock Puzzles [BGJ+16])**.** A weak time-lock puzzle (wTLP) $\mathtt{wTLP} = (\mathsf{Gen}, \mathsf{Solve})$ is satisfying the syntax and completeness requirements of Definition 4.2.1, but with the following relaxed efficiency requirement for the Gen algorithm: Gen can be computed by a uniform circuit of size $\mathsf{poly}(T, \lambda)$ and depth $\mathsf{poly}(\log T, \lambda)$.

We say $\mathtt{wTLP}$ is secure if it is a secure TLP in the sense of Definition 4.2.2.

Definition 4.2.4 (Time-Lock Puzzles with Pre-processing [BGJ+16])**.** A time-lock puzzle with pre-processing (ppTLP) is a tuple of algorithms $\mathtt{ppTLP} = (\mathsf{Preproc}, \mathsf{Gen}, \mathsf{Solve})$ with the following properties:

- $(P, K) \leftarrow \mathsf{Preproc}(1^\lambda, 1^T)$ is a probabilistic algorithm that takes as input the security parameter 1^λ and a difficulty parameter 1^T and outputs a state P and a short string $K \in \{0,1\}^\lambda$. $\mathsf{Preproc}$ can be computed by a uniform circuit of total size $T \cdot \mathsf{poly}(\lambda)$ and depth $\mathsf{poly}(\log T, \lambda)$.
- $(Z, s) \leftarrow \mathsf{Gen}(K)$ is a probabilistic algorithm that takes as input the secret key K and outputs a puzzle Z together with a solution s. Gen can be computed in (sequential) time $\mathsf{poly}(\log T, \lambda)$.
- $s \leftarrow \mathsf{Solve}(P, Z)$ is a deterministic algorithm that takes as input a state P and puzzle Z and outputs a solution s. Solve can be computed in time $T \cdot \mathsf{poly}(\lambda)$.

A time-lock puzzle with pre-processing is *correct* if for all λ, for all hardness parameters T it holds

$$\Pr\left[s = s' : (P, K) \leftarrow \mathsf{Preproc}(1^\lambda, 1^T), (Z, s) \leftarrow \mathsf{Gen}(K), s' \leftarrow \mathsf{Solve}(P, Z)\right] = 1.$$

Definition 4.2.5 (Security of ppTLP [BGJ+16])**.** We say that ppTLP is *secure with gap* $0 < \epsilon < 1$, if there exists a polynomial $\tilde{T}(\cdot)$ such that for all polynomials $T(\cdot) \geq \tilde{T}(\cdot)$ and every polynomial-size adversary $\mathcal{A} = \{\mathcal{A}_\lambda\}_{\lambda\in\mathbb{N}}$, where the depth of $\mathcal{A}_\lambda$ is bounded from above by $T^\epsilon(\lambda)$, there exists a negligible function $\mathsf{negl}(\cdot)$ such that for all $\lambda \in \mathbb{N}$, for all $k = \mathsf{poly}(\lambda)$ it holds

$$\left|\Pr\left[b = b' \;:\; \begin{array}{r} b \xleftarrow{\$} \{0,1\} \\ (P,K) \leftarrow \mathsf{Preproc}(1^\lambda, 1^T) \\ ((Z_i, s_{0,i}) \leftarrow \mathsf{Gen}(K))_{i\in[k]} \\ (s_{1,i} \xleftarrow{\$} S)_{i\in[k]} \\ b' \leftarrow \mathcal{A}_\lambda(P, (Z_i, s_{b,i})_{i\in[k]}) \end{array}\right] - \frac{1}{2}\right| \leq \mathsf{negl}(\lambda).$$

4.2.1 Instantiating TLPs from Sequential Squaring

The instantiation of a TLP from the sequential squaring assumption (Definition 2.5.3) is straightforward:

- $\mathsf{Gen}(1^\lambda, T)$: Run $(p, q, N) \leftarrow \mathsf{GenMod}(1^\lambda)$. Compute $t := 2^T \bmod |\mathbb{QR}_N|$, where $|\mathbb{QR}_N| = \frac{(p-1)(q-1)}{4}$. Randomly sample $x \xleftarrow{\$} \mathbb{QR}_N$ and compute the value $s := x^t \bmod N$. Set $Z := (N, T, x)$ and output (Z, s).
- $\mathsf{Solve}(Z)$: Compute $s := x^{2^T} \bmod N$ by repeated squaring.

The security of this construction is directly implied by the security of the sequential squaring assumption.

4.2.2 Instantiating TLPs from Randomized Encodings

Time-lock puzzles and their variants can be instantiated using different variants of randomized encodings [IK00, AIK06]. This was shown by Bitansky *et al.* [BGJ+16] and here we provide short overview of their results. We remark that the security of our TLP when based on the one from [BGJ+16] (where the adversary outputs two solutions (s_0, s_1) and obtains a puzzle for one of them) can be argued in an analogous way to the construction of KEMs from PKE.

Constructions.

Let RE be a randomized encoding scheme. For $s \in \{0,1\}^\lambda$ and $T \leq 2^\lambda$, let M_s^T be a machine that, on any input $x \in \{0,1\}^\lambda$ outputs the string s after T steps. Furthermore, M_s^T is described by 3λ bits (which is possible for large enough λ). Then the (w)TLP is constructed as follows:

- $\mathsf{Gen}(1^\lambda, T)$: Sample $s \xleftarrow{\$} \{0,1\}^\lambda, \widehat{M_s^T}(0^\lambda) \leftarrow \mathtt{RE}.\mathsf{Encode}(M_s^T, 0^\lambda, T, 1^\lambda)$ and output $(Z := \widehat{M_s^T}(0^\lambda), s)$.
- $\mathsf{Solve}(Z)$: Return $\mathtt{RE}.\mathsf{Decode}(Z)$.

Theorem 4.2.1 (Thm 3.7 [BGJ+16])**.** *Let $0 < \epsilon < 1$. Assume that, for every polynomial bounded function $T(\cdot)$, there exists a non-parallelizing language $L \in Dtime(T(\cdot))$ with gap ϵ. Let* RE *in above construction be a secure succinct randomized encoding in the sense of Definition 2.3.16. Then, for any $\underline{\epsilon} < \epsilon$, the above construction is a time-lock puzzle with gap $\underline{\epsilon}$.*

Succinct REs can be constructed assuming the existence of one-way functions and indistinguishability obfuscation (cf. [KLW15]). If we relax the succinctness requirement on RE, we obtain wTLP.

Theorem 4.2.2 (Thm 3.10 [BGJ+16])**.** *Let $0 < \epsilon < 1$. Assume that, for every polynomial bounded function $T(\cdot)$, there exists a non-parallelizing language $L \in Dtime(T(\cdot))$ with gap ϵ. Let* RE *in above construction be a secure succinct randomized encoding in the sense of Definitions 2.3.14 and 2.3.18. Then, for any $\underline{\epsilon} < \epsilon$, the above construction is a weak time-lock puzzle with gap $\underline{\epsilon}$.*

Using the fact that garbled circuits yield randomized encodings (cf. e.g., for discussion [App17]), we have the following:

Corollary 4.2.1. *Assuming the existence of one-way functions, there exists a randomized encoding scheme.*

Hence, we can construct wTLPs from one-way functions.

Time-lock puzzles with pre-processing are constructed using a reusable randomized encoding scheme RE that can be obtained assuming sub-exponential hardness of the LWE problem [GKP+13b]. The construction of ppTLP is as follows:

- $\mathsf{Preproc}(1^\lambda, T)$: Sample $(\widehat{U}, K') \leftarrow \mathtt{RE}.\mathsf{Preproc}(3\lambda, \lambda, T, 1^\lambda)$ and return $(P := \widehat{U}, K := K')$.
- $\mathsf{Gen}(K)$: Sample $s \xleftarrow{\$} \{0,1\}^\lambda, \widehat{M_s^T}(0^\lambda) \leftarrow \mathtt{RE}.\mathsf{Encode}(M_s^T, 0^\lambda, K)$ and output $(Z := \widehat{M_s^T}(0^\lambda), s)$.

- $\mathsf{Solve}(P, Z)$: Return $\mathtt{RE}.\mathsf{Decode}(P, Z)$.

For the construction we have the following:

Theorem 4.2.3 (Thm 4.8 [BGJ+16])**.** *Let $0 < \epsilon < 1$. Assume that, for every polynomial bounded function $T(\cdot)$, there exists a non-parallelizing language $L \in Dtime(T(\cdot))$ with gap ϵ. Then, for any $\underline{\epsilon} < \epsilon$, the above construction is a time-lock puzzle with pre-processing with gap $\underline{\epsilon}$.*

Remark 4.2.1. As mentioned in [MT19], for certain applications (e.g., e-voting or sealed-bid auctions) it might be perfectly acceptable to run an expensive setup ahead of time.

4.3 Sequential Time-Lock Puzzles

In this section, we introduce *sequential* time-lock puzzles along with their security and propose an instantiation which we prove secure under a new assumption called the *gap sequential squaring* assumption. We also show that this assumption holds, assuming that factoring is hard in the strong algebraic group model (SAGM) of Katz *et al.* [KLX20].

Sequential time-lock puzzles are a particularly useful generalization of basic TLPs, which yield practical timed-release encryption schemes. To this end, we generalize Definition 4.2.1 by allowing the Gen algorithm to take multiple different time parameters as input, which then produces a corresponding set of puzzles.

Definition 4.3.1. A *sequential time-lock puzzle* is tuple of algorithms $\mathtt{sTLP} = (\mathsf{Gen}, \mathsf{Solve})$ with the following syntax.

- $(Z_i, s_i)_{i\in[n]} \leftarrow \mathsf{Gen}(1^\lambda, (T_i)_{i\in[n]})$ is a probabilistic algorithm which takes as input the security parameter 1^λ and n non-negative integers $(T_i)_{i\in[n]}$ and outputs n puzzles together with their solutions $(Z_i, s_i)_{i\in[n]}$ in time at most $\mathsf{poly}((\log T_i)_{i\in[n]}, \lambda)$. Without loss of generality we assume in the sequel that the set $(T_i)_{i\in[n]}$ is ordered and hence $T_i < T_{i+1}$ for all $i \in [n-1]$.

- $s_i \leftarrow \mathsf{Solve}(Z_i, s_{i-1})$ is a deterministic algorithm that takes as input a puzzle Z_i and a solution for puzzle Z_{i-1} and outputs a solution s_i, where we define $s_0 := \bot$. We require that Solve runs in time at most $(T_i - T_{i-1}) \cdot \mathsf{poly}(\lambda)$, where we define $T_0 := 0$.

We say a sequential time-lock puzzle is *correct* if for all $\lambda, n \in \mathbb{N}$, for all sets of hardness parameters $(T_j)_{j\in[n]}$ such that $\forall j \in [n-1] : T_j < T_{j+1}$, for all $i \in [n]$ it holds

$$\Pr\left[s_i = s_i' : (Z_j, s_j)_{j\in[n]} \leftarrow \mathsf{Gen}(1^\lambda, (T_j)_{j\in[n]}), s_i' \leftarrow \mathsf{Solve}(Z_i, s_{i-1})\right] = 1.$$

$\underline{\mathsf{ExpsTLP}^b_{\mathcal{A}_i}(\lambda)\text{:}}$
$(Z_j, s_j)_{j\in[n]} \leftarrow \mathsf{Gen}(1^\lambda, (T_j(\lambda))_{j\in[n]})$
$(y_j := \mathsf{F}(T_j(\lambda), s_j))_{j\in\{[n]\setminus\{i\}\}}$
if $b = 0 : y_i := \mathsf{F}(T_i(\lambda), s_i)$
if $b = 1 : y_i \xleftarrow{\$} Y$
return $b' \leftarrow \mathcal{A}_{i,\lambda}((Z_j, y_j)_{j\in[n]})$

Figure 4.2: Security experiment for sequential time-lock puzzles.

In order to define a security notion for sequential time-lock puzzles that is useful for our application of constructing particularly efficient timed-release encryption schemes, we need to introduce an additional function $\mathsf{F} : \mathbb{N} \times S \to Y$ that takes as input a pair a hardness parameter $T \in \mathbb{N}$ together with a solution $s \in S$ and outputs elements of some set Y. Instead of requiring that elements s_i are indistinguishable from random, we require that $y_i = \mathsf{F}(T_i, s_i)$ is indistinguishable from random. We explain the necessity for the function F after the following definition.

Definition 4.3.2. Consider the security experiment $\mathsf{ExpsTLP}^b_{\mathcal{A}_i}(1^\lambda)$ in Figure 4.2. We say that a sequential time-lock puzzle $\mathtt{sTLP}$ is *secure with gap* $0 < \epsilon < 1$ *and with respect to the function* F, if for all polynomials n in λ there exists a polynomial $\tilde{T}(\cdot)$ such that for all sets of polynomials $(T_j(\cdot))_{j\in[n]}$ fulfilling that $\forall j \in [n] : T_j(\cdot) \geq \tilde{T}(\cdot)$, for all $i \in [n]$ and every polynomial-size adversary $\mathcal{A}_i = \{\mathcal{A}_{i,\lambda}\}_{\lambda\in\mathbb{N}}$, where the depth of $\mathcal{A}_{i,\lambda}$ is bounded from above by $T_i^\epsilon(\lambda)$, there exists a negligible function $\mathsf{negl}(\cdot)$ such that for all $\lambda \in \mathbb{N}$ it holds

$$\mathbf{Adv}^{\mathtt{sTLP}}_{\mathcal{A}_i} = \left|\Pr\left[\mathsf{ExpsTLP}^0_{\mathcal{A}_i}(\lambda) = 1\right] - \Pr\left[\mathsf{ExpsTLP}^1_{\mathcal{A}_i}(\lambda) = 1\right]\right| \leq \mathsf{negl}(\lambda).$$

On the need for function F.

The novelty of our security definition is that it is defined with a respect to some function F. Let us explain why this is necessary.

Our ultimate goal is to build a sequential timed-release encryption scheme based on a sequential time-lock puzzle. The setup of sTRE generates a set of public parameters which correspond to an increasing sequence of hardness parameters. We consider sTRE to be secure if an adversary whose runtime T is less than T_i can not get any useful information about plaintext from the ciphertext which was encrypted with respect to hardness parameter T_i (see Definition 4.4.2). The main idea of our construction is to use solutions of the puzzles as random coins which are used by the algorithm Gen of a PKE scheme to produce a set of public keys

$\mathsf{pk}_1, \ldots, \mathsf{pk}_n$. To prove the security with respect to an adversary whose runtime is $T < T_i$, a reduction must be able to simulate all public keys $\mathsf{pk}_1, \ldots, \mathsf{pk}_{i-1}$ perfectly and therefore it must also know the corresponding solutions $s_1, \ldots, s_{i-1}$.

One option would be to compute the required solutions $s_1, \ldots, s_{i-1}$ directly in the reduction. In this case the runtime of the reduction is at least $T + T_{i-1}$ where T is the runtime of the adversary and time T_{i-1} is needed to compute the solutions $s_1, \ldots, s_{i-1}$. Depending on the values T, T_{i-1} and T_i the runtime of the reduction can be larger as T_i, therefore this approach does not work.

Another option is to provide the solutions $s_1, \ldots, s_{i-1}$ to a reduction as part of the sTLP instance. Unfortunately, this does not work either, because given the value s_{i-1} one can compute the solution s_i in time $T_i - T_{i-1}$. Therefore the reduction would only be able to break the assumption that a puzzle for hardness parameter T_i is secure against adversaries which run in time less than $T_i - T_{i-1}$ and we can not achieve any security for adversaries whose runtime T is $T_i - T_{i-1} \leq T < T_i$.

Our solution to overcome this difficulty is to construct a TRE scheme which does not directly use the real solutions s_i, but instead it uses values $\mathsf{F}(T_i, s_i)$ where one can think of F as a hard-to-invert function. This way we are able to formulate a hardness assumption for TLPs where the reduction in the security proof of the TRE scheme receives $\mathsf{F}(T_1, s_1), \ldots, \mathsf{F}(T_{i-1}, s_{i-1}), \mathsf{F}(T_{i+1}, s_{i+1}), \ldots, \mathsf{F}(T_n, s_n)$ as additional "advice" that can be used to provide a proper simulation. At the same time it is reasonable to assume that no adversary (our reduction, here) is able to distinguish $\mathsf{F}(T_i, s_i)$ from random, even if it runs in time up to $T < T_i$, which is exactly the upper bound that we have on the TRE adversary.

We remark that Malavolta *et al.* [MT19, Section 5.4] also describe a similar approach that allows using multiple hardness parameters at once. The technical difficulty described here should arise in their construction as well. However, they provide only informal description without any security proof, hence this is not clarified. We believe that a similar assumption involving an "advice" for the reduction is also necessary for a security proof of the construction suggested in their work.

4.3.1 Instantiating Sequential TLPs from Sequential Squaring

In order to obtain a sequential TLP, we define a variant of the sequential squaring assumption in which an adversary is given oracle access to a *decisional sequential squaring verification function* DSSvf. DSSvf is defined with respect to values x and N, and it takes as input a hardness parameter T' and a value $y' \in \mathbb{QR}_N$ and outputs 1 if $y' = x^{2^{T'}} \bmod N$, otherwise it outputs 0. The assumption essentially states that the computational sequential squaring assumption remains hard, even if the adversary is given access to DSSvf, akin to other gap assumptions [OP01].

Definition 4.3.3 (The Gap Sequential Squaring (GSS) Assumption)**.** The *gap sequential squaring assumption* with gap $0 < \epsilon < 1$ holds relative to GenMod if there exists a polynomial $\tilde{T}(\cdot)$ such that for all polynomials $T(\cdot) \geq \tilde{T}(\cdot)$ and for every polynomial-size adversary $\mathcal{A} = \{\mathcal{A}_\lambda\}_{\lambda\in\mathbb{N}}$, where the depth of $\mathcal{A}_\lambda$ is bounded from above by $T^\epsilon(\lambda)$, there exists a negligible function $\mathsf{negl}(\cdot)$ such that for all $\lambda \in \mathbb{N}$ it holds

$$\mathbf{Adv}^{\mathtt{GSS}}_{\mathcal{A}} = \Pr\left[y = x^{2^T} \bmod N : \begin{array}{r}(p,q,N) \leftarrow \mathsf{GenMod}(1^\lambda), x \xleftarrow{\$} \mathbb{QR}_N \\ y \leftarrow \mathcal{A}_\lambda^{\mathsf{DSSvf}(\cdot,\cdot)}(N, T(\lambda), x)\end{array}\right] \leq \mathsf{negl}(\lambda),$$

where $\mathsf{DSSvf}(\cdot,\cdot)$ is an oracle which takes as input a hardness parameter T' and a value y' and outputs 1 if and only if $y' = x^{2^{T'}} \bmod N$.

Now we are ready to construct our sequential TLP:

- $\mathsf{Gen}(1^\lambda, (T_i)_{i\in[n]})$: Run $(p,q,N) \leftarrow \mathsf{GenMod}(1^\lambda)$. Randomly sample $x \xleftarrow{\$} \mathbb{QR}_N$ and compute values $t_i := 2^{T_i} \bmod |\mathbb{QR}_N|$, $s_i := x^{t_i} \bmod N$ for all $i \in [n]$ where $|\mathbb{QR}_N| = \frac{(p-1)(q-1)}{4}$. Set $T_0 := 0$ and output $((N, x, T_i, T_{i-1}), s_i))_{i\in[n]}$.
- $\mathsf{Solve}((N, x, T_i, T_{i-1}), s_{i-1})$: Compute value $s_{i-1}^{T_i - T_{i-1}} \bmod N$ by repeated squaring.

Theorem 4.3.1. *If the gap sequential squaring assumption with gap ϵ holds relative to* GenMod *and* F *is modelled as a random oracle, then for any $\underline{\epsilon} < \epsilon$, the* $\mathtt{TLP} = (\mathsf{Gen}, \mathsf{Solve})$ *defined above is a secure sequential time-lock puzzle with gap $\underline{\epsilon}$ and with respect to the function* F*.*

Proof. Let $\tilde{T}_{\mathtt{GSS}}(\lambda)$ be the polynomial whose existence is guaranteed by the GSS assumption. Let $\mathsf{poly}_{\mathsf{RO}}(\lambda)$ be the fixed polynomial which bounds the time required to execute Step 1 and simulate random oracle answers as specified in Step 2 of the adversary $\mathcal{B}_\lambda$ defined below. Set $\underline{T}(\lambda) := (\mathsf{poly}_{\mathsf{RO}}(\lambda))^{1/\underline{\epsilon}}$. Set $\tilde{T}_{\mathtt{sTLP}} := \max(\tilde{T}_{\mathtt{GSS}}, \underline{T})$.

For any n which is polynomial in λ, any tuple $(T_j(\cdot))_{j\in[n]}$ fulfilling that $\forall j \in [n]$ holds $T_j(\cdot) \geq \tilde{T}_{\mathtt{sTLP}}(\cdot)$, any $i \in [n]$, from any polynomial-size adversary $\mathcal{A}_i = \{\mathcal{A}_{i,\lambda}\}_{\lambda\in\mathbb{N}}$, where the depth of $\mathcal{A}_{i,\lambda}$ is bounded from above by $T_i^\epsilon(\lambda)$, we can construct a polynomial-size adversary $\mathcal{B} = \{\mathcal{B}_\lambda\}_{\lambda\in\mathbb{N}}$ whose depth is bounded from above by $T_i^\epsilon(\lambda)$ such that

$$\mathbf{Adv}^{\mathtt{GSS}}_{\mathcal{B}} = \mathbf{Adv}^{\mathtt{sTLP}}_{\mathcal{A}_i}.$$

Intuitively, an adversary can not distinguish $\mathsf{F}(T_i, s_i)$ from random without asking (T_i, s_i) to F, since F is a random oracle. More formally, let Query be the event that $\mathcal{A}_i$ asks (T_i, s_i) to F, where s_i is the correct solution of the puzzle (N, T_i, x).

Then, by arguing as in Shoup's Difference Lemma [Sho04], we get:

$$\begin{aligned}
\mathbf{Adv}^{\mathtt{sTLP}}_{\mathcal{A}_i} &= \left|\Pr\left[\mathsf{ExpsTLP}^0_{\mathcal{A}_i}(\lambda)=1\right] - \Pr\left[\mathsf{ExpsTLP}^1_{\mathcal{A}_i}(\lambda)=1\right]\right| \\
&= \Big| \Pr\left[\mathsf{ExpsTLP}^0_{\mathcal{A}_i}(\lambda)=1 \wedge \mathsf{Query}\right] + \Pr\left[\mathsf{ExpsTLP}^0_{\mathcal{A}_i}(\lambda)=1 \wedge \overline{\mathsf{Query}}\right] \\
&\quad - \Pr\left[\mathsf{ExpsTLP}^1_{\mathcal{A}_i}(\lambda)=1 \wedge \mathsf{Query}\right] - \Pr\left[\mathsf{ExpsTLP}^1_{\mathcal{A}_i}(\lambda)=1 \wedge \overline{\mathsf{Query}}\right] \Big| \\
&= \Big| \Pr\left[\mathsf{ExpsTLP}^0_{\mathcal{A}_i}(\lambda)=1 \wedge \mathsf{Query}\right] - \Pr\left[\mathsf{ExpsTLP}^1_{\mathcal{A}_i}(\lambda)=1 \wedge \mathsf{Query}\right] \Big| \\
&\leq \Pr[\mathsf{Query}].
\end{aligned}$$

The second equation follows from the fact that if Query does not happen, then the value y_i is uniformly distributed in both experiments. Therefore the event $\mathsf{ExpsTLP}^0_{\mathcal{A}_i}(\lambda)=1 \wedge \overline{\mathsf{Query}}$ occurs if and only if $\mathsf{ExpsTLP}^1_{\mathcal{A}_i}(\lambda)=1 \wedge \overline{\mathsf{Query}}$, so that we get

$$\Pr\left[\mathsf{ExpsTLP}^0_{\mathcal{A}_i}(\lambda)=1 \wedge \overline{\mathsf{Query}}\right] = \Pr\left[\mathsf{ExpsTLP}^1_{\mathcal{A}_i}(\lambda)=1 \wedge \overline{\mathsf{Query}}\right].$$

The last inequality follows from the fact that both $\Pr\left[\mathsf{ExpsTLP}^0_{\mathcal{A}_i}(\lambda)=1 \wedge \mathsf{Query}\right]$ and $\Pr\left[\mathsf{ExpsTLP}^1_{\mathcal{A}_i}(\lambda)=1 \wedge \mathsf{Query}\right]$ are numbers between 0 and $\Pr[\mathsf{Query}]$.

Hence, in order to complete the proof, we show that $\Pr[\mathsf{Query}]$ is negligible, provided that the gap sequential squaring (GSS) assumption holds. To this end, we define an adversary $\mathcal{B}$ which breaks the GSS assumption.

The adversary $\mathcal{B}_\lambda(N, T(\lambda), x)$:

1. Sets $Z_j := (N, T_j(\lambda), T_{j-1}(\lambda), x)_{j\in[n]}$, where $T_i(\lambda) := T(\lambda)$ and for all $j \in [n]$ randomly samples values y_j from the image of F.

2. Runs $\mathcal{A}_{i,\lambda}((Z_j, y_j)_{j\in[n]})$. It initializes an empty list Q. When $\mathcal{A}_{i,\lambda}$ makes a query $(T'(\lambda), s')$ to F, $\mathcal{B}_\lambda$ answers it as follows:
   - If there is an entry in Q of the form $((T'(\lambda), s'), y)$ for some y, it returns y.
   - If $T'(\lambda) = T_j(\lambda)$ for some $j \in [n]$ and $\mathsf{DSSvf}(T'(\lambda), s') = 1$, stores $((T'(\lambda), s'), y_j)$ in Q. If $i = j$ then it outputs s' as a solution to the GSS problem. Otherwise, it returns y_j to $\mathcal{A}_{i,\lambda}$.
   - Otherwise, it samples a uniform random value y from the image of F, stores $((T'(\lambda), s'), y)$ in Q and returns y to $\mathcal{A}_{i,\lambda}$.

Notice that $\mathcal{B}$ consistently simulates the random oracle for $\mathcal{A}_i$. Moreover, when Query occurs, then $\mathcal{B}$ outputs the correct solution to the GSS problem. Hence,

$$\Pr[\mathsf{Query}] \leq \mathbf{Adv}^{\mathtt{GSS}}_{\mathcal{B}}.$$

Additionally, $\mathcal{B}$ fulfils the depth constraint:

$$\mathsf{depth}(\mathcal{B}_\lambda) = \mathsf{poly}_{\mathsf{RO}}(\lambda) + \mathsf{depth}(\mathcal{A}_{i,\lambda}) \leq \underline{T}^{\epsilon}(\lambda) + T^{\epsilon}(\lambda) \leq 2T^{\epsilon}(\lambda) = o(T^{\epsilon}(\lambda)).$$

On the other hand $T(\cdot) \geq \tilde{T}_{\mathtt{sTLP}}(\cdot) \geq \tilde{T}_{\mathtt{GSS}}(\cdot)$ as required.
This concludes the proof. □

4.3.2 Hardness of the Gap Sequential Squaring Assumption

Now we show that the gap sequential squaring problem is at least as hard as factoring N in the Strong Algebraic Group Model (SAGM) (see Section 2.7). In order to prove the hardness of the Gap Sequential Squaring Problem, we use the following well-known fact which states that we can factor N if a multiple of $\varphi(N)$ is known. The following lemma extends Lemma 1 of [KLX20] from a success probability of $1/2$ to a success probability negligibly close to 1, which is required for our proof.

Lemma 4.3.1. *Let $(p, q, N) \leftarrow \mathsf{GenMod}(1^\lambda)$ and let $m = \alpha\varphi(N)$ for some positive integer $\alpha \in \mathbb{Z}^+$. There exists a PPT algorithm $\mathsf{Factor}(N, m)$ that on input (N, m) outputs $p', q' \in \mathbb{N}$, $p', q' > 1$ such that $N = p'q'$ with probability at least $1 - 2^{-\lambda}$.*

Proof. Lemma 1 of [KLX20] guarantees an existence of efficient $\mathsf{Factor}(N, m)$ algorithm which has success probability at least $1/2$. By running this algorithm λ-times with independent randomness, we obtain claimed success probability. □

We are now ready to state and prove the hardness of the GSS assumption.

Theorem 4.3.2. *If the factoring assumption holds relative to* GenMod*, then the gap sequential squaring assumption with gap ϵ holds relative to* GenMod *in the SAGM for any $0 < \epsilon < 1$.*

Proof. We show that for any positive integer T the gap sequential squaring assumption holds against adversaries which perform at most $T-1$ algebraic steps. Notice that this means that the gap $\epsilon = \log_T(T-1)$. Since $\lim_{T\to\infty} \log_T(T-1) = 1$, ϵ can be defined arbitrary close to 1 by choosing $\tilde{T}$ appropriately. Let $\mathcal{A}$ be a strongly algebraic adversary which executes at most $T-1$ algebraic steps and asks at most ℓ queries to DSSvf oracle. Let $x \in \mathbb{QR}_N$ be the group element which is given to $\mathcal{A}$ as part of its challenge. Following [KLX20], we recursively define for any $y \in \mathbb{QR}_N$ output by $\mathcal{A}$ the discrete logarithm of y with respect to $\mathcal{A}$ and x, $\mathsf{DL}_{\mathcal{A}}(x, y) \in \mathbb{Z}^+$ as follows:

1. $\mathsf{DL}_{\mathcal{A}}(x, x) = 1$.

2. If $\mathcal{A}$ outputs (y, y_1, y_2) in an algebraic step, then $\mathsf{DL}_{\mathcal{A}}(x, y) = \mathsf{DL}_{\mathcal{A}}(x, y_1) + \mathsf{DL}_{\mathcal{A}}(x, y_2)$.

3. If $\mathcal{A}$ outputs (y, y_1) in an algebraic step, then $\mathsf{DL}_{\mathcal{A}}(x, y) = -\mathsf{DL}_{\mathcal{A}}(x, y_1)$.

Now we can make the following observation. Assume that x is a generator of $\mathbb{QR}_N$. If $\mathcal{A}$ outputs a solution s' of a puzzle (N, T', x) in one of its queries to the DSSvf oracle or as the solution to the puzzle challenge, and at the same time it holds that $\mathsf{DL}_{\mathcal{A}}(x, s') \neq 2^{T'}$, then $|\mathbb{QR}_N|$ divides $2^{T'} - \mathsf{DL}_{\mathcal{A}}(x, s')$. This is because $1 = s'(s')^{-1} = x^{2^{T'}}(x^{\mathsf{DL}_{\mathcal{A}}(x,s')})^{-1} = x^{2^{T'} - \mathsf{DL}_{\mathcal{A}}(x,s')} \mod N$. Because $\mathsf{DL}_{\mathcal{A}}(x, s') \neq 2^{T'}$ this implies that $2^{T'} - \mathsf{DL}_{\mathcal{A}}(x, s')$ is a multiple of the order of the group $\mathbb{QR}_N$. Recall that $|\mathbb{QR}_N| = \varphi(N)/4$. Hence $4(2^{T'} - \mathsf{DL}_{\mathcal{A}}(x, s'))$ is multiple of $\varphi(N)$. By Lemma 4.3.1 we are able to factor N with probability at least $1 - 2^{-\lambda}$.

With this in mind, we are ready to construct an adversary $\mathcal{B}$ breaking the factoring assumption

The adversary $\mathcal{B}(N)$:

1. Samples randomly $x \xleftarrow{\$} \mathbb{QR}_N$ and runs $\mathcal{A}$ on input (N, T, x).

2. Whenever $\mathcal{A}$ asks a query (T', s') to DSSvf, $\mathcal{B}$ recursively computes $\mathsf{DL}_{\mathcal{A}}(x, s')$ and proceeds in the following way:
   a) If $2^{T'} = \mathsf{DL}_{\mathcal{A}}(x, s')$ then it returns 1 to $\mathcal{A}$.
   b) Otherwise, it sets $m := 4(2^{T'} - \mathsf{DL}_{\mathcal{A}}(x, s'))$ and executes the factoring algorithm from Lemma 4.3.1. If the factoring algorithm is successful, it outputs the corresponding factors as a solution to the factoring problem. Else it outputs 0 to $\mathcal{A}$.

3. When $\mathcal{A}$ returns a solution s, $\mathcal{B}$ computes $\mathsf{DL}_{\mathcal{A}}(x, s)$, sets $m := 4(2^{T'} - \mathsf{DL}_{\mathcal{A}}(x, s'))$, calls the factoring algorithm from Lemma 4.3.1 and returns whatever this algorithm returns.

Let us analyse the success probability of $\mathcal{B}$. Let FACTOR be the event that $\mathcal{B}$ successfully outputs the factorization of N and GSS be the event that $\mathcal{A}$ outputs the correct solution of the gap sequential squaring problem. Let GNR denote the event that the sampled x in Step 1 is a generator. Because x is sampled uniformly at random and $\mathbb{QR}_N$ has $\varphi(|\mathbb{QR}_N|) = (p'-1)(q'-1)$ generators, this event happens with overwhelming probability. Concretely, $\Pr[\mathsf{GNR}] = 1 - \frac{1}{p'} - \frac{1}{q'} + \frac{1}{p'q'}$.

In Step 2(b) we have that $2^{T'} \neq \mathsf{DL}_{\mathcal{A}}(x, s')$ and hence if s' was the solution of the given puzzle, then we should be able to factor N with probability at least $1 - 2^{-\lambda}$ by Lemma 4.3.1. Notice that $\mathcal{B}$ answers a DSSvf-query incorrectly only if s' is a solution of the puzzle and the factoring algorithm was unsuccessful. Let FAIL_i denote the event that $\mathcal{B}$ answered the i-th DSSvf query incorrectly and let

FAIL denote the event that it answered any of the ℓ queries incorrectly. Then $\Pr[\mathsf{FAIL}_i|\mathsf{GNR}] \leq \frac{1}{2^\lambda}$. Hence, the probability of FAIL can be upper bounded by a union bound:

$$\Pr[\mathsf{FAIL}|\mathsf{GNR}] = \Pr\left[\bigvee_{i=1}^{\ell} \mathsf{FAIL}_i|\mathsf{GNR}\right] \leq \frac{\ell}{2^\lambda}.$$

We obtain that $\mathcal{B}$ answers all DSSvf-queries correctly with probability

$$\Pr[\overline{\mathsf{FAIL}}|\mathsf{GNR}] = 1 - \Pr[\mathsf{FAIL}|\mathsf{GNR}] \geq 1 - \frac{\ell}{2^\lambda}.$$

If the event FAIL does not occur, then $\mathcal{B}$ perfectly simulates the Gap Sequential Squaring experiment. Perfect simulation guarantees that in Step 3, $\mathcal{A}$ outputs a correct solution with probability $\mathbf{Adv}^{\mathsf{GSS}}_{\mathcal{A}}$. Therefore $\Pr[\mathsf{GSS}|\overline{\mathsf{FAIL}}] = \mathbf{Adv}^{\mathsf{GSS}}_{\mathcal{A}}$. However, we are interested in $\Pr[\mathsf{GSS}|\mathsf{GNR} \wedge \overline{\mathsf{FAIL}}]$, which can be bounded as follows:

$$\begin{aligned}
\mathbf{Adv}^{\mathsf{GSS}}_{\mathcal{A}} &= \Pr[\mathsf{GSS}|\overline{\mathsf{FAIL}}] \\
&= \Pr[\mathsf{GSS}|\mathsf{GNR} \wedge \overline{\mathsf{FAIL}}] \Pr[\mathsf{GNR}] + \Pr[\mathsf{GSS}|\overline{\mathsf{GNR}} \wedge \overline{\mathsf{FAIL}}] \Pr[\overline{\mathsf{GNR}}] \\
&\leq \Pr[\mathsf{GSS}|\mathsf{GNR} \wedge \overline{\mathsf{FAIL}}] \Pr[\mathsf{GNR}] + \Pr[\overline{\mathsf{GNR}}].
\end{aligned}$$

Hence,

$$\Pr[\mathsf{GSS}|\mathsf{GNR} \wedge \overline{\mathsf{FAIL}}] \Pr[\mathsf{GNR}] \geq \mathbf{Adv}^{\mathsf{GSS}}_{\mathcal{A}} - \Pr[\overline{\mathsf{GNR}}] = \mathbf{Adv}^{\mathsf{GSS}}_{\mathcal{A}} - \frac{1}{p'} - \frac{1}{q'} + \frac{1}{p'q'}.$$

Now, because $\mathcal{A}$ can perform at most $T-1$ group operations and the only group element given as input is x, it must hold $\mathsf{DL}_{\mathcal{A}}(x,s) < 2^{T-1}$ (this is formally proven in [KLX20, Claim in Theorem 2]. Therefore $2^T \neq \mathsf{DL}_{\mathcal{A}}(x,s)$, which implies that $\varphi(N)|4(2^T - \mathsf{DL}_{\mathcal{A}}(x,s))$. By using Lemma 4.3.1 we are able to factor N with probability at least $1-2^{-\lambda}$. Hence, $\Pr[\mathsf{FACTOR}|\mathsf{GNR} \wedge \overline{\mathsf{FAIL}} \wedge \mathsf{GSS}] \geq 1-2^{-\lambda}$. We conclude that

$$\begin{aligned}
&\mathbf{Adv}^{\mathsf{Factor}}_{\mathcal{B}} = \Pr[\mathsf{FACTOR}] \\
&= \Pr[\mathsf{FACTOR}|\mathsf{GNR}] \Pr[\mathsf{GNR}] + \Pr[\mathsf{FACTOR}|\overline{\mathsf{GNR}}] \Pr[\overline{\mathsf{GNR}}] \\
&\geq \Pr[\mathsf{FACTOR}|\mathsf{GNR}] \Pr[\mathsf{GNR}] \\
&\geq \Pr[\mathsf{FACTOR}|\mathsf{GNR} \wedge \overline{\mathsf{FAIL}}] \Pr[\overline{\mathsf{FAIL}}|\mathsf{GNR}] \Pr[\mathsf{GNR}] \\
&\geq \Pr[\mathsf{FACTOR}|\mathsf{GNR} \wedge \overline{\mathsf{FAIL}} \wedge \mathsf{GSS}] \Pr[\mathsf{GSS}|\mathsf{GNR} \wedge \overline{\mathsf{FAIL}}] \Pr[\overline{\mathsf{FAIL}}|\mathsf{GNR}] \Pr[\mathsf{GNR}] \\
&\geq (1 - \frac{1}{2^\lambda})(1 - \frac{\ell}{2^\lambda})(\mathbf{Adv}^{\mathsf{GSS}}_{\mathcal{A}} - \frac{1}{p'} - \frac{1}{q'} + \frac{1}{p'q'}),
\end{aligned}$$

which completes the proof. □

Extension to the *Strong* Sequential Squaring Assumption.

Malavolta *et al.* [MT19] and Katz *et al.* [KLX20] also consider the so-called *strong* sequential squaring assumption which is stated in Definition 2.5.4 and involves a two-stage adversary $\mathcal{A} = \{(\mathcal{A}_{1,\lambda}, \mathcal{A}_{2,\lambda})\}_{\lambda\in\mathbb{N}}$.

It is straightforward to define a strong *gap* sequential squaring assumption (SGSS), with a two-stage adversary $\mathcal{A} = \{(\mathcal{A}_{1,\lambda}, \mathcal{A}_{2,\lambda})\}_{\lambda\in\mathbb{N}}$ where $\mathcal{A}_{1,\lambda}$ is unbounded, but independent of the sequential squaring challenge, while $\mathcal{A}_{2,\lambda}$ is bounded and additionally has access to a DSSvf oracle as defined in Definition 4.3.3. We note that the hardness of the strong gap sequential squaring assumption can be proven in similar way as the hardness of the GSS assumption. Although the GSS assumption is sufficient for building a sequential TLP, the SGSS assumption might be useful for future applications.

4.4 (Sequential) Timed-Release Encryption

In this section, we give generic constructions of (sequential) timed-release encryption schemes based on (sequential) TLPs. There exist several definitions for TRE and we base ours on that of Unruh [Unr14]. However, we introduce two additional algorithms Setup and Solve which leads to better modularity and applicability of TRE, as we illustrate in Section 4.4.5.

Definition 4.4.1. A sequential timed-release encryption scheme $\mathtt{TRE}$ with message space $\mathcal{M}$ is a tuple of algorithms $\mathtt{TRE} = (\mathsf{Setup}, \mathsf{Enc}, \mathsf{Solve}, \mathsf{Dec})$ with the following syntax.

- $(\mathsf{pp}_{e,i}, \mathsf{pp}_{d,i})_{i\in[n]} \leftarrow \mathsf{Setup}(1^\lambda, (T_i)_{i\in[n]})$ is a probabilistic algorithm that takes as input the security parameter 1^λ and a set of hardness parameters $(T_i)_{i\in[n]}$ with $T_i < T_{i+1}$ for all $i \in [n-1]$ and outputs a set of public encryption parameters and public decryption parameters $(\mathsf{pp}_{e,i}, \mathsf{pp}_{d,i})_{i\in[n]}$. We require that Setup runs in time $\mathsf{poly}((\log T_i)_{i\in[n]}, \lambda)$.

- $s_i \leftarrow \mathsf{Solve}(\mathsf{pp}_{d,i}, s_{i-1})$ is a deterministic algorithm that takes as input public decryption parameters $\mathsf{pp}_{d,i}$ and the solution from a previous iteration s_{i-1}, where $s_0 := \bot$ and outputs a solution s_i. We require that Solve runs in time at most $(T_i - T_{i-1}) \cdot \mathsf{poly}(\lambda)$.

- $c \leftarrow \mathsf{Enc}(\mathsf{pp}_{e,i}, m)$ is a probabilistic algorithm that takes as input public encryption parameters $\mathsf{pp}_{e,i}$ and message $m \in \mathcal{M}$ and outputs a ciphertext c.

- $m/\perp \leftarrow \mathsf{Dec}(T_i, s_i, c)$ is a deterministic algorithm that takes as input a hardness parameter T_i, a solution s_i and a ciphertext c and outputs $m \in \mathcal{M}$ or $\perp$.

We say a sequential timed-release encryption scheme is *correct* if for all $\lambda, n \in \mathbb{N}$, for all sets of hardness parameters $(T_j)_{j\in[n]}$ such that $\forall j \in [n-1] : T_j < T_{j+1}$, for all $i \in [n]$ and for all messages $m \in \mathcal{M}$ it holds

$$\Pr\left[m = m' \ : \begin{array}{r}(\mathsf{pp}_{e,i}, \mathsf{pp}_{d,i})_{i\in[n]} \leftarrow \mathsf{Setup}(1^\lambda, (T_j)_{j\in[n]}) \\ s_i \leftarrow \mathsf{Solve}(\mathsf{pp}_{d,i}, s_{i-1}) \\ m' \leftarrow \mathsf{Dec}(T_i, s_i, \mathsf{Enc}(\mathsf{pp}_{e,i}, m))\end{array}\right] = 1.$$

Note that the above definition also defines "non-sequential" TRE, by setting $n = 1$. In that case the value T_i is not needed as an input for the Dec algorithm, however, for sequential TRE, this value is necessary. For ease of the notation, the definition of Dec is unified.

Definition 4.4.2. A sequential timed-release encryption scheme is *secure with gap* $0 < \epsilon < 1$ if for all polynomials n in λ there exists a polynomial $\tilde{T}(\cdot)$ such that for all sets of polynomials $(T_j)_{j\in[n]}$ fulfilling that $\forall j \in [n] : T_j(\cdot) \geq \tilde{T}(\cdot)$, for all $i \in [n]$ and every polynomial-size adversary $\mathcal{A}_i = \{\mathcal{A}_{i,\lambda}\}_{\lambda\in\mathbb{N}}$, where the depth of $\mathcal{A}_{i,\lambda}$ is bounded from above by $T_i^\epsilon(\lambda)$, there exists a negligible function $\mathsf{negl}(\cdot)$ such that for all $\lambda \in \mathbb{N}$ it holds

$$\mathbf{Adv}^{\mathtt{TRE}}_{\mathcal{A}_i} = \left|\Pr\left[b = b' \ : \begin{array}{r}(\mathsf{pp}_{e,i}, \mathsf{pp}_{d,i})_{i\in[n]} \leftarrow \mathsf{Setup}(1^\lambda, (T_j(\lambda))_{j\in[n]}) \\ (m_0, m_1) \leftarrow \mathcal{A}_{i,\lambda}((\mathsf{pp}_{e,i}, \mathsf{pp}_{d,i})_{i\in[n]}) \\ b \xleftarrow{\$} \{0,1\};\ c \leftarrow \mathsf{Enc}(\mathsf{pp}_{e,i}, m_b) \\ b' \leftarrow \mathcal{A}_{i,\lambda}(c)\end{array}\right] - \frac{1}{2}\right| \leq \mathsf{negl}(\lambda).$$

We require that $|m_0| = |m_1|$.

4.4.1 Basic TRE Construction

For our constructions we use a (sequential) time-lock puzzle and a public-key encryption scheme. For security of a PKE scheme we require IND-CPA security, since this is sufficient to construct a TRE scheme achieving Definition 4.4.2. A stronger CCA-style security notion for TRE would be achievable by using IND-CCA secure PKE. However, we consider this as not very useful for TRE, since it is unclear where in an application a "CCA-oracle" could plausibly exist in an application *before* the release time is reached since the decryption key is hidden until this point in time. After the release time, the ciphertext will be decryptable

anyway, so we have no security expectations. However, some applications may require the non-malleability of ciphertexts, which could be achieved via an IND-CCA-secure public-key encryption scheme, for instance. We discuss this later in Section 4.5 when talking about Timed Public-Key Encryption.

Construction.

Let $\mathtt{TLP} = (\mathtt{TLP.Gen}, \mathtt{TLP.Solve})$ be a TLP with solution space S and let $\mathtt{PKE} = (\mathtt{PKE.Gen}, \mathtt{PKE.Enc}, \mathtt{PKE.Dec})$ be a PKE scheme. Figure 4.3 describes our construction of a TRE scheme. As we have already mentioned, the hardness parameter T is not necessary as input for Dec, hence we leave it out of the construction.

$\underline{\mathsf{Setup}(1^\lambda, T)}$	$\underline{\mathsf{Solve}(\mathsf{pp}_d)}$
$(Z, s) \leftarrow \mathtt{TLP.Gen}(1^\lambda, T)$	$s \leftarrow \mathtt{TLP.Solve}(\mathsf{pp}_d)$
$(\mathsf{pk}, \mathsf{sk}) \leftarrow \mathtt{PKE.Gen}(1^\lambda; s)$	return s
return $\mathsf{pp}_e := \mathsf{pk}, \mathsf{pp}_d := Z$	
$\underline{\mathsf{Enc}(\mathsf{pp}_e, m)}$	$\underline{\mathsf{Dec}(s, c)}$
return $c \leftarrow \mathtt{PKE.Enc}(\mathsf{pp}_e, m)$	$(\mathsf{pk}, \mathsf{sk}) \leftarrow \mathtt{PKE.Gen}(1^\lambda; s)$
	return $m \leftarrow \mathtt{PKE.Dec}(\mathsf{sk}, c)$

Figure 4.3: Construction of $\mathtt{TRE}$

Theorem 4.4.1. *If* $\mathtt{TLP} = (\mathtt{TLP.Gen}, \mathtt{TLP.Solve})$ *is a secure time-lock puzzle with gap* ϵ *and* $\mathtt{PKE} = (\mathtt{PKE.Gen}, \mathtt{PKE.Enc}, \mathtt{PKE.Dec})$ *is an IND-CPA-secure public-key encryption scheme, then for any* $\underline{\epsilon} < \epsilon$, $\mathtt{TRE} = (\mathsf{Setup}, \mathsf{Solve}, \mathsf{Enc}, \mathsf{Dec})$ *defined in Figure 4.3 is a secure timed-release encryption scheme with gap* $\underline{\epsilon}$.

Proof. Observe that correctness of the scheme is directly implied by the correctness of the PKE scheme and the TLP.

To prove security we define two games G_0 and G_1 and show that these are computationally indistinguishable. For $i \in \{0, 1\}$ we denote by $\mathsf{G}_i = 1$ the event that the adversary $\mathcal{A} = \{\mathcal{A}_\lambda\}_{\lambda \in \mathbb{N}}$ outputs b' in game G_i such that $b = b'$.

Game 0. Game G_0 corresponds to the original security experiment, where we use the Setup directly from our construction.

Game 1. In G_1 we replace Setup with the alternative setup algorithm Setup' from Figure 4.4, in which we sample s' uniformly at random from S and use it as randomness for $\mathtt{PKE.Gen}$.

$\underline{\mathsf{Setup}'(1^\lambda, T)}$
$(Z, s) \leftarrow \mathtt{TLP}.\mathsf{Gen}(1^\lambda, T)$; $s' \xleftarrow{\$} S$
$(\mathsf{pk}, \mathsf{sk}) \leftarrow \mathtt{PKE}.\mathsf{Gen}(1^\lambda; s')$
return $\mathsf{pp}_e := \mathsf{pk}, \mathsf{pp}_d := Z$

Figure 4.4: Alternative setup

Let $\tilde{T}_{\mathtt{TLP}}(\lambda)$ be the polynomial whose existence is guaranteed by the security of TLP. Let $\mathsf{poly}_{\mathtt{PKE}}(\lambda)$ be the fixed polynomial which bounds the time required to run $\mathtt{PKE}.\mathsf{Gen}$ and $\mathtt{PKE}.\mathsf{Enc}$. Set $\underline{T} := (\mathsf{poly}_{\mathtt{PKE}}(\lambda))^{1/\epsilon}$. Set $\tilde{T}_{\mathtt{TRE}} := \max(\tilde{T}_{\mathtt{TLP}}, \underline{T})$.

Lemma 4.4.1. *From any polynomial-size adversary $\mathcal{A} = \{\mathcal{A}_\lambda\}_{\lambda\in\mathbb{N}}$, where the depth of $\mathcal{A}_\lambda$ is bounded from above by $T^\epsilon(\lambda)$ for some $T(\cdot) \geq \tilde{T}_{\mathtt{TRE}}(\cdot)$ we can construct a polynomial-size adversary $\mathcal{B} = \{\mathcal{B}_\lambda\}_{\lambda\in\mathbb{N}}$, where the depth of $\mathcal{B}_\lambda$ is bounded by $T^\epsilon(\lambda)$ with*

$$\mathbf{Adv}^{\mathtt{TLP}}_{\mathcal{B}} = |\Pr[\mathsf{G}_0 = 1] - \Pr[\mathsf{G}_1 = 1]|\,.$$

To prove this lemma we construct an adversary $\mathcal{B}$ as follows.

The adversary $\mathcal{B}_\lambda(Z, s)$:

1. Runs $(\mathsf{pk}, \mathsf{sk}) \leftarrow \mathtt{PKE}.\mathsf{Gen}(1^\lambda, s)$.
2. Runs $(m_0, m_1) \leftarrow \mathcal{A}_\lambda(\mathsf{pk}, Z)$.
3. Samples $b \xleftarrow{\$} \{0, 1\}$ uniformly at random and computes $c \leftarrow \mathtt{PKE}.\mathsf{Enc}(\mathsf{pk}, m_b)$.
4. Finally, runs $b' \leftarrow \mathcal{A}_\lambda(c)$ and returns the truth value of $b' = b$.

Note that if s is the solution of the puzzle Z, then $\mathcal{B}$ simulates G_0 perfectly. If s is random, then it simulates G_1 perfectly. Moreover, $\mathcal{B}$ meets the depth constraint:

$$\mathsf{depth}(\mathcal{B}_\lambda) = \mathsf{poly}_{\mathtt{PKE}}(\lambda) + \mathsf{depth}(\mathcal{A}_\lambda) \leq \underline{T}^\epsilon(\lambda) + T^\epsilon(\lambda) \leq 2T^\epsilon(\lambda) = o(T^\epsilon(\lambda)).$$

Also $T(\cdot) \geq \tilde{T}_{\mathtt{TRE}}(\cdot) \geq \tilde{T}_{\mathtt{TLP}}(\cdot)$. Thus, we can conclude that

$$|\Pr[\mathsf{G}_0 = 1] - \Pr[\mathsf{G}_1 = 1]| = \mathbf{Adv}^{\mathtt{TLP}}_{\mathcal{B}}$$

as required.

Now we can show that we can construct an adversary $\mathcal{B}'$ against the security of the PKE scheme.

Lemma 4.4.2. *From any polynomial-size adversary $\mathcal{A} = \{\mathcal{A}_\lambda\}_{\lambda\in\mathbb{N}}$ we can construct a polynomial-size adversary $\mathcal{B}' = \{\mathcal{B}'_\lambda\}_{\lambda\in\mathbb{N}}$ such that*

$$\mathbf{Adv}^{\mathtt{PKE}}_{\mathcal{B}'} = \left|\Pr[\mathsf{G}_1 = 1] - \frac{1}{2}\right|.$$

The proof of this lemma is straightforward. Notice that in G_1 we use fresh randomness which is independent of the puzzle Z. We construct $\mathcal{B}'$ as follows.

The adversary $\mathcal{B}'_\lambda(\mathsf{pk})$:

1. Runs $(Z, s) \leftarrow \mathtt{TLP}.\mathsf{Gen}(1^\lambda, T)$.
2. Runs $(m_0, m_1) \leftarrow \mathcal{A}_\lambda(\mathsf{pk}, Z)$.
3. Outputs (m_0, m_1) to its experiment and receives c.
4. Finally returns $b' \leftarrow \mathcal{A}_\lambda(c)$.

Adversary $\mathcal{B}'$ simulates G_1 perfectly. Since $\mathcal{A}$ has polynomially-bounded size, this proves the lemma. By combining Lemmas 4.4.1 and 4.4.2 we obtain following:

$$\begin{aligned}\mathbf{Adv}^{\mathtt{TRE}}_{\mathcal{A}} &= \left|\Pr[\mathsf{G}_0 = 1] - \frac{1}{2}\right| \leq |\Pr[\mathsf{G}_0 = 1] - \Pr[\mathsf{G}_1 = 1]| + \left|\Pr[\mathsf{G}_1 = 1] - \frac{1}{2}\right| \\ &= \mathbf{Adv}^{\mathtt{TLP}}_{\mathcal{B}} + \mathbf{Adv}^{\mathtt{PKE}}_{\mathcal{B}'},\end{aligned}$$

which concludes the proof. □

4.4.2 Sequential TRE

Next, let $\mathtt{sTLP} = (\mathtt{sTLP}.\mathsf{Gen}, \mathtt{sTLP}.\mathsf{Solve})$ be a sequential TLP in the sense of Definition 4.3.1 and let $\mathtt{PKE} = (\mathtt{PKE}.\mathsf{Gen}, \mathtt{PKE}.\mathsf{Enc}, \mathtt{PKE}.\mathsf{Dec})$ be a PKE scheme. Let $\mathsf{F} : \mathbb{N} \times S \to Y$ be a function that maps the hardness parameter space $\mathbb{N}$ and the solution space S of $\mathtt{TLP}$ to the randomness space of algorithm $\mathtt{PKE}.\mathsf{Gen}$. Our constructions of a sequential TRE scheme $\mathtt{TRE} = (\mathsf{Setup}, \mathsf{Enc}, \mathsf{Solve}, \mathsf{Dec})$ is given in Figure 4.5.

$\underline{\mathsf{Setup}(1^\lambda, (T_i)_{i\in[n]})}$	$\underline{\mathsf{Solve}(\mathsf{pp}_{d,i}, s_{i-1})}$
$(Z_i, s_i)_{i\in[n]} \leftarrow \mathtt{sTLP}.\mathsf{Gen}(1^\lambda, (T_i)_{i\in[n]})$	$s_i \leftarrow \mathtt{sTLP}.\mathsf{Solve}(\mathsf{pp}_{d,i}, s_{i-1})$
$((\mathsf{pk}_i, \mathsf{sk}_i) \leftarrow \mathtt{PKE}.\mathsf{Gen}(1^\lambda; \mathsf{F}(T_i, s_i)))_{i\in[n]}$	return s_i
return $(\mathsf{pp}_{e,i} := \mathsf{pk}_i, \mathsf{pp}_{d,i} := Z_i)_{i\in[n]}$	
$\underline{\mathsf{Enc}(\mathsf{pp}_{e,i}, m)}$	$\underline{\mathsf{Dec}(T_i, s_i, c)}$
return $c \leftarrow \mathtt{PKE}.\mathsf{Enc}(\mathsf{pp}_{e,i}, m)$	$(\mathsf{pk}_i, \mathsf{sk}_i) \leftarrow \mathtt{PKE}.\mathsf{Gen}(1^\lambda; \mathsf{F}(T_i, s_i))$
	return $m \leftarrow \mathtt{PKE}.\mathsf{Dec}(\mathsf{sk}_i, c)$

Figure 4.5: Construction of sequential $\mathtt{TRE}$

Theorem 4.4.2. *If* $\mathtt{sTLP} = (\mathtt{sTLP.Gen}, \mathtt{sTLP.Solve})$ *is a secure sequential time-lock puzzle with gap* ϵ *and w.r.t. function* F *and* $\mathtt{PKE} = (\mathtt{PKE.Gen}, \mathtt{PKE.Enc}, \mathtt{PKE.Dec})$ *is an IND-CPA-secure public-key encryption scheme, then for any* $\underline{\epsilon} < \epsilon$, $\mathtt{TRE} = (\mathsf{Setup}, \mathsf{Enc}, \mathsf{Solve}, \mathsf{Dec})$ *defined in Figure 4.5 is a secure sequential timed-release encryption with gap* $\underline{\epsilon}$.

Proof. Note that correctness of the scheme is directly implied by the correctness of the PKE scheme and the sequential TLP.

To prove security we define two games G_0 and G_1. For $j \in \{0, 1\}$ we denote by $\mathsf{G}_j = 1$ the event that the adversary $\mathcal{A}_i = \{\mathcal{A}_{i,\lambda}\}_{\lambda\in\mathbb{N}}$ outputs b' in the game G_j such that $b = b'$.

Game 0. Game G_0 is the original security experiment with scheme $\mathtt{TRE}$.

Game 1. In G_1 we replace Setup with the alternative setup algorithm Setup' from Figure 4.6, which takes as input i. For all $j \in [n] \setminus \{i\}$ we produce keys pk_j using $\mathsf{F}(T_j, s_j)$. The remaining key pk_i is generated using fresh randomness sampled uniformly from the image of the function F.

$\underline{\mathsf{Setup}'(1^\lambda, (T_j)_{j\in[n]}, i)}$
$(Z_j, s_j)_{j\in[n]} \leftarrow \mathtt{sTLP.Gen}(1^\lambda, (T_j)_{j\in[n]})$
$((\mathsf{pk}_j, \mathsf{sk}_j) \leftarrow \mathtt{PKE.Gen}(1^\lambda; \mathsf{F}(T_j, s_j)))_{j\in\{[n]\setminus\{i\}\}}$
$y_i \xleftarrow{\$} Y, (\mathsf{pk}_i, \mathsf{sk}_i) \leftarrow \mathtt{PKE.Gen}(1^\lambda; y_i)$
return $(\mathsf{pp}_{e,j} := \mathsf{pk}_j, \mathsf{pp}_{d,j} := Z_j)_{j\in[n]}$

Figure 4.6: Alternative setup

Let $\tilde{T}_{\mathtt{sTLP}}(\lambda)$ be the polynomial whose existence is guaranteed by the security of sTLP. Let $\mathsf{poly}_{\mathtt{PKE}}(\lambda)$ be the fixed polynomial which bounds the time required to run $\mathtt{PKE.Gen}$ n-times and to run $\mathtt{PKE.Enc}$ once. Set $\underline{T} := (\mathsf{poly}_{\mathtt{PKE}}(\lambda))^{1/\underline{\epsilon}}$. Set $\tilde{T}_{\mathtt{TRE}} := \max(\tilde{T}_{\mathtt{sTLP}}, \underline{T})$.

Lemma 4.4.3. *For any* n *which is polynomial in* λ, *for any set of polynomials* $(T_j(\cdot))_{j\in[n]}$ *fulfilling that* $\forall j \in [n]$ *holds* $T_j(\cdot) \geq \tilde{T}_{\mathtt{TRE}}(\cdot)$, *for any* $i \in [n]$, *from any polynomial-size adversary* $\mathcal{A}_i = \{\mathcal{A}_{i,\lambda}\}_{\lambda\in\mathbb{N}}$, *where the depth of* $\mathcal{A}_{i,\lambda}$ *is bounded from above by* $T_i^{\underline{\epsilon}}(\lambda)$, *we can construct a polynomial-size adversary* $\mathcal{B}_i = \{\mathcal{B}_{i,\lambda}\}_{\lambda\in\mathbb{N}}$ *whose depth is bounded from above by* $T_i^{\epsilon}(\lambda)$ *such that*

$$\mathbf{Adv}^{\mathtt{sTLP}}_{\mathcal{B}_i} = \left|\Pr[\mathsf{G}_0 = 1] - \Pr[\mathsf{G}_1 = 1]\right|.$$

To prove this claim we construct an adversary $\mathcal{B}_i$ as follows.
The adversary $\mathcal{B}_{i,\lambda}((Z_j, y_j)_{j\in[n]})$:

1. Runs $((\mathsf{pk}_j, \mathsf{sk}_j) \leftarrow \mathtt{PKE}.\mathsf{Gen}(1^\lambda; y_j))_{j\in[n]}$.

2. Runs $(m_0, m_1) \xleftarrow{\$} \mathcal{A}_{i,\lambda}((\mathsf{pk}_j, Z_j)_{j\in[n]})$.

3. Samples $b \xleftarrow{\$} \{0,1\}$ uniformly at random and computes $c \leftarrow \mathtt{PKE}.\mathsf{Enc}(\mathsf{pk}_i, m_b)$.

4. Runs $b' \leftarrow \mathcal{A}_{i,\lambda}(c)$ and returns the truth value of $b' = b$.

If $y_j = \mathsf{F}(T_j, s_j)$ for all $j \in [n]$, then $\mathcal{B}_i$ simulates G_0 perfectly. If the y_i is random, then it simulates G_1 perfectly. Therefore we obtain

$$\Pr[\mathsf{G}_0 = 1] = \Pr\left[\mathsf{ExpsTLP}^0_{\mathcal{B}_i}(1^\lambda) = 1\right] \text{ and } \Pr[\mathsf{G}_1 = 1] = \Pr\left[\mathsf{ExpsTLP}^1_{\mathcal{B}_i}(1^\lambda) = 1\right]$$

and thus

$$\begin{aligned}\mathbf{Adv}^{\mathtt{sTLP}}_{\mathcal{B}_i} &= \left|\Pr\left[\mathsf{ExpsTLP}^0_{\mathcal{B}_i}(1^\lambda) = 1\right] - \Pr\left[\mathsf{ExpsTLP}^1_{\mathcal{B}_i}(1^\lambda) = 1\right]\right| \\ &= \left|\Pr[\mathsf{G}_0 = 1] - \Pr[\mathsf{G}_1 = 1]\right|.\end{aligned}$$

Moreover, $\mathcal{B}_i$ fulfils the depth constraint:

$$\mathsf{depth}(\mathcal{B}_{i,\lambda}) = \mathsf{poly}_{\mathtt{PKE}}(\lambda) + \mathsf{depth}(\mathcal{A}_{i,\lambda}) \leq \underline{T}^\epsilon_i(\lambda) + T^\epsilon_{\bar{i}}(\lambda) \leq 2T^\epsilon_{\bar{i}}(\lambda) = o(T^\epsilon_i(\lambda)).$$

Also $T_i(\cdot) \geq \tilde{T}_{\mathtt{TRE}}(\cdot) \geq \tilde{T}_{\mathtt{sTLP}}(\cdot)$ as required.

Finally, we show that we can construct an adversary against security of the PKE scheme.

Lemma 4.4.4. *For any n which is polynomial in λ, for any set of polynomials $(T_j(\cdot))_{j\in[n]}$ fulfilling that $\forall j \in [n]$ holds $T_j(\cdot) \geq \tilde{T}_{\mathtt{TRE}}(\cdot)$, for any $i \in [n]$, from any polynomial-size adversary $\mathcal{A}_i = \{\mathcal{A}_{i,\lambda}\}_{\lambda\in\mathbb{N}}$ we can construct a polynomial-size adversary $\mathcal{B}' = \{\mathcal{B}'_\lambda\}_{\lambda\in\mathbb{N}}$ such that*

$$\mathbf{Adv}^{\mathtt{PKE}}_{\mathcal{B}'} = \left|\Pr[\mathsf{G}_1 = 1] - \frac{1}{2}\right|.$$

The proof is essentially identical to the corresponding step from proof of Theorem 4.4.1, adopted to the sequential setting. We construct $\mathcal{B}'$ as follows.

The adversary $\mathcal{B}'_\lambda(\mathsf{pk})$:

1. Runs $(Z_j, s_j)_{j\in[n]} \leftarrow \mathtt{sTLP}.\mathsf{Gen}(1^\lambda, (T_j)_{j\in[n]})$.

2. For all $j \in \{[n] \setminus \{i\}\}$ sets $\mathsf{pk}_j := \mathtt{PKE}.\mathsf{Gen}(1^\lambda; \mathsf{F}(T_j, s_j))$. The i-th public key is defined as $\mathsf{pk}_i := \mathsf{pk}$.

3. Runs adversary $(m_0, m_1) \leftarrow \mathcal{A}_{i,\lambda}((\mathsf{pk}_j, Z_j)_{j\in[n]})$.

4. Sends (m_0, m_1) to the challenger and receives c.

5. Finally, it returns $b' \leftarrow \mathcal{A}_{i,\lambda}(c)$.

Note that adversary $\mathcal{B}'$ simulates G_1 perfectly, which yields Lemma 4.4.4. Finally, combining Lemmas 4.4.3 and 4.4.4 we obtain

$$\begin{aligned}\mathbf{Adv}^{\mathtt{TRE}}_{\mathcal{A}_i} &= \left|\Pr[\mathsf{G}_0 = 1] - \frac{1}{2}\right| = |\Pr[\mathsf{G}_1 = 1] - \Pr[\mathsf{G}_2 = 1]| + \left|\Pr[\mathsf{G}_2 = 1] - \frac{1}{2}\right| \\ &\leq \mathbf{Adv}^{\mathtt{sTLP}}_{\mathcal{B}_i} + \mathbf{Adv}^{\mathtt{PKE}}_{\mathcal{B}'},\end{aligned}$$

which concludes the proof. □

4.4.3 Properties of TRE Construction

In this section, we discuss some interesting properties which are achievable by our generic construction of (sequential) timed-release encryption.

Homomorphic Timed-Release Encryption.

Our construction of sequential timed-release encryption can be simply adjusted to the homomorphic setting. At first we define the notion of homomorphic sequential timed-release encryption and then we explain how we can achieve it.

Definition 4.4.3. A homomorphic sequential timed-release encryption scheme $\mathtt{HTRE}$ for a circuit class $\mathcal{C} = \{\mathcal{C}_\lambda\}_{\lambda\in\mathbb{N}}$ is a sequential TRE scheme with one additional algorithm Eval defined as follows:

- $c \leftarrow \mathsf{Eval}(\mathsf{pp}_{e,i}, C, (c_j)_{j\in[l]})$ is a probabilistic algorithm that takes as input public parameters pp_e, a circuit $C \in \mathcal{C}_\lambda$ and a set of ciphertexts $(c_j)_{j\in[l]}$, which were all produced using the same public encryption parameters $\mathsf{pp}_{e,i}$, and outputs a ciphertext c.

A HTRE scheme is *correct* if for all $\lambda, n \in \mathbb{N}$, for all sets of hardness parameters $(T_j)_{j\in[n]}$ such that $\forall j \in [n-1] : T_j < T_{j+1}$, for all $i \in [n]$, all circuits $C \in \mathcal{C}_\lambda$, all inputs $(m_1, \ldots, m_\ell)$, all $(\mathsf{pp}_{e,i}, \mathsf{pp}_{d,i})$ in support of $\mathsf{Setup}(1^\lambda, (T_j)_{j\in[n]})$, and all $c_{i,j}$ in the support of $\mathsf{Enc}(\mathsf{pp}_{e,i}, m_j)$, the following conditions are satisfied:

- There exists a negligible function $\mathsf{negl}(\cdot)$ such that

$$\Pr[\mathsf{Dec}(\mathsf{Solve}(\mathsf{pp}_{d,i}, s_i), \mathsf{Eval}(\mathsf{pp}_{e,i}, C, (c_{i,j})_{j\in[\ell]}))) \neq C((m_j)_{j\in[\ell]})] \leq \mathsf{negl}(\lambda).$$

A HTRE scheme is *secure*, if it is a secure $\mathtt{TRE}$ scheme.

Moreover, homomorphic timed-release encryption should also satisfy compactness. We say that HTRE is *compact* if there exist two polynomials $\mathsf{poly}(\cdot)$ and $\mathsf{poly}'(\cdot)$ such that for all $\lambda, n \in \mathbb{N}$, for all sets of hardness parameters $(T_j)_{j\in[n]}$

such that $\forall j \in [n-1] : T_j < T_{j+1}$, for all $i \in [n]$, all circuits $C \in \mathcal{C}_\lambda$, all inputs $(m_1, \ldots, m_l)$, all $(\mathsf{pp}_{e,i}, \mathsf{pp}_{d,i})$ in support of $\mathsf{Setup}(1^\lambda, (T_i)_{i\in[n]})$, and all $c_{i,j}$ in the support of $\mathsf{Enc}(\mathsf{pp}_{e,i}, m_j)$, the following conditions are satisfied:

- $\left|\mathsf{Eval}(\mathsf{pp}_{e,i}, C, (c_{i,j})_{j\in[l]})\right| = \mathsf{poly}(\lambda, |C((m_j)_{j\in[l]})|)$.
- The runtime of $\mathsf{Eval}(\mathsf{pp}_{e,i}, C, (c_{i,j})_{j\in[l]})$ is bounded by $\mathsf{poly}'(\lambda, |C|)$.

To obtain homomorphic sequential timed-release encryption it is sufficient to use homomorphic encryption in the construction depicted in Figure 4.5.

Theorem 4.4.3. *Let* $\mathtt{TLP} = (\mathtt{TLP.Gen}, \mathtt{TLP.Solve})$ *be a secure time-lock puzzle and* $\mathtt{PKE} = (\mathtt{PKE.Gen}, \mathtt{PKE.Enc}, \mathtt{PKE.Dec}, \mathtt{PKE.Eval})$ *be a secure homomorphic encryption scheme. Let* $(\mathsf{Setup}, \mathsf{Enc}, \mathsf{Solve}, \mathsf{Dec})$ *be as defined in Figure 4.5 and define* Eval *as follows:*

$$\underline{\mathsf{Eval}(\mathsf{pp}_{e,i}, C, (c_j)_{j\in[l]})}$$
$$c \leftarrow \mathtt{PKE.Eval}(\mathsf{pp}_{e,i}, C, (c_j)_{j\in[l]}))$$

Then $(\mathsf{Setup}, \mathsf{Enc}, \mathsf{Solve}, \mathsf{Dec}, \mathsf{Eval})$ *is a secure homomorphic sequential timed-release encryption.*

Proof. Notice that correctness and security are directly implied by the previous proof. The reason is that TRE and HTRE have the same correctness and security definitions and this relation holds also between standard public-key encryption and homomorphic encryption. It remains to prove compactness. The first condition of compactness is trivially fulfilled by the compactness of the homomorphic encryption scheme. Moreover, notice that the Eval algorithm is independent of T. Therefore, the second condition of compactness is also satisfied. □

Publicly Verifiable Timed-Release Encryption.

The notion of publicly verifiable time-lock puzzles was suggested in [EFKP20]. This type of TLPs enables public verification if the provided solution of the puzzle is indeed the real solution. In the case that a puzzle does not have a solution, it should be also possible to verify this fact. In more detail, a publicly verifiable time-lock puzzle consist of $(\mathsf{Gen}, \mathsf{Solve}, \mathsf{Vrfy})$ where Solve additionally with solution s outputs a proof π that solution s is correct and Vrfy takes as input a puzzle Z, a solution s together with a proof π and outputs 0 or 1.

We observe that our construction of timed-release encryption provides also a similar form of public verifiability with respect to public parameters $\mathsf{pp}_e, \mathsf{pp}_d$ produced by the setup. At first, notice that because we generate a time-lock puzzle

in the setup which is trusted and we assume perfect correctness of the underlying TLP, the case that TLP does not have a solution can not happen. Therefore, if someone claims to provide a candidate solution s for a puzzle generated in the setup, it is sufficient to run the Gen algorithm of the underlying public-key encryption scheme with the randomness s and check if the resulting public-key matches the public-key which was produced during the setup. Hence the Vrfy algorithm is simply

$$\begin{aligned} &\underline{\mathsf{Vrfy}(\mathsf{pp}_e, \mathsf{pp}_d, s, \pi)} \\ &(\mathsf{pk}, \mathsf{sk}) \leftarrow \mathtt{PKE.Gen}(1^\lambda; s) \\ &\text{return } \mathsf{pp}_e = \mathsf{pk} \end{aligned}$$

where s is a candidate solution, $\mathsf{pp}_e, \mathsf{pp}_d$ are public encryption/decryption parameters from our TRE construction and $\pi := \bot$. Notice that if a public-key encryption scheme is perfectly complete, then for one pk there can not be different secret keys output by $\mathtt{PKE.Gen}$ which would behave differently in their decryption behaviour. In this case, the correctness of the decryption is guaranteed. There might also be another $s' \neq s$ which could generate the same public/secret key pair, but as long as we are interested only in the correctness of decryption, this case is not interesting for us. If one would like to enforce that only the real solution of the puzzle should pass the verification, then there are two options. Let us denote by R the randomness space of $\mathtt{PKE.Gen}$ algorithm for which it produces different keys. If $S \subseteq R$, then there does not exist another value $s' \neq s$ which would yield the same public-key. If $R \subset S$, then one can apply a collision-resistant hash function and then finding another $s' \neq s$ such that $H(s) = H(s')$ would be infeasible.

Optimal amortized costs.

As we have already discussed in the introduction, our timed-release encryption scheme achieves the "solve one, get many for free" property which in other words means it allows to decrypt a large number of ciphertexts by solving only one puzzle. This is in contrast to homomorphic time-lock puzzles which are useful in applications where one is interested only in the result of some computation applied to messages encrypted in the corresponding puzzles, however, if access to all messages is needed, then indeed all puzzles have to be solved independently when relying on HTLPs.

We remark that our scheme allows to decrypt an arbitrary number of messages which are encrypted using the same public encryption parameters. Hence for n independently time-locked messages $m_1, \ldots, m_n$ our scheme is the first one to

achieve an optimal amortized cost of decryption per ciphertext of

$$\frac{n \cdot T_{\mathsf{PKE.Dec}} + T_{\mathsf{TLP}}}{n}$$

where $T_{\mathsf{PKE.Dec}}$ is the time required to run the decryption algorithm of $\mathtt{PKE}$ scheme and T_{TLP} is the time required to solve the puzzle. Note that this approaches $T_{\mathsf{PKE.Dec}}$ with increasing n. This also holds for our sequential TRE scheme.

Sequential TRE with public servers.

Another interesting property achievable by our notion of sequential timed-release encryption is the possibility to combine it with some public server whose task is to compute solutions s_i which correspond to hardness parameters T_i. Here is particularly useful that the solution s_i can be computed from the solution s_{i-1} in time which is determined by the hardness parameter $T_i - T_{i-1}$ and therefore one sever is sufficient to compute all solutions. The validity of published solutions can be publicly verified as explained earlier and consequently the solutions can be used to decrypt an arbitrary number of ciphertexts. Moreover, the server is independent of these ciphertexts, which is not achievable by prior constructions. Hence, the burden of executing the expensive computation is delegated to a server and for decrypting parties it is sufficient to wait until the required solution is published.

At this point, it is useful to remark that the above-mentioned proposal is indeed different from the notion of timed-release encryption which relies on trusted agents. The main difference is that we only require a trusted setup, however, the latter one requires a trusted party that is not only involved in running the setup but must be trusted the whole time until confidentially releasing the secret keys at a specified point in time. We note that when combining sequential TRE with a public server, there is no way for the server to reveal solutions before the specified time has passed.

Our notion of (sequential) TRE relies on a trusted setup which is not required by standard TLPs. This might be problematic especially in scenarios where the trusted setup should be performed by a third-party server or in the case when the server becomes unavailable. To mitigate these issues, one could run n servers in parallel. Messages can be split into shares using a (k, n)-threshold secret sharing scheme (e.g., [Sha79]) and public parameters of each server would be used to encrypt a share of the message. The security of the threshold secret sharing scheme guarantees that the message remains hidden even if $k - 1$ servers would collude. At the same time, k shares are sufficient to recover the message and therefore even if $(n - k)$ servers are unavailable, messages would be still recoverable.

4.4.4 Integrating Timed-Release Features into Functional Encryption

Our timed-release encryption enables us to incorporate timed-release features in functional encryption (FE). For details regarding FE see Section 2.3.5 where we also discuss one concrete variant of functional encryption, namely identity-based encryption (IBE). At first, we formally define a new primitive which combines properties of both functional and timed-release encryption. Then we construct this primitive generically and finally, we discuss one possible application.

Timed-Release Functional Encryption.

We introduce the notion of a (sequential) timed-release functional encryption (TRFE) scheme. Similarly to FE, TRFE provides a public key pk that is used for encryption of arbitrary messages and a master secret key msk which is associated with a class of functions $\mathcal{F}$. However, in TRFE the master secret key msk is used to generate a decryption key dk_i which is associated with both a function $f \in \mathcal{F}$ and a hardness parameter T_i. In order to decrypt, one has to compute the solution s_i which corresponds to the hardness parameter T_i. Then using the solution s_i and decryption key dk_i, one can obtain value $f(x)$ from a ciphertext c that is encryption of the message x under pk.

Definition 4.4.4. A (sequential) TRFE scheme TRFE for a class of functions $\mathcal{F} = \{\mathcal{F}_\lambda\}_{\lambda\in\mathbb{N}}$ consists of five efficient algorithms $(\mathsf{Setup}, \mathsf{KeyGen}, \mathsf{Enc}, \mathsf{Solve}, \mathsf{Dec})$. Let $\mathcal{X}_\lambda$ be the input space of $\mathcal{F}_\lambda$ and let $\mathcal{Y}_\lambda$ be the output space of $\mathcal{F}_\lambda$.

- $(\mathsf{pk}, \mathsf{msk}, (\mathsf{pp}_{e,j}, \mathsf{pp}_{d,j})_{j\in[n]}) \leftarrow \mathsf{Setup}(1^\lambda, \mathcal{F}, (T_j)_{j\in[n]})$ is a probabilistic algorithm that takes as input the security parameter 1^λ, a class of functions $\mathcal{F}$, and hardness parameters $(T_j)_{j\in[n]}$ with $T_1 < \ldots < T_n$ and outputs a public key pk, a master secret key msk, and public encryption and decryption parameters $\mathsf{PP} := (\mathsf{pp}_{e,j}, \mathsf{pp}_{d,j})_{j\in[n]}$. We require that Setup runs in time at most $\mathsf{poly}((\log T_j)_{j\in[n]}, \lambda)$.

- $\mathsf{dk}_i \leftarrow \mathsf{KeyGen}(\mathsf{msk}, (\mathsf{pp}_{e,j})_{j\in[n]}, f, i)$ is a probabilistic algorithm that takes as input the master secret key msk, public encryption parameters $(\mathsf{pp}_{e,j})_{j\in[n]}$, a function $f \in \mathcal{F}_\lambda$, and index $i \in [n]$ and outputs a decryption key dk_i.

- $c \leftarrow \mathsf{Enc}(\mathsf{pk}, x)$ is a probabilistic algorithm that takes as input a public key pk and a message $x \in \mathcal{X}_\lambda$ and outputs a ciphertext c.

- $s_i \leftarrow \mathsf{Solve}(\mathsf{pp}_{d,i}, s_{i-1})$ is a deterministic algorithm that takes as input public decryption parameters $\mathsf{pp}_{d,i}$ and the solution from a previous iteration s_{i-1}, where $s_0 = \perp$, and outputs the solution s_i. We require that Solve runs

in time at most $(T_i - T_{i-1}) \cdot \mathsf{poly}(\lambda)$ where T_i and T_{i-1} are the associated hardness parameters for s_i and s_{i-1}, respectively.

- $f(x') \leftarrow \mathsf{Dec}(\mathsf{dk}_i, T_i, s_i, c)$ is a deterministic algorithm that takes as input a decryption key dk_i, a hardness parameter T_i, a solution s_i, function f, and ciphertext c and outputs $f(x') \in \mathcal{Y}_\lambda \cup \{\perp\}$.

We say a sequential timed-release functional encryption scheme for class of functions $\mathcal{F}$ is *correct* if for all $\lambda, n \in \mathbb{N}$, for all sets of hardness parameters $(T_j)_{j\in[n]}$ such that $T_1 < \ldots < T_n$, for all $f \in \mathcal{F}_\lambda$, $i \in [n]$,$x \in \mathcal{X}$, it holds:

$$\Pr\left[\mathsf{Dec}(\mathsf{dk}_i, T_i, s_i, c) = f(x) \ : \begin{array}{r}(\mathsf{pk}, \mathsf{msk}, \mathsf{PP}) \leftarrow \mathsf{Setup}(1^\lambda, \mathcal{F}, (T_j)_{j\in[n]}) \\ \mathsf{dk}_i \leftarrow \mathsf{KeyGen}(\mathsf{msk}, (\mathsf{pp}_{e,j})_{j\in[n]}, f, i) \\ s_i \leftarrow \mathsf{Solve}(\mathsf{pp}_{d,i}, s_{i-1}), s_0 := \perp \\ c \leftarrow \mathsf{Enc}(\mathsf{pk}, x)\end{array}\right] = 1.$$

Definition 4.4.5. A (sequential) TRFE scheme TRFE is *IND-CPA secure with gap* $0 < \epsilon < 1$ if for all polynomials n in λ there exists a polynomial $\tilde{T}(\cdot)$ such that for all sets of polynomials $(T_j)_{j\in[n]}$ fulfilling that $\forall j \in [n] : T_j(\cdot) \geq \tilde{T}(\cdot)$, for all $i \in [n]$ and every polynomial-size adversary $\mathcal{A}_i = \{\mathcal{A}_{i,\lambda}\}_{\lambda\in\mathbb{N}}$, where the depth of $\mathcal{A}_{i,\lambda}$ is bounded from above by $T_i^\epsilon(\lambda)$, there exists a negligible function $\mathsf{negl}(\cdot)$ such that for all $\lambda \in \mathbb{N}$ it holds

$$\mathbf{Adv}_{\mathcal{A}_i}^{\mathsf{TRFE}} = \left|\Pr\left[b = b' : \begin{array}{r}(\mathsf{pk}, \mathsf{msk}, \mathsf{PP}) \leftarrow \mathsf{Gen}(1^\lambda, \mathcal{F}, (T_j)_{j\in[n]}) \\ (x_0, x_1) \leftarrow \mathcal{A}_{i,\lambda}^{\mathsf{KEYGEN}(\cdot,\cdot)}(\mathsf{pk}, \mathsf{PP}) \\ b \leftarrow \{0,1\}, c \leftarrow \mathsf{Enc}(\mathsf{pk}, x_b) \\ b' \leftarrow \mathcal{A}_{i,\lambda}^{\mathsf{KEYGEN}(\cdot,\cdot)}(c)\end{array}\right] - \frac{1}{2}\right| \leq \mathsf{negl}(\lambda),$$

where the oracle $\mathsf{KEYGEN}(f, h)$ returns $\mathsf{KeyGen}(\mathsf{msk}, (\mathsf{pp}_{e,j})_{j\in[n]}, f, h)$ with the restriction that if $\mathcal{A}_\lambda$ queries the oracle with function $f \in \mathcal{F}$ at index $h \in [i-1]$, then it must hold $f(x_0) = f(x_1)$.

Construction of TRFE.

Let $\mathtt{TRE} = (\mathtt{TRE.Setup}, \mathtt{TRE.Solve}, \mathtt{TRE.Enc}, \mathtt{TRE.Dec})$ be a(sequential) timed-release encryption scheme and $\mathtt{FE} = (\mathtt{FE.Gen}, \mathtt{FE.KeyGen}, \mathtt{FE.Enc}, \mathtt{FE.Dec})$ be a functional encryption scheme. We construct a timed-release functional encryption scheme $\mathsf{TRFE} = (\mathsf{Setup}, \mathsf{KeyGen}, \mathsf{Enc}, \mathsf{Solve}, \mathsf{Dec})$ as given in Figure 4.7. Let the message space of $\mathtt{TRE}$ be the functional secret key space of $\mathtt{FE}$ which is the output of $\mathtt{FE.KeyGen}$.

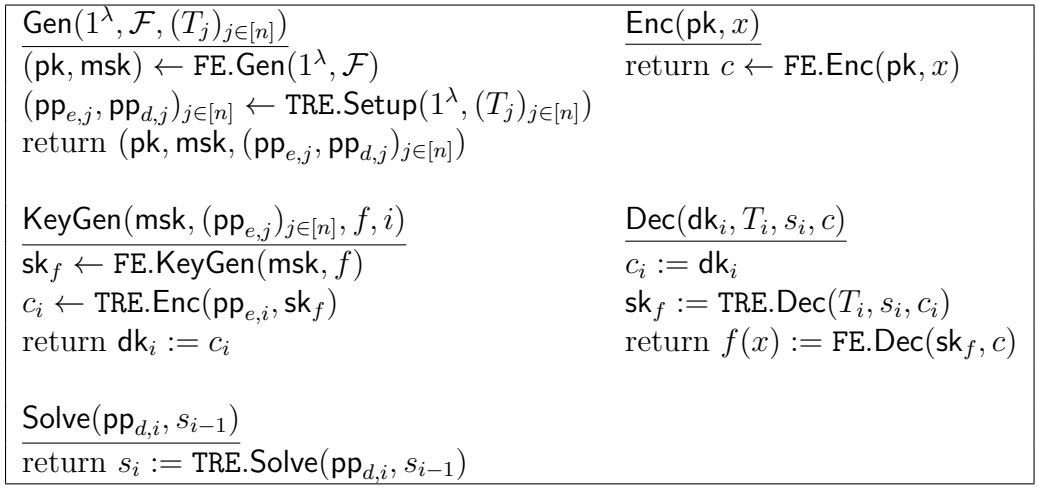

Figure 4.7: Construction of TRFE.

Theorem 4.4.4. *If* $\mathtt{TRE} = (\mathtt{TRE}.\mathsf{Setup}, \mathtt{TRE}.\mathsf{Solve}, \mathtt{TRE}.\mathsf{Enc}, \mathtt{TRE}.\mathsf{Dec})$ *is a secure sequential timed-release encryption scheme with gap* ϵ *and* $\mathtt{FE} = (\mathtt{FE}.\mathsf{Gen}, \mathtt{FE}.\mathsf{KeyGen}, \mathtt{FE}.\mathsf{Enc}, \mathtt{FE}.\mathsf{Dec})$ *is an IND-CPA-secure functional encryption scheme, then for any* $\underline{\epsilon} \leq \epsilon$, $\mathsf{TRFE} = (\mathsf{Setup}, \mathsf{KeyGen}, \mathsf{Enc}, \mathsf{Solve}, \mathsf{Dec})$ *defined in Figure 4.7 is an IND-CPA-secure timed-release functional encryption scheme with gap* $\underline{\epsilon}$.

Proof. Correctness of the TRFE scheme is implied by the correctness of the TRE scheme and the correctness of the FE scheme.

To prove security, we proceed in a sequence of games G_0 and G_1. For $j \in \{0, 1\}$ we denote by $\mathsf{G}_j = 1$ the event that the adversary $\mathcal{A}_i = \{\mathcal{A}_{i,\lambda}\}_{\lambda\in\mathbb{N}}$ outputs b' in game G_j such that $b = b'$. We denote the functional secret key space by $\mathcal{K}$. We remark that in the proof we use an alternative security definition for sequential TRE, where security is defined using two experiments and in one experiment adversary is always given an encryption of the message m_0, in the second experiment it is always given an encryption of m_1 (in the similar way how security of a sTLP is defined). This definition is equivalent to the original one.

Game 0. Game G_0 is the original security experiment.

Game 1. In G_1, all KEYGEN-queries for indices k with $i \leq k \leq n$ are answered by sampling a decryption key of an appropriate length from $\mathcal{K}$ uniformly at random and encrypting it using TRE.

Lemma 4.4.5. *Let q be the total number of* KEYGEN*-queries of type (f, k) where $i \leq k \leq n$. There exists a negligible function* $\mathsf{negl}(\cdot)$ *such that*

$$|\Pr[\mathsf{G}_0 = 1] - \Pr[\mathsf{G}_1 = 1]| \leq \mathsf{negl}(\lambda).$$

This can be shown by a hybrid argument where we go over all q queries in a step-by-step fashion. Since the proof is always the same depending on the index of the actual query, we show the proof for the ℓ-th query only. Hence we define a sequence of hybrid games $\mathsf{G}_{0,\ell}$ for $\ell \in \{0, 1, \ldots, q\}$. In $\mathsf{G}_{0,\ell}$ the first ℓ queries with index $j \geq i$ are answered by sampling a secret key uniformly at random from $\mathcal{K}$. Notice that $\mathsf{G}_0 = \mathsf{G}_{0,0}$ and $\mathsf{G}_1 = \mathsf{G}_{0,q}$. Now we show that games $\mathsf{G}_{0,\ell-1}$ and $\mathsf{G}_{0,\ell}$ are indistinguishable. Assume that ℓ-th query to KEYGEN-oracle is for index $k \geq i$.

Let $\tilde{T}_{\mathtt{TRE}}(\lambda)$ be the polynomial whose existence is guaranteed by the security of sTRE. Let $\mathsf{poly}_{\mathtt{FE}}(\lambda)$ be the fixed polynomial which bounds the time required to run FE.Gen, FE.Enc and answer the queries of $\mathcal{A}_{i,\lambda}$ as described in Step 2 of the adversary $\mathcal{B}_{k,\lambda}$ defined below. Set $\underline{T} := (\mathsf{poly}_{\mathtt{FE}}(\lambda))^{1/\epsilon}$. Set $\tilde{T}_{\mathtt{FTRE}} := \max(\tilde{T}_{\mathtt{TRE}}, \underline{T})$.

Claim 4.4.1. *For any n which is polynomial in λ, any set of polynomials $(T_j(\cdot))_{j\in[n]}$ fulfilling that $\forall j \in [n]$ holds $T_j(\cdot) \geq \tilde{T}_{\mathtt{FTRE}}(\cdot)$, for any $i \in [n]$, from any polynomial-size adversary $\mathcal{A}_i = \{\mathcal{A}_{i,\lambda}\}_{\lambda\in\mathbb{N}}$, where the depth of $\mathcal{A}_{i,\lambda}$ is bounded from above by $T_i^{\epsilon}(\lambda)$, we can construct a polynomial-size adversary $\mathcal{B}_k = \{\mathcal{B}_{k,\lambda}\}_{\lambda\in\mathbb{N}}$ whose depth is bounded from above by $T_k^{\epsilon}(\lambda)$ such that*

$$\mathbf{Adv}^{\mathtt{TRE}}_{\mathcal{B}_k} = |\Pr[\mathsf{G}_{0,\ell-1} = 1] - \Pr[\mathsf{G}_{0,\ell} = 1]|\,.$$

The adversary $\mathcal{B}_{k,\lambda}((\mathsf{pp}_{e,j}, \mathsf{pp}_{d,j})_{j\in[n]})$:

1. Runs $(\mathsf{pk}, \mathsf{msk}) \leftarrow \mathtt{FE.Gen}(1^\lambda, \mathcal{F})$ and sets $d := 0$
2. Runs $(x_0, x_1) \leftarrow \mathcal{A}_{i,\lambda}(\mathsf{pk}, (\mathsf{pp}_{e,j}, \mathsf{pp}_{d,j})_{j\in[n]})$ and it answers KEYGEN-queries (f, h) as follows. At first if $i \leq h \leq n$ sets $d := d + 1$. Then
   - If $f \notin \mathcal{F}_\lambda \vee h \notin [n]$ returns $\perp$.
   - If $(h \in [i-1]) \vee (i \leq h \leq n \wedge d < \ell)$, it runs $\mathsf{sk}_f \leftarrow \mathtt{FE.KeyGen}(\mathsf{msk}, f)$, $c_h \leftarrow \mathtt{TRE.Enc}(\mathsf{pp}_{e,h}, \mathsf{sk}_f)$ and returns c_h.
   - If $i \geq h \geq n \wedge d = \ell$, it computes $\mathsf{sk}_0 \leftarrow \mathtt{FE.KeyGen}(\mathsf{msk}, f)$ and samples randomly $\mathsf{sk}_1 \xleftarrow{\$} \mathcal{K}$ such that $|\mathsf{sk}_0| = |\mathsf{sk}_1|$, sends $(\mathsf{sk}_0, \mathsf{sk}_1)$ to its challenger and receives c as answers. It returns c.
   - Otherwise, computes $\mathsf{sk}_f \xleftarrow{\$} \mathcal{K}, c_h \leftarrow \mathtt{TRE.Enc}(\mathsf{pp}_{e,h}, \mathsf{sk}_f)$ and returns c_h.
3. Picks randomly $b \xleftarrow{\$} \{0, 1\}$ and computes $c \leftarrow \mathtt{FE.Enc}(\mathsf{pk}, x_b)$.
4. Runs $b' \leftarrow \mathcal{A}_{i,\lambda}(c)$, answers KEYGEN-queries as before and outputs the truth value of $b = b'$.

If c is the encryption of sk_0, then $\mathcal{B}$ simulates $\mathsf{G}_{0,\ell-1}$ perfectly, otherwise it simulates $\mathsf{G}_{0,\ell}$ perfectly. Therefore

$$\mathbf{Adv}^{\mathtt{TRE}}_{\mathcal{B}_k} = \left|\Pr[\mathsf{G}_{0,\ell-1} = 1] - \Pr[\mathsf{G}_{0,\ell} = 1]\right|,$$

as claimed.

Furthermore, $\mathcal{B}$ meets the depth constraint:

$$\begin{aligned}\mathsf{depth}(\mathcal{B}_{k,\lambda}) &= \mathsf{poly}_{\mathtt{FE}}(\lambda) + \mathsf{depth}(\mathcal{A}_{i,\lambda}) = \underline{T}^{\epsilon}(\lambda) + T^{\epsilon}_{i}(\lambda) \leq \underline{T}^{\epsilon}(\lambda) + T^{\epsilon}_{k}(\lambda) \leq 2T^{\epsilon}_{k}(\lambda) \\ &= o(T^{\epsilon}_{k}(\lambda)).\end{aligned}$$

Also $T_i(\cdot) \geq \tilde{T}_{\mathtt{FTRE}}(\cdot) \geq \tilde{T}_{\mathtt{TRE}}(\cdot)$ as required. This proves the claim.

$\mathcal{A}_i$ makes at most q queries for indices k, where $i \leq k \leq n$ and q is some polynomial in security parameter. Because $\mathbf{Adv}^{\mathtt{TRE}}_{\mathcal{B}_k} \leq \mathsf{negl}'(\lambda)$ for some negligible function $\mathsf{negl}'(\cdot)$, and the sum of q negligible functions is still negligible function for q which is polynomial in the security parameter. Hence, we can conclude

$$|\Pr[\mathsf{G}_0 = 1] - \Pr[\mathsf{G}_1 = 1]| \leq \mathsf{negl}(\lambda),$$

where $\mathsf{negl}(\cdot)$ is some negligible function.

Now we construct an adversary against the security of the FE scheme.

Lemma 4.4.6. *For any n which is polynomial in λ, for any set of polynomials $(T_j(\cdot))_{j\in[n]}$ fulfilling that $\forall j \in [n]$ holds $T_j(\cdot) \geq \tilde{T}_{\mathtt{TRE}}(\cdot)$, for any $i \in [n]$, from any polynomial-size adversary $\mathcal{A}_i = \{\mathcal{A}_{i,\lambda}\}_{\lambda\in\mathbb{N}}$ we can construct a polynomial-size adversary $\mathcal{B}' = \{\mathcal{B}'_\lambda\}_{\lambda\in\mathbb{N}}$ such that*

$$\mathbf{Adv}^{\mathtt{FE}}_{\mathcal{B}'} = \left|\Pr[\mathsf{G}_1 = 1] - \frac{1}{2}\right|.$$

The adversary $\mathcal{B}'_\lambda(\mathsf{pk})$:

1. Runs $(\mathsf{pp}_{e,j}, \mathsf{pp}_{d,j})_{j\in[n]} \leftarrow \mathtt{TRE}.\mathsf{Setup}(1^\lambda, (T_j)_{j\in[n]})$.
2. Runs $(x_0, x_1) \leftarrow \mathcal{A}_{i,\lambda}(\mathsf{pk}, (\mathsf{pp}_{e,j}, \mathsf{pp}_{d,j})_{j\in[n]})$ and answers KEYGEN-queries (f, h) as follows:
   - If $f \notin \mathcal{F}_\lambda \vee h \notin [n]$, it returns $\perp$.
   - If $h < i$ forwards f to KEYGEN-oracle of $\mathcal{B}'_\lambda$, encrypts the answer sk_f under $\mathsf{pp}_{e,h}$ computing $c_h \leftarrow \mathtt{TRE}.\mathsf{Enc}(\mathsf{pp}_{e,h}, \mathsf{sk}_f)$ and returns c_h.
   - Otherwise, samples randomly $\mathsf{sk}_f \xleftarrow{\$} \mathcal{K}$ of appropriate length, computes $c_h \leftarrow \mathtt{TRE}.\mathsf{Enc}(\mathsf{pp}_{e,h}, \mathsf{sk}_f)$ and returns c_h.
3. Forwards (x_0, x_1) to its challenger and receives c.

4. Runs $b' \leftarrow \mathcal{A}_{i,\lambda}(c)$, answers KEYGEN-queries as before and outputs b'.

$\mathcal{B}$ simulates the TRFE experiment perfectly, which yields Lemma 4.4.6. Finally, combining Lemmas 4.4.5 and 4.4.6 we obtain

$$\begin{aligned}\mathbf{Adv}^{\mathtt{FTRE}}_{\mathcal{A}_i} &= \left|\Pr[\mathsf{G}_0 = 1] - \frac{1}{2}\right| = \left|\Pr[\mathsf{G}_0 = 1] - \Pr[\mathsf{G}_1 = 1]\right| + \left|\Pr[\mathsf{G}_1 = 1] - \frac{1}{2}\right| \\ &\leq \mathsf{negl}(\lambda) + \mathbf{Adv}^{\mathtt{FE}}_{\mathcal{B}'},\end{aligned}$$

which concludes the proof. □

Application to locked-key IBE.

Since identity-based encryption is a special case of functional encryption, as an interesting application of TRFE we can time-lock secret keys of an IBE scheme. In more detail, the central authority in an IBE scheme can attach hardness parameters to generated keys. As a consequence, those keys become usable only sequentially. This, for example, enables an IBE key authority to produce all secret keys in the beginning and afterwards go off-line.

4.4.5 Applications

Subsequently, we discuss the applications in [MT19] when we use our (homomorphic) TRE approach in contrast to HTLPs of MT19. All the following applications have in common that they require decrypting a set of encrypted messages at some given time. Our approach to TRE allows decrypting an arbitrary number of messages at a specified time by solving one puzzle. In [MT19] this is achieved by a homomorphic evaluation of puzzles and then solving one or more resulting puzzles. The drawback of this solution is that one needs to wait until all puzzles of interest have been collected, then execute a homomorphic evaluation, and only after that, the resulting puzzles can be solved. Our scheme allows starting to solve the puzzle immediately after Setup is run. In all of these applications we are able to use our TRE approach *without* any homomorphic property.

E-voting. We focus on designing an e-voting protocol in absence of a trusted party, where voters can cast their preference without any bias. Similarly to [MT19], we neither consider privacy nor authenticity of the votes. The crucial property of our $\mathtt{TRE}$ is that the setup can be reused for producing an arbitrary number of ciphertexts and for that reason it is enough to run Solve only once. The output s of Solve allows obtaining the secret key which is then used to decrypt all ciphertexts that have been produced using the corresponding pp_e. Therefore, if we encrypt all votes using the same pp_e, we are able to decrypt all ciphertexts at the same time. Then it is easy to obtain the final result by combining decrypted plaintexts. We

assume a scenario with n voters and m candidates. All voters have access to a public append-only bulletin board. The e-voting protocol consists of an *election setup*, a *voting phase* and a *counting phase*. The election setup outputs public parameters which can be used during the voting phase and the counting phase. During the voting phase, every voter V_i outputs a vote for a candidate of its preference. In the counting phase, the votes are counted and the candidate with a maximum number of votes is announced. In order to avoid any bias, the votes have to be kept secret during the voting phase. Our e-voting protocol is given in Figure 4.8.

Election Setup
Run $(\mathsf{pp}_e, \mathsf{pp}_d) \leftarrow \mathtt{TRE}.\mathsf{Setup}(1^\lambda, T)$ and publish public parameters to be accessible by all the voters.
Run $s \leftarrow \mathtt{TRE}.\mathsf{Solve}(\mathsf{pp}_d)$.

Voting Phase
Every voter V_i simply encrypts its preferred candidate C_j.
Compute $v_i \leftarrow \mathtt{TRE}.\mathsf{Enc}(\mathsf{pp}_e, C_j)$ and output v_i.

Counting Phase
For all v_i run $C_j \leftarrow \mathtt{TRE}.\mathsf{Dec}(s, v_i)$.
Output the candidate C_j with the maximum number of votes.

Figure 4.8: E-voting protocol

Notice that the security of the TRE scheme guarantees that all votes remain hidden during the whole voting phase. In the e-voting protocol proposed in [MT19], we have to wait until the voting phase is finished and then we can combine puzzles from the voting phase to m resulting puzzles (one per candidate where votes are encoded as 0 and 1 respectively). Then, these m puzzles can be solved, which requires at least time T and solving m puzzles in parallel. Hence, it requires time T after the voting phase is over to be able to announce the results. This is in contrast to what we can do with our TRE, in which we can encrypt the respective encoding of the candidate $i \in [m]$ directly, and can start to solve a *single* puzzle immediately after Setup is run and hence the results are available at the beginning of the counting phase.

Multi-Party Coin Flipping. In multi-party coin flipping, we assume n parties which want to flip a coin in the following way: 1) The value of the coin is unbiased even if $n-1$ parties collude and 2) all parties agree on the same value for the coin.

The approach proposed in [MT19] relies on HTLPs and their protocol consists of

three phases: Setup, Coin Flipping, and Announcement of the result. Similarly to the e-voting protocol, one is only able to start solving the puzzle in the last phase and hence obtains the results after time T. We are able to avoid this problem, by using our TRE approach, where we can start to solve the puzzle already after the setup phase.

Sealed-Bid Auctions. Here we consider an auction with n bidders. The protocol consists of two phases: a bidding phase and an opening phase. Bids should be kept secret during the bidding phase and later revealed in the opening phase. Time-lock puzzles are used in this scenario to mitigate the issue that some bidders can go offline after the bidding phase. If we only use standard time-lock puzzles, then the number of puzzles that have to be solved in the opening phase is equal to the number of bidders who went offline. In [MT19] this problem was resolved by using HLTPs. Again, this solution has the same issues as the ones discussed above and can be avoided using our TRE approach.

Multi-Party Contract Signing. In multi-party contract signing, we assume n parties which want to jointly sign a contract. The parties are mutually distrusting and the contract is valid only if it is signed by all parties. The protocol in [MT19] consists of four phases: setup, key generation, signing, and aggregation, and combines aggregate signatures from RSA with multiplicatively homomorphic time-lock puzzles with a setup that allows producing puzzles for different hardness parameters. We remark that this type of time-lock puzzles are in some sense equivalent to our sequential timed-release encryption.[1] The protocol runs in ℓ rounds and in the i-th round every party should create a puzzle with hardness $T_{\ell-i+1}$ which contains a signature of the required message. Hence, the hardness of the puzzles decreases in every round. If some parties have not broadcasted their puzzles in any round, the parties will homomorphically evaluate puzzles from the previous round and solve the resulting puzzle.

Consider a scenario, where in the i-th round some party does not broadcast its puzzle. Then if we do not take the time for the homomorphic evaluation into account, we need time $T_{\ell-i+1}$ to solve the resulting puzzle after this event happened. On the other hand, if we use sequential TRE, we can obtain the result in time $T_{\ell-i+1}$ after the setup was executed. Moreover, we can combine sequential TRE with an arbitrary aggregate signature scheme, because we do not need to perform any homomorphic evaluation.

Remark 4.4.1. Note that in all of these applications we could use Offline Time-Lock Encryption instead of Timed-Release Encryption. Since OTLE relies on extractability obfuscation, which is not practical or even feasible yet, we discussed

[1] Though they only discuss them informally in [MT19] and as mentioned in Section 4.3 it seems that it is not possible to prove it secure as it is proposed.

all applications using TRE. This also enables better comparability with HTLPs.

4.5 Timed Public-Key Encryption

Timed Public-Key Encryption (TPKE) has been proposed by Katz *et al.* [KLX20] concurrently with our notion of Timed-Release Encryption. The main conceptual difference between the two is that TPKE has an additional decryption algorithm that allows for fast decryption using a secret key. Moreover, the security definition of TPKE requires that the time which is needed for decryption without the secret key starts to run from the point of creating a ciphertext and the scheme should be IND-CCA secure. TPKE has been used as a building block for non-malleable timed-commitments, which explains the need for IND-CCA security. There is currently only one known construction of TPKE scheme which has some inefficiency issues. Concretely, the time required to encrypt a single message is proportional to the hardness parameter T. We solve this issue in two ways. One can observe that if we allow that time starts to run from the point of running the setup algorithm, then our TRE scheme can be adjusted to TPKE. In case that such a security adjustment is not desirable, we propose two constructions of TPKE which are more efficient than the one described by Katz *et al.* [KLX20].

At first we recall the definition of a timed public-key encryption scheme as stated in [KLX20].

Definition 4.5.1. A timed public-key encryption scheme $\texttt{TPKE}$ with message space $\mathcal{M}$ is tuple of algorithms $\texttt{TPKE} = (\mathsf{KGen}, \mathsf{Enc}, \mathsf{Dec}_f, \mathsf{Dec}_s)$ with following syntax.

- $(\mathsf{pk}, \mathsf{sk}) \leftarrow \mathsf{KGen}(1^\lambda, T)$ is a probabilistic algorithm that takes as input the security parameter 1^λ and a hardness parameter T and outputs a public/secret key pair.
- $c \leftarrow \mathsf{Enc}(\mathsf{pk}, m)$ is a probabilistic algorithm that takes as input a public key pk and a message m and outputs a ciphertext c in time at most t_e.
- $m \leftarrow \mathsf{Dec}_f(\mathsf{sk}, c)$ is a deterministic fast decryption algorithm that takes as input a secret key sk and a ciphertext c and outputs $m \in \mathcal{M} \cup \{\perp\}$ in time at most t_{fd}.
- $m \leftarrow \mathsf{Dec}_s(\mathsf{pk}, c)$ is a deterministic slow decryption algorithm that takes as input a public key pk and a ciphertext c and outputs $m \in \mathcal{M} \cup \{\perp\}$ in time at most t_{sd}.

We say $\texttt{TPKE}$ is correct if for all $\lambda, T \in \mathbb{N}$ and all $m \in \mathcal{M}$ holds:

$$\Pr\left[\mathsf{Dec}_f(\mathsf{sk}, c) = \mathsf{Dec}_s(\mathsf{pk}, c) = m : \begin{matrix}(\mathsf{pk}, \mathsf{sk}) \leftarrow \mathsf{KGen}(1^\lambda, T) \\ c \leftarrow \mathsf{Enc}(\mathsf{pk}, m)\end{matrix}\right] = 1.$$

As the names of the decryption algorithms in Definition 4.5.1 suggest, it should hold that t_{fd} is much smaller than t_{sd}. Now we state the original security definition, however, adjusted to the computational model which is used in this thesis.

Definition 4.5.2. A TPKE scheme $\texttt{TPKE}$ is *IND-CCA secure* with gap $0 < \epsilon < 1$ if there exists a polynomial $\tilde{T}(\cdot)$ such that for all polynomials $T(\cdot) \geq \tilde{T}(\cdot)$ and every polynomial-size adversary $\mathcal{A} = \{(\mathcal{A}_{1,\lambda}, \mathcal{A}_{2,\lambda})\}_{\lambda\in\mathbb{N}}$, where the depth of $\mathcal{A}_{2,\lambda}$ is at most $T^{\epsilon}(\lambda)$, there exists a negligible function $\mathsf{negl}(\cdot)$ such that for all $\lambda \in \mathbb{N}$ it holds

$$\mathbf{Adv}^{\texttt{TPKE}}_{\mathcal{A}} = \left| \Pr \left[b = b' \ : \begin{array}{r} (\mathsf{pk}, \mathsf{sk}) \leftarrow \mathsf{KGen}(1^{\lambda}, T(\lambda)) \\ (m_0, m_1, \mathsf{st}) \leftarrow \mathcal{A}^{\mathsf{DEC}(\cdot)}_{1,\lambda}(\mathsf{pk}) \\ b \xleftarrow{\$} \{0,1\};\ c^* \leftarrow \mathsf{Enc}(\mathsf{pk}, m_b) \\ b' \leftarrow \mathcal{A}^{\mathsf{DEC}(\cdot)}_{2,\lambda}(c^*, \mathsf{st}) \end{array} \right] - \frac{1}{2} \right| \leq \mathsf{negl}(\lambda),$$

where the oracle $\mathsf{DEC}(c)$ returns $\mathsf{Dec}_f(\mathsf{sk}, c)$ with the restriction that $\mathcal{A}_{2,\lambda}$ is not allowed to query the oracle $\mathsf{DEC}(\cdot)$ for decryption of the challenge ciphertext c^*. We require that $|m_0| = |m_1|$.

We also assume relaxation of the above security definition and let the time required to decrypt messages using the slow decryption starts with the running of KGen. Because this security definition is weaker we denote it as *weak* IND-CCA security.

Definition 4.5.3. A TPKE scheme $\texttt{TPKE}$ is *weak IND-CCA secure* with gap $0 < \epsilon < 1$ if there exists a polynomial $\tilde{T}(\cdot)$ such that for all polynomials $T(\cdot) \geq \tilde{T}(\cdot)$ and every polynomial-size adversary $\mathcal{A} = \{\mathcal{A}_{\lambda}\}_{\lambda\in\mathbb{N}}$, where the the depth of $\mathcal{A}_{\lambda}$ is at most $T^{\epsilon}(\lambda)$, there exists a negligible function $\mathsf{negl}(\cdot)$ such that for all $\lambda \in \mathbb{N}$ it holds

$$\mathbf{Adv}^{\texttt{TPKE}}_{\mathcal{A}} = \left| \Pr \left[b = b' \ : \begin{array}{r} (\mathsf{pk}, \mathsf{sk}) \leftarrow \mathsf{KGen}(1^{\lambda}, T(\lambda)) \\ (m_0, m_1) \leftarrow \mathcal{A}^{\mathsf{DEC}(\cdot)}_{\lambda}(\mathsf{pk}) \\ b \xleftarrow{\$} \{0,1\};\ c^* \leftarrow \mathsf{Enc}(\mathsf{pk}, m_b) \\ b' \leftarrow \mathcal{A}^{\mathsf{DEC}(\cdot)}_{\lambda}(c^*) \end{array} \right] - \frac{1}{2} \right| \leq \mathsf{negl}(\lambda),$$

where the oracle $\mathsf{DEC}(c)$ returns $\mathsf{Dec}_f(\mathsf{sk}, c)$ with the restriction that after giving adversary $\mathcal{A}_{\lambda}$ the challenge ciphertext c^*, $\mathcal{A}_{\lambda}$ is not allowed to query the oracle $\mathsf{DEC}(\cdot)$ for the challenge ciphertext c^*. We require that $|m_0| = |m_1|$.

$\underline{\mathsf{KGen}(1^\lambda, T)}$	$\underline{\mathsf{Dec}_s(\mathsf{pk}, c)}$
$(Z, s) \leftarrow \mathtt{TLP}.\mathsf{Gen}(1^\lambda, T)$	Parse pk as (pk', Z)
$(\mathsf{pk}', \mathsf{sk}') \leftarrow \mathtt{PKE}.\mathsf{Gen}(1^\lambda; s)$	$s \leftarrow \mathtt{TLP}.\mathsf{Solve}(Z)$
return $\mathsf{pk} := (\mathsf{pk}', Z), \mathsf{sk} := \mathsf{sk}'$	$(\mathsf{pk}', \mathsf{sk}') \leftarrow \mathtt{PKE}.\mathsf{Gen}(1^\lambda; s)$
	return $m \leftarrow \mathtt{PKE}.\mathsf{Dec}(\mathsf{sk}', c)$
$\underline{\mathsf{Enc}(\mathsf{pk}, m)}$	$\underline{\mathsf{Dec}_f(\mathsf{sk}, c)}$
Parse pk as (pk', Z)	return $m \leftarrow \mathtt{PKE}.\mathsf{Dec}(\mathsf{sk}', c)$
return $c \leftarrow \mathtt{PKE}.\mathsf{Enc}(\mathsf{pk}', m)$	

Figure 4.9: Construction of weak IND-CCA secure TPKE

4.5.1 Weak IND-CCA-secure TPKE

To obtain TPKE from our TRE construction we simply let Setup of TRE output a secret key, which can be then used for fast decryption.

Theorem 4.5.1. *Let* $\mathtt{TLP} = (\mathtt{TLP}.\mathsf{Gen}, \mathtt{TLP}.\mathsf{Solve})$ *be a secure time-lock puzzle with gap* ϵ *and* $\mathtt{PKE} = (\mathtt{PKE}.\mathsf{Gen}, \mathtt{PKE}.\mathsf{Enc}, \mathtt{PKE}.\mathsf{Dec})$ *be an IND-CCA-secure public-key encryption scheme, then* $\mathtt{TPKE} = (\mathsf{KGen}, \mathsf{Enc}, \mathsf{Dec}_f, \mathsf{Dec}_s)$ *defined in Figure 4.9 is a weak IND-CCA-secure timed public-key encryption scheme with gap* $\underline{\epsilon}$ *for any* $\underline{\epsilon} < \epsilon$.

Proof. The proof of this theorem is essentially the same as the proof of Theorem 4.4.1. We only have to argue that in each game we are able to answer decryption queries. In the game G_1 this can be done using sk of the underlying PKE scheme. In the game G_2, where we build a reduction against CCA security of the underlying PKE scheme, we can use decryption oracle $\mathsf{DEC}(\cdot)$ to answer decryption queries. □

We note that the constructed TPKE can be used in the construction of non-malleable timed commitment via the framework of Katz *et al.* [KLX20], however, the security definition of non-malleable timed commitment must be adjusted to allow the time required to perform forced open of a commitment starts with the generation of the parameters. Notice that this would lead to the "solve one, get many for free" property in a sense that by solving one puzzle many commitments can be opened at once.

4.5.2 Generic Construction of IND-CCA-secure TPKE

Now we build a TPKE scheme achieving the original IND-CCA security notation generically from any timed-release encryption. For security, we require a slightly

stronger security notion than the one assumed in Definition 4.4.2. We allow an adversary to execute some preprocessing at the beginning of an experiment and we define security using two experiments. In the first experiment, an adversary is given an encryption of the message of its choice and in the second experiment, the adversary is given an encryption of the random message. We require that even with pre-processing the adversary is not able to distinguish between these two experiments unless the adversary has sufficient running time to decrypt messages. Defining security using two experiments allows us for a simpler security proof of TPKE.

$\underline{\mathsf{ExpTRE\text{-}PP}^b_{\mathcal{A}}(\lambda)\text{:}}$
$\mathsf{st} \leftarrow \mathcal{A}_{1,\lambda}$
$(\mathsf{pp}_e, \mathsf{pp}_d) \leftarrow \mathsf{Setup}(1^\lambda, T(\lambda))$
$m_0 \leftarrow \mathcal{A}_{2,\lambda}(\mathsf{pp}_e, \mathsf{pp}_d, \mathsf{st})$
$m_1 \xleftarrow{\$} \mathcal{M}$ s.t. $|m_0| = |m_1|$
$c \leftarrow \mathsf{Enc}(\mathsf{pp}_e, m_b)$
return $b' \leftarrow \mathcal{A}_{2,\lambda}(c)$

Figure 4.10: Security experiment for TRE secure against pre-processing.

Definition 4.5.4. Consider the security experiment $\mathsf{ExpTLP\text{-}PP}^b_{\mathcal{A}}(\lambda)$ depicted in Figure 4.10. A timed-release encryption is *secure against pre-processing with gap* $0 < \epsilon < 1$ if there exists a polynomial $\tilde{T}(\cdot)$ such that for all polynomials $T(\cdot) \geq \tilde{T}(\cdot)$ and every polynomial-size adversary $\mathcal{A} = \{(\mathcal{A}_{1,\lambda}, \mathcal{A}_{2,\lambda})\}_{\lambda\in\mathbb{N}}$, where the depth of $\mathcal{A}_{2,\lambda}$ is at most $T^\epsilon(\lambda)$, there exists a negligible function $\mathsf{negl}(\cdot)$ such that for all $\lambda \in \mathbb{N}$ it holds

$$\mathbf{Adv}^{\texttt{TRE-PP}}_{\mathcal{A}} = \left|\Pr\left[\mathsf{ExpTRE\text{-}PP}^0_{\mathcal{A}}(\lambda) = 1\right] - \Pr\left[\mathsf{ExpTRE\text{-}PP}^1_{\mathcal{A}}(\lambda) = 1\right]\right| \leq \mathsf{negl}(\lambda).$$

We can build TRE secure against pre-processing from a TLP and PKE similarly as in our basic construction of TRE in Figure 4.3, however, the TLP has to be also secure against pre-processing. Therefore we define a stronger security notion for TLPs, where an adversary can execute some pre-processing at the beginning of the game, and even with the pre-processing the solution of the puzzle should be indistinguishable from random unless the adversary has sufficient running time to solve the puzzle. Notice that if we instantiate a TLP with sequential squaring, then this security notion is directly implied by the strong sequential squaring assumption (Definition 2.5.4).

Definition 4.5.5. Consider the security experiment $\mathsf{ExpTLP\text{-}PP}^b_{\mathcal{A}}(\lambda)$ depicted in Figure 4.11. We say that a time-lock puzzle $\texttt{TLP}$ is secure against pre-processing

$\underline{\mathsf{ExpTLP\text{-}PP}^b_{\mathcal{A}}(\lambda)\text{:}}$
$\mathsf{st} \leftarrow \mathcal{A}_{1,\lambda}, b \xleftarrow{\$} \{0,1\}$
$(Z, s_0) \leftarrow \mathsf{Gen}(1^\lambda, T(\lambda))$
$s_1 \xleftarrow{\$} S$
return $b' \leftarrow \mathcal{A}_{2,\lambda}(Z, s_b, \mathsf{st})$

Figure 4.11: Security experiment for TLP secure against pre-processing.

with gap $0 < \epsilon < 1$, if there exists a polynomial $\tilde{T}(\cdot)$ such that for all polynomials $T(\cdot) \geq \tilde{T}(\cdot)$ and every polynomial-size adversary $\mathcal{A} = \{(\mathcal{A}_{1,\lambda}, \mathcal{A}_{2,\lambda})\}_{\lambda\in\mathbb{N}}$, where the depth of $\mathcal{A}_{2,\lambda}$ is at most $T^\epsilon(\lambda)$, there exists a negligible function $\mathsf{negl}(\cdot)$ such that for all $\lambda \in \mathbb{N}$ it holds

$$\mathbf{Adv}^{\text{TLP-PP}}_{\mathcal{A}} = \left|\Pr\left[\mathsf{ExpTLP\text{-}PP}^0_{\mathcal{A}}(\lambda) = 1\right] - \Pr\left[\mathsf{ExpTLP\text{-}PP}^1_{\mathcal{A}}(\lambda) = 1\right]\right| \leq \mathsf{negl}(\lambda).$$

Now we show that the construction in Figure 4.3 yields a TRE scheme secure against pre-processing if the TLP is also secure against preprocessing. In the proof we uses the RoR-CPA security definition for PKE.

Theorem 4.5.2. *If* $\mathtt{TLP} = (\mathtt{TLP.Gen}, \mathtt{TLP.Solve})$ *is a secure time-lock puzzle against pre-processing with gap* ϵ *and* $\mathtt{PKE} = (\mathtt{PKE.Gen}, \mathtt{PKE.Enc}, \mathtt{PKE.Dec})$ *is an RoR-CPA-secure public-key encryption scheme, then for any* $\underline{\epsilon} < \epsilon$, $\mathtt{TRE} = (\mathsf{Setup}, \mathsf{Solve}, \mathsf{Enc}, \mathsf{Dec})$ *defined in Figure 4.3 is a secure timed-release encryption scheme against pre-processing with gap* $\underline{\epsilon}$.

Proof. This proof is similar to the security proof of our basic TRE construction. Correctness of the scheme is directly implied by the correctness of the PKE scheme and the TLP.

To prove security, we define a sequence of games $\mathsf{G}_0 - \mathsf{G}_3$. For $i \in \{0,1,2,3\}$ we denote by $\mathsf{G}_i = 1$ the event that the adversary $\mathcal{A} = \{(\mathcal{A}_{1,\lambda}, \mathcal{A}_{2,\lambda})\}_{\lambda\in\mathbb{N}}$ outputs $b' = 1$ in game G_i.

Game 0. Game G_0 corresponds to the experiment $\mathsf{ExpTLP\text{-}PP}^0_{\mathcal{A}}$, where $\mathcal{A}$ is given encryption of the message m_0.

Game 1. In G_1 we sample s uniformly at random from S and use it as randomness for Gen of PKE.

Let $\tilde{T}_{\mathtt{TLP}}(\lambda)$ be the polynomial whose existence is guaranteed by the security of TLP. Let $\mathsf{poly}_{\mathtt{PKE}}(\lambda)$ be the fixed polynomial which bounds the time required to run $\mathtt{PKE.Gen}$ and $\mathtt{PKE.Enc}$. Set $\underline{T} := (\mathsf{poly}_{\mathtt{PKE}}(\lambda))^{1/\underline{\epsilon}}$. Set $\tilde{T}_{\mathtt{TRE}} := \max(\tilde{T}_{\mathtt{TLP}}, \underline{T})$.

Lemma 4.5.1. *From any polynomial-size adversary $\mathcal{A} = \{(\mathcal{A}_{1,\lambda}, \mathcal{A}_{2,\lambda})\}_{\lambda\in\mathbb{N}}$, where the depth of $\mathcal{A}_{2,\lambda}$ is bounded from above by $T^{\underline{\epsilon}}(\lambda)$ for some $T(\cdot) \geq \tilde{T}_{\mathtt{TRE}}(\cdot)$ we can construct a polynomial-size adversary $\mathcal{B} = \{(\mathcal{B}_{1,\lambda}, \mathcal{B}_{2,\lambda})\}_{\lambda\in\mathbb{N}}$ where the depth of $\mathcal{B}_{2,\lambda}$ is at most $T^{\epsilon}(\lambda)$ with*

$$\mathbf{Adv}^{\text{TLP-PP}}_{\mathcal{B}} = |\Pr[\mathsf{G}_0 = 1] - \Pr[\mathsf{G}_1 = 1]|\,.$$

We construct an adversary $\mathcal{B} = \{(\mathcal{B}_{1,\lambda}, \mathcal{B}_{2,\lambda})\}_{\lambda\in\mathbb{N}}$ as follows.
The adversary $\mathcal{B}_{1,\lambda}$:

1. Runs $\mathsf{st} \leftarrow \mathcal{A}_{1,\lambda}$.
2. Outputs the state st.

The adversary $\mathcal{B}_{2,\lambda}(Z, s_b, \mathsf{st})$:

1. Runs $(\mathsf{pk}, \mathsf{sk}) \leftarrow \mathtt{PKE}.\mathsf{Gen}(1^\lambda; s_b)$.
2. Runs $m_0 \leftarrow \mathcal{A}_{2,\lambda}((\mathsf{pk}, Z), \mathsf{st})$.
3. Computes $c \leftarrow \mathtt{PKE}.\mathsf{Enc}(\mathsf{pk}, m_0)$.
4. Finally, runs $b' \leftarrow \mathcal{A}_{2,\lambda}(c)$ and returns b'.

If the challenger picks $b = 0$, then s_b is the real solution of Z and $\mathcal{B}$ perfectly simulates G_0, otherwise s_b is random value and $\mathcal{B}$ perfectly simulates G_1. Hence, we obtain

$$\mathbf{Adv}^{\text{TLP-PP}}_{\mathcal{B}} = |\Pr[\mathsf{G}_0 = 1] - \Pr[\mathsf{G}_1 = 1]|\,.$$

Moreover, $\mathcal{B}_2$ fulfils the depth constraint:

$$\mathsf{depth}(\mathcal{B}_{2,\lambda}) = \mathsf{poly}_{\mathtt{PKE}}(\lambda) + \mathsf{depth}(\mathcal{A}_{2,\lambda}) \leq \underline{T^{\epsilon}}(\lambda) + T^{\underline{\epsilon}}(\lambda) \leq 2T^{\underline{\epsilon}}(\lambda) = o(T^{\epsilon}(\lambda)).$$

Also $T(\cdot) \geq \tilde{T}_{\mathtt{TRE}}(\cdot) \geq \tilde{T}_{\mathtt{TLP}}(\cdot)$ as required.

Game 2. In G_2 instead of encrypting the message m_0, we compute c as an encryption of a random message.

Lemma 4.5.2. *From any polynomial-size adversary $\mathcal{A} = \{(\mathcal{A}_{1,\lambda}, \mathcal{A}_{2,\lambda})\}_{\lambda\in\mathbb{N}}$ we can construct a polynomial size adversary $\mathcal{B}' = \{\mathcal{B}'_\lambda\}_{\lambda\in\mathbb{N}}$ such that*

$$|\Pr[\mathsf{G}_1 = 1] - \Pr[\mathsf{G}_2 = 1]| = \mathbf{Adv}^{\mathtt{PKE},\mathsf{RoR}}_{\mathcal{B}'}.$$

Notice that in G_2 we use fresh randomness which is independent of the puzzle Z. We construct $\mathcal{B}'$ as follows.
The adversary $\mathcal{B}'_\lambda(\mathsf{pk})$:

1. Runs $\mathsf{st} \leftarrow \mathcal{A}_{1,\lambda}$.
2. Runs $(Z, s) \leftarrow \mathtt{TLP}.\mathsf{Gen}(1^\lambda, T)$.
3. Runs $m_0 \leftarrow \mathcal{A}_{2,\lambda}((\mathsf{pk}, Z), \mathsf{st})$.
4. Outputs m_0 to its experiment and receives c.
5. Finally returns $b' \leftarrow \mathcal{A}_{2,\lambda}(c)$.

If the challenger picks $b = 0$, then c is an encryption of the value m_0 and $\mathcal{B}'$ perfectly simulates G_1, otherwise c is an encryption of a random message and $\mathcal{B}'$ perfectly simulates G_2. Hence, we obtain

$$\mathbf{Adv}_{\mathcal{B}'}^{\mathtt{PKE},\mathsf{RoR}} = \left|\Pr[\mathsf{G}_1 = 1] - \Pr[\mathsf{G}_2 = 1]\right|.$$

Game 3. In game G_3 we use the real solution of the puzzle as randomness for Gen of PKE.

Let $\tilde{T}_{\mathtt{TLP}}(\lambda)$, $\mathsf{poly}_{\mathtt{PKE}}(\lambda)$, $\underline{T} := (\mathsf{poly}_{\mathtt{PKE}}(\lambda))^{1/\epsilon}$, and $\tilde{T}_{\mathtt{TRE}}$ are defined as before in game G_1.

Lemma 4.5.3. *From any polynomial-size adversary $\mathcal{A} = \{(\mathcal{A}_{1,\lambda}, \mathcal{A}_{2,\lambda})\}_{\lambda\in\mathbb{N}}$, where the depth of $\mathcal{A}_{2,\lambda}$ is bounded from above by $T^{\epsilon}(\lambda)$ for some $T(\cdot) \geq \tilde{T}_{\mathtt{TRE}}(\cdot)$ we can construct a polynomial-size adversary $\mathcal{B} = \{(\mathcal{B}_{1,\lambda}, \mathcal{B}_{2,\lambda})\}_{\lambda\in\mathbb{N}}$ where the depth of $\mathcal{B}_{2,\lambda}$ is at most $T^{\epsilon}(\lambda)$ with*

$$\mathbf{Adv}_{\mathcal{B}}^{\mathtt{TLP\text{-}PP}} = \left|\Pr[\mathsf{G}_2 = 1] - \Pr[\mathsf{G}_3 = 1]\right|.$$

This lemma can be proven in a similar way as Lemma 4.5.1.

Notice that G_3 corresponds to $\mathsf{ExpTLP\text{-}PP}^1_{\mathcal{A}}$. By combining Lemmas 4.5.1-4.5.3 we obtain following:

$$\begin{aligned}
&\mathbf{Adv}_{\mathcal{A}}^{\mathtt{TRE\text{-}PP}} = \left|\Pr[\mathsf{G}_0 = 1] - \Pr[\mathsf{G}_3 = 1]\right| \\
&= \left|\sum_{i=0}^{2} \left(\Pr[\mathsf{G}_i = 1] - \Pr[\mathsf{G}_{i+1} = 1]\right)\right| \leq \sum_{i=0}^{2} \left|\Pr[\mathsf{G}_i = 1] - \Pr[\mathsf{G}_{i+1} = 1]\right| \\
&\leq 2\mathbf{Adv}_{\mathcal{B}}^{\mathtt{TLP\text{-}PP}} + \mathbf{Adv}_{\mathcal{B}'}^{\mathtt{PKE},\mathsf{RoR}},
\end{aligned}$$

which concludes the proof.

□

We can now construct a TPKE scheme as follows. Let $\mathtt{TBE} = (\mathtt{TBE.Gen}, \mathtt{TBE.Enc}, \mathtt{TBE.Dec})$ be a tag-based encryption scheme, $\mathtt{OTS} = (\mathtt{OTS.Gen}, \mathtt{OTS.Sign}, \mathtt{OTS.Vrfy})$ be a signature scheme, and $\mathtt{TRE} = (\mathtt{TRE.Setup}, \mathtt{TRE.Solve}, \mathtt{TRE.Enc}, \mathtt{TRE.Dec})$ be a TRE scheme. Figure 4.12 describes our construction of a TPKE scheme. Notice that we run $\mathtt{TRE.Setup}$ in the Enc algorithm of $\mathtt{TPKE}$. Since the runtime of $\mathtt{TRE.Setup}$ is $\mathsf{poly}(\log T, \lambda)$, the Enc algorithm has also runtime $\mathsf{poly}'(\log T, \lambda)$. We remark that in the TPKE construction of Katz *et al.* [KLX20] the Enc algorithm runs in time $T \cdot \mathsf{poly}''(\lambda)$ and hence our construction provides significant efficiency improvement.

$\underline{\mathsf{KGen}(1^\lambda, T)}$	$\underline{\mathsf{Dec}_f(\mathsf{sk}, c)}$
$(\mathsf{pk}_1, \mathsf{sk}_1) \leftarrow \mathtt{TBE.Gen}(1^\lambda)$	Parse c as $(\mathsf{vk}_{\mathsf{OT}}, \mathsf{pp}_d, c_1, c_2, \sigma)$
$\mathsf{pk} := (\mathsf{pk}_1, T)$, $\mathsf{sk} := \mathsf{sk}_1$	if $\mathtt{OTS.Vrfy}(\mathsf{vk}_{\mathsf{OT}}, (\mathsf{pp}_d, c_1, c_2), \sigma) = 1$
return $(\mathsf{pk}, \mathsf{sk})$	return $m \leftarrow \mathtt{TBE.Dec}(\mathsf{sk}, \mathsf{vk}_{\mathsf{OT}}, c_1)$
	return $\perp$
$\underline{\mathsf{Enc}(\mathsf{pk}, m)}$	$\underline{\mathsf{Dec}_s(\mathsf{pk}, c)}$
$(\mathsf{pp}_e, \mathsf{pp}_d) \leftarrow \mathtt{TRE.Setup}(1^\lambda, T)$	Parse c as $(\mathsf{vk}_{\mathsf{OT}}, \mathsf{pp}_d, c_1, c_2, \sigma)$
$(\mathsf{vk}_{\mathsf{OT}}, \mathsf{sk}_{\mathsf{OT}}) \leftarrow \mathtt{OTS.Gen}(1^\lambda)$	if $\mathtt{OTS.Vrfy}(\mathsf{vk}_{\mathsf{OT}}, (\mathsf{pp}_d, c_1, c_2), \sigma) = 1$
$c_1 \leftarrow \mathtt{TBE.Enc}(\mathsf{pk}_1, \mathsf{vk}_{\mathsf{OT}}, m)$	$s \leftarrow \mathtt{TRE.Solve}(\mathsf{pp}_d)$
$c_2 \leftarrow \mathtt{TRE.Enc}(\mathsf{pp}_e, m)$	return $m \leftarrow \mathtt{TRE.Dec}(s, c_2)$
$\sigma \leftarrow \mathtt{OTS.Sign}(\mathsf{sk}_{\mathsf{OT}}, (\mathsf{pp}_d, c_1, c_2))$	return $\perp$
return $c := (\mathsf{vk}_{\mathsf{OT}}, \mathsf{pp}_d, c_1, c_2, \sigma)$	

Figure 4.12: Generic construction of TPKE.

Theorem 4.5.3. *If* $\mathtt{TBE} = (\mathtt{TBE.Gen}, \mathtt{TBE.Enc}, \mathtt{TBE.Dec})$ *is a tag-based encryption scheme that is selective-tag weakly secure against chosen ciphertext attacks,* $\mathtt{OTS} = (\mathtt{OTS.Gen}, \mathtt{OTS.Sign}, \mathtt{OTS.Vrfy})$ *is a strong one-time signature scheme,* $\mathtt{TRE} = (\mathtt{TRE.Setup}, \mathtt{TRE.Solve}, \mathtt{TRE.Enc}, \mathtt{TRE.Dec})$ *is a secure timed-release encryption scheme against pre-processing with gap* ϵ*, then for any* $\underline{\epsilon} < \epsilon$*,* $(\mathsf{KGen}, \mathsf{Enc}, \mathsf{Dec}_f, \mathsf{Dec}_s)$ *defined in Figure 4.12 is an IND-CCA-secure timed public-key encryption scheme with gap* $\underline{\epsilon}$*.*

Proof. Correctness of the scheme is implied by the correctness of the tag-based encryption scheme, the signature scheme, and the timed-release encryption scheme.

To prove security we define two games $\mathsf{G}_0 - \mathsf{G}_1$. For $i \in \{0, 1\}$ we denote by $\mathsf{G}_i = 1$ the event that the adversary $\mathcal{A} = \{(\mathcal{A}_{1,\lambda}, \mathcal{A}_{2,\lambda})\}_{\lambda \in \mathbb{N}}$ outputs b' in the game G_i such that $b = b'$.

Game 0. Game G_0 corresponds to the original security experiment. Decryption queries are answered using sk_1.

Game 1. In G_1 instead of encrypting the given message twice, we compute c_2 as an encryption of a random message.

Let $\tilde{T}_{\mathtt{TRE}}(\lambda)$ be the polynomial whose existence is guaranteed by the security of TRE. Let $\mathsf{poly}_{\mathcal{B}}(\lambda)$ be the fixed polynomial which bounds the time required to execute Steps 1–3 and answer decryption queries in Step 4 of the adversary $\mathcal{B}_{2,\lambda}$ defined below. Set $\underline{T} := (\mathsf{poly}_{\mathcal{B}}(\lambda))^{1/\underline{\epsilon}}$. Set $\tilde{T}_{\mathtt{TPKE}} := \max(\tilde{T}_{\mathtt{TRE}}, \underline{T})$.

Lemma 4.5.4. *From any polynomial-size adversary $\mathcal{A} = \{(\mathcal{A}_{1,\lambda}, \mathcal{A}_{2,\lambda})\}_{\lambda\in\mathbb{N}}$, where the depth of $\mathcal{A}_{2,\lambda}$ is bounded from above by $T^{\underline{\epsilon}}(\lambda)$ for some $T(\cdot) \geq \tilde{T}_{\mathtt{TPKE}}(\cdot)$ we can construct a polynomial-size adversary $\mathcal{B} = \{(\mathcal{B}_{1,\lambda}, \mathcal{B}_{2,\lambda})\}_{\lambda\in\mathbb{N}}$ where the depth of $\mathcal{B}_{2,\lambda}$ is at most $T^{\epsilon}(\lambda)$ with*

$$\mathbf{Adv}^{\mathtt{TRE\text{-}PP}}_{\mathcal{B}} = |\Pr[\mathsf{G}_0 = 1] - \Pr[\mathsf{G}_1 = 1]|\,.$$

We construct an adversary $\mathcal{B} = \{(\mathcal{B}_{1,\lambda}, \mathcal{B}_{2,\lambda})\}_{\lambda\in\mathbb{N}}$ as follows.

The adversary $\mathcal{B}_{1,\lambda}$:

1. Runs $(\mathsf{pk}_1, \mathsf{sk}_1) \leftarrow \mathtt{TBE.Gen}(1^\lambda)$.
2. Runs $(m_0, m_1, \mathsf{st}) \leftarrow \mathcal{A}_{1,\lambda}(\mathsf{pk}_1)$ and answers decryption queries exactly as in Dec_f using the key sk_1.
3. Outputs the state $\mathsf{st}' = (\mathsf{pk}_1, \mathsf{sk}_1, m_0, m_1, \mathsf{st})$.

The adversary $\mathcal{B}_{2,\lambda}(\mathsf{pp}_e, \mathsf{pp}_d, \mathsf{st}')$:

1. Parses st' as $(\mathsf{pk}_1, \mathsf{sk}_1, m_0, m_1, \mathsf{st})$.
2. Randomly picks $b \xleftarrow{\$} \{0,1\}$, sends m_b to the challenger, and obtains a ciphertext c_2 as a response.
3. Runs $(\mathsf{vk}_{\mathsf{OT}}, \mathsf{sk}_{\mathsf{OT}}) \leftarrow \mathtt{OTS.Gen}(1^\lambda)$, and computes $c_1 \leftarrow \mathtt{TBE.Enc}(\mathsf{pk}_1, \mathsf{vk}_{\mathsf{OT}}, m_b)$, $\sigma \leftarrow \mathtt{OTS.Sign}(\mathsf{vk}_{\mathsf{OT}}, (\mathsf{pp}_d, c_1, c_2))$.
4. Runs $b' \leftarrow \mathcal{A}_{2,\lambda}((\mathsf{vk}_{\mathsf{OT}}, \mathsf{pp}_d, c_1, c_2, \sigma), \mathsf{st})$. Decryption queries are answered as before.
5. Returns the truth value of $b = b'$.

If c_2 is an encryption of the value m_b, then $\mathcal{B}$ perfectly simulates G_0, otherwise c_2 is an encryption of a random message and $\mathcal{B}$ perfectly simulates G_1. Hence, we obtain

$$\mathbf{Adv}^{\mathtt{TRE\text{-}PP}}_{\mathcal{B}} = |\Pr[\mathsf{G}_0 = 1] - \Pr[\mathsf{G}_1 = 1]|\,.$$

Moreover, $\mathcal{B}_2$ fulfils the depth constraint:

$$\mathsf{depth}(\mathcal{B}_{2,\lambda}) = \mathsf{poly}_{\mathcal{B}}(\lambda) + \mathsf{depth}(\mathcal{A}_{2,\lambda}) \leq \underline{T}^{\underline{\epsilon}}(\lambda) + T^{\underline{\epsilon}}(\lambda) \leq 2T^{\underline{\epsilon}}(\lambda) = o(T^{\epsilon}(\lambda)).$$

Also $T(\cdot) \geq \tilde{T}_{\mathtt{TPKE}}(\cdot) \geq \tilde{T}_{\mathtt{TRE}}(\cdot)$ as required.

Lemma 4.5.5. *From any polynomial-size adversary* $\mathcal{A} = \{(\mathcal{A}_{1,\lambda}, \mathcal{A}_{2,\lambda})\}_{\lambda\in\mathbb{N}}$ *we can construct polynomial-size adversaries* $\mathcal{F} = \{\mathcal{F}_\lambda\}_{\lambda\in\mathbb{N}}$ *and* $\mathcal{B}' = \{\mathcal{B}'_\lambda\}_{\lambda\in\mathbb{N}}$ *such that* $|\Pr[\mathsf{G}_1 = 1] - 1/2| \leq \mathbf{Adv}^{\mathtt{OTS}}_{\mathcal{F}} + \mathbf{Adv}^{\mathtt{TBE}}_{\mathcal{B}'}$.

Let $\mathcal{A} = \{(\mathcal{A}_{1,\lambda}, \mathcal{A}_{2,\lambda})\}_{\lambda\in\mathbb{N}}$ be an adversary in game G_1. Let $(\mathsf{vk}^*_{\mathsf{OT}}, \mathsf{pp}^*_d, c^*_1, c^*_2, \sigma^*)$ be the challenge ciphertext and let FRG denote the event that $\mathcal{A} = \{(\mathcal{A}_{1,\lambda}, \mathcal{A}_{2,\lambda})\}_{\lambda\in\mathbb{N}}$ submits a valid ciphertext $(\mathsf{vk}^*_{\mathsf{OT}}, \mathsf{pp}_d, c_1, c_2, \sigma)$ to the decryption oracle in G_1, where $(\mathsf{pp}^*_d, c^*_1, c^*_2, \sigma^*) \neq (\mathsf{pp}_d, c_1, c_2, \sigma)$. We prove the lemma in two steps. At first we prove that

$$\Pr[\mathsf{FRG}] \leq \mathbf{Adv}^{\mathtt{OTS}}_{\mathcal{F}}. \tag{4.1}$$

In the second step we prove the following:

$$\left|\Pr[\mathsf{G}_1 = 1 \wedge \overline{\mathsf{FRG}}] + \frac{1}{2}\Pr[\mathsf{FRG}] - \frac{1}{2}\right| \leq \mathbf{Adv}^{\mathtt{TBE}}_{\mathcal{B}'}. \tag{4.2}$$

Notice that

$$\begin{aligned}\left|\Pr[\mathsf{G}_1 = 1] - \frac{1}{2}\right| \leq& \left|\Pr[\mathsf{G}_1 = 1 \wedge \mathsf{FRG}] - \frac{1}{2}\Pr[\mathsf{FRG}]\right| + \\ &\left|\Pr[\mathsf{G}_1 = 1 \wedge \overline{\mathsf{FRG}}] + \frac{1}{2}\Pr[\mathsf{FRG}] - \frac{1}{2}\right| \\ \leq& \frac{1}{2}\Pr[\mathsf{FRG}] + \left|\Pr[\mathsf{G}_1 = 1 \wedge \overline{\mathsf{FRG}}] + \frac{1}{2}\Pr[\mathsf{FRG}] - \frac{1}{2}\right|.\end{aligned}$$

We start with the first inequality. Assuming that the event FRG has happened, we construct a forger $\mathcal{F}$ against the strong one-time signature scheme.

The adversary $\mathcal{F}_\lambda(\mathsf{vk}^*_{\mathsf{OT}})$:

1. Runs $(\mathsf{pk}_1, \mathsf{sk}_1) \leftarrow \mathtt{TBE}.\mathsf{Gen}(1^\lambda)$.
2. Runs $(m_0, m_1, \mathsf{st}) \leftarrow \mathcal{A}_{1,\lambda}(\mathsf{pk}_1)$ and answers decryption queries exactly as in Dec_f using the key sk_1. If $\mathcal{A}_{1,\lambda}(\mathsf{pk}_1)$ submits a valid ciphertext $(\mathsf{vk}^*_{\mathsf{OT}}, \mathsf{pp}_d, c_1, c_2, \sigma)$ to its decryption oracle before requesting the challenge ciphertext, then output the forgery $((\mathsf{pp}_d, c_1, c_2), \sigma)$ and stop.
3. Otherwise, samples $b \xleftarrow{\$} \{0,1\}$ and runs $(\mathsf{pp}_e, \mathsf{pp}^*_d) \leftarrow \mathtt{TRE}.\mathsf{Setup}(1^\lambda, T), c^*_1 \leftarrow \mathtt{TBE}.\mathsf{Enc}(\mathsf{pk}_1, \mathsf{vk}^*_{\mathsf{OT}}, m_b), c^*_2 \leftarrow \mathtt{TRE}.\mathsf{Enc}(\mathsf{pp}_e, m')$, where $m' \xleftarrow{\$} \mathcal{M}$ is a random message such that $|m'| = |m_0|$.
4. Obtains signature σ^* on $(\mathsf{pp}^*_d, c^*_1, c^*_2)$ from $\mathsf{SIGN}(\cdot)$.

5. Runs $\mathcal{A}_{2,\lambda}((\mathsf{vk}^*_{\mathsf{OT}}, \mathsf{pp}^*_d, c^*_1, c^*_2, \sigma^*), \mathsf{st})$ and answers decryption queries as before. If $\mathcal{A}_{2,\lambda}$ submits a valid ciphertext $(\mathsf{vk}^*_{\mathsf{OT}}, \mathsf{pp}_d, c_1, c_2, \sigma)$ to its decryption oracle, then we must have $(\mathsf{pp}^*_d, c^*_1, c^*_2, \sigma^*) \neq (\mathsf{pp}_d, c_1, c_2, \sigma)$. Therefore, it outputs $((\mathsf{pp}_d, c_1, c_2), \sigma)$ as a forgery.

It is easy to see that $\Pr[\mathsf{FRG}] \leq \mathbf{Adv}^{\mathtt{OTS}}_{\mathcal{F}}$.

Now we prove the second inequality. We construct an adversary $\mathcal{B}'$ which breaks the security of the tag-based encryption scheme as follows.

The adversary $\mathcal{B}'_\lambda$:

1. Runs $(\mathsf{vk}^*_{\mathsf{OT}}, \mathsf{sk}^*_{\mathsf{OT}}) \leftarrow \mathtt{OTS}.\mathsf{Gen}(1^\lambda)$ and outputs tag $\mathsf{vk}^*_{\mathsf{OT}}$ to its challenger.
2. Obtains pk_1 from the challenger.
3. Runs $(m_0, m_1, \mathsf{st}) \leftarrow \mathcal{A}_{1,\lambda}(\mathsf{pk}_1)$ and answers queries $(\mathsf{vk}_{\mathsf{OT}}, Z, c_1, c_2, \sigma)$ to the decryption oracle as follows:
   - If $\mathsf{vk}_{\mathsf{OT}} = \mathsf{vk}^*_{\mathsf{OT}}$ checks if $\mathtt{OTS}.\mathsf{Vrfy}(\mathsf{vk}^*_{\mathsf{OT}}, (\mathsf{pp}_d, c_1, c_2), \sigma) = 1$. If so, it outputs a random bit to its challenger and aborts. Otherwise, it returns $\perp$.
   - If $\mathsf{vk}_{\mathsf{OT}} \neq \mathsf{vk}^*_{\mathsf{OT}}$ and $\mathtt{OTS}.\mathsf{Vrfy}(\mathsf{vk}_{\mathsf{OT}}, (\mathsf{pp}_d, c_1, c_2), \sigma) = 0$, then returns $\perp$.
   - If $\mathsf{vk}_{\mathsf{OT}} \neq \mathsf{vk}^*_{\mathsf{OT}}$ and $\mathtt{OTS}.\mathsf{Vrfy}(\mathsf{vk}_{\mathsf{OT}}, (\mathsf{pp}_d, c_1, c_2), \sigma) = 1$, then forwards $(c_1, \mathsf{vk}_{\mathsf{OT}})$ to the challenger and returns the response to $\mathcal{A}_{1,\lambda}$.
4. Forwards messages (m_0, m_1) to the challenger and obtains c^*_1.
5. Runs $(\mathsf{pp}_e, \mathsf{pp}^*_d) \leftarrow \mathtt{TRE}.\mathsf{Setup}(1^\lambda, T)$, samples $m' \xleftarrow{\$} \mathcal{M}$ such that $|m'| = |m_0|$ and computes $c^*_2 \leftarrow \mathtt{TRE}.\mathsf{Enc}(\mathsf{pp}_e, m'), \sigma^* \leftarrow \mathtt{OTS}.\mathsf{Sign}(\mathsf{sk}^*_{\mathsf{OT}}, (\mathsf{pp}^*_d, c^*_1, c^*_2))$.
6. Runs $b' \leftarrow \mathcal{A}_{2,\lambda}((\mathsf{vk}^*_{\mathsf{OT}}, \mathsf{pp}^*_d, c^*_1, c^*_2, \sigma^*), \mathsf{st})$ and answers decryption queries as before.
7. Outputs b'.

Notice that $\mathcal{B}'$ never requests a decryption for the tag $\mathsf{vk}^*_{\mathsf{OT}}$ and moreover it provides a perfect simulation until the event FRG occurs. It follows that:

$$\mathbf{Adv}^{\mathtt{TBE}}_{\mathcal{B}'} = |\Pr[\mathsf{G}_1 = 1 \wedge \overline{\mathsf{FRG}}] + \frac{1}{2}\Pr[\mathsf{FRG}] - \frac{1}{2}|.$$

By combining Lemmas 4.5.4-4.5.5 we obtain following:

$$\begin{aligned}
\mathbf{Adv}^{\mathtt{TPKE}}_{\mathcal{A}} &= \left|\Pr[\mathsf{G}_0 = 1] - \frac{1}{2}\right| \\
&\leq |\Pr[\mathsf{G}_0 = 1] - \Pr[\mathsf{G}_1 = 1]| + \left|\Pr[\mathsf{G}_1 = 1] - \frac{1}{2}\right| \\
&\leq \mathbf{Adv}^{\mathtt{TRE\text{-}PP}}_{\mathcal{B}} + \mathbf{Adv}^{\mathtt{OTS}}_{\mathcal{F}} + \mathbf{Adv}^{\mathtt{TBE}}_{\mathcal{B}'},
\end{aligned}$$

which concludes the proof. □

4.5.3 Construction of IND-CCA-secure TPKE from SSSA

Finally, we construct an IND-CCA-secure TPKE scheme where the running time of the encryption algorithm is independent of the hardness parameter T. Let $\mathtt{TBE} = (\mathtt{TBE.Gen}, \mathtt{TBE.Enc}, \mathtt{TBE.Dec})$ be a tag-based encryption scheme, $\mathtt{PKE} = (\mathtt{PKE.Gen}, \mathtt{PKE.Enc}, \mathtt{PKE.Dec})$ be a public-key encryption scheme and $\mathtt{OTS} = (\mathtt{OTS.Gen}, \mathtt{OTS.Sign}, \mathtt{OTS.Vrfy})$ be a signature scheme. The construction is given in Figure 4.13 and it is not generic anymore. We rely on the strong sequential squaring assumption and we use algebraic properties of the underlying group $\mathbb{QR}_N$.

$\underline{\mathsf{KGen}(1^\lambda), T}$
$(\mathsf{pk}_1, \mathsf{sk}_1) \leftarrow \mathtt{TBE.Gen}(1^\lambda)$
$(p, q, N) \leftarrow \mathsf{GenMod}(1^\lambda)$
$\varphi(N) := (p-1)(q-1)$
$g \xleftarrow{\$} \mathbb{QR}_N$
$t := 2^T \bmod \varphi(N)/4$
$h := g^t \bmod N$
return $(\mathsf{pk} := (\mathsf{pk}_1, N, T, g, h), \mathsf{sk} := \mathsf{sk}_1)$

$\underline{\mathsf{Enc}(\mathsf{pk}, m)}$
$r \xleftarrow{\$} [\lfloor N/4 \rfloor]$
Compute $x := g^r \bmod N$
Compute $y := h^r \bmod N$
$(\mathsf{pk}_2, \mathsf{sk}_2) \leftarrow \mathtt{PKE.Gen}(1^\lambda; y)$
$(\mathsf{vk}_{\mathsf{OT}}, \mathsf{sk}_{\mathsf{OT}}) \leftarrow \mathtt{OTS.Gen}(1^\lambda)$
$c_1 \leftarrow \mathtt{TBE.Enc}(\mathsf{pk}_1, \mathsf{vk}_{\mathsf{OT}}, m)$
$c_2 \leftarrow \mathtt{PKE.Enc}(\mathsf{pk}_2, m)$
$\sigma \leftarrow \mathtt{OTS.Sign}(\mathsf{sk}_{\mathsf{OT}}, (x, c_1, c_2))$
return $c := (\mathsf{vk}_{\mathsf{OT}}, x, c_1, c_2, \sigma)$

$\underline{\mathsf{Dec}_s(\mathsf{pk}, c)}$
Parse c as $(\mathsf{vk}_{\mathsf{OT}}, x, c_1, c_2, \sigma)$
if $\mathtt{OTS.Vrfy}(\mathsf{vk}_{\mathsf{OT}}, (x, c_1, c_2), \sigma) = 1$
  Compute $y := x^{2^T} \bmod N$
  $(\mathsf{pk}', \mathsf{sk}') \leftarrow \mathtt{PKE.Gen}(1^\lambda; y)$
  return $m \leftarrow \mathtt{PKE.Dec}(\mathsf{sk}', c')$
return $\perp$

$\underline{\mathsf{Dec}_f(\mathsf{sk}, c)}$
Parse c as $(\mathsf{vk}_{\mathsf{OT}}, x, c_1, c_2, \sigma)$
if $\mathtt{OTS.Vrfy}(\mathsf{vk}_{\mathsf{OT}}, (x, c_1, c_2), \sigma) = 1$
  return $m \leftarrow \mathtt{TBE.Dec}(\mathsf{sk}, \mathsf{vk}_{\mathsf{OT}}, c_1)$
return $\perp$

Figure 4.13: Construction of TPKE

Theorem 4.5.4. *If* $\mathtt{TBE} = (\mathtt{TBE.Gen}, \mathtt{TBE.Enc}, \mathtt{TBE.Dec})$ *is a tag-based encryption scheme that is selective-tag weakly secure against chosen ciphertext attacks,* $\mathtt{OTS} = (\mathtt{OTS.Gen}, \mathtt{OTS.Sign}, \mathtt{OTS.Vrfy})$ *is a strong one-time signature scheme,* $\mathtt{PKE} = (\mathtt{PKE.Gen}, \mathtt{PKE.Enc}, \mathtt{PKE.Dec})$ *is an RoR-CPA-secure public-key encryption scheme and the strong sequential squaring assumption with gap* ϵ *holds relative to* GenMod*, then* $(\mathsf{KGen}, \mathsf{Enc}, \mathsf{Dec}_f, \mathsf{Dec}_s)$ *defined in Figure 4.13 is an IND-CCA-secure timed public-key encryption scheme with* $\underline{\epsilon}$*, for any* $\underline{\epsilon} < \epsilon$*.*

Proof. Correctness of the scheme is implied by the correctness of the tag-based encryption scheme, the signature scheme, and the public-key encryption scheme.

To prove security we define a sequence of games $\mathsf{G}_0 - \mathsf{G}_3$. For $i \in \{0,1,2,3\}$ we denote by $\mathsf{G}_i = 1$ the event that the adversary $\mathcal{A} = \{(\mathcal{A}_{1,\lambda}, \mathcal{A}_{2,\lambda})\}_{\lambda\in\mathbb{N}}$ outputs b' in the game G_i such that $b = b'$.

Game 0. Game G_0 corresponds to the original security experiment. Decryption queries are answered using sk_1.

Let GNR denote the event that the sampled g in KGen is a generator of $\mathbb{QR}_N$. Recall that $N = pq$ where $p = 2p' + 1$ and $q = 2q' + 1$. Because g is sampled uniformly at random and $\mathbb{QR}_N$ has $\varphi(|\mathbb{QR}_N|) = (p'-1)(q'-1)$ generators, this event happens with overwhelming probability. Concretely, $\Pr[\mathsf{GNR}] = 1 - \frac{1}{p'} - \frac{1}{q'} + \frac{1}{p'q'}$. Therefore the following holds.

Lemma 4.5.6.

$$\begin{aligned}\Pr[\mathsf{G}_0 = 1] &= \Pr[\mathsf{G}_0 = 1|\mathsf{GNR}]\Pr[\mathsf{GNR}] + \Pr[\mathsf{G}_0 = 1|\overline{\mathsf{GNR}}]\Pr[\overline{\mathsf{GNR}}] \\ &\leq \Pr[\mathsf{G}_0 = 1|\mathsf{GNR}]\Pr[\mathsf{GNR}] + \Pr[\overline{\mathsf{GNR}}] \\ &= \Pr[\mathsf{G}_0 = 1|\mathsf{GNR}]\left(1 - \frac{1}{p'} - \frac{1}{q'} + \frac{1}{p'q'}\right) + \frac{1}{p'} + \frac{1}{q'} - \frac{1}{p'q'}.\end{aligned}$$

Game 1. In G_1 we sample r uniformly at random from $[|\mathbb{QR}_N|]$.

Lemma 4.5.7.

$$|\Pr[\mathsf{G}_0 = 1|\mathsf{GNR}] - \Pr[\mathsf{G}_1 = 1|\mathsf{GNR}]| \leq \left(\frac{1}{p} + \frac{1}{q} - \frac{1}{N}\right).$$

At first we remark that for upper bounding the difference between the games we use a statistical argument. Because r appears only in the exponent of the group generator, we later sample a random element x from the group $\mathbb{QR}_N$ which can be done efficiently. Since the only difference between the two games is in the set from which we sample r, the advantage of any adversary can be upper bounded by the statistical distance between two random variables X and Y both defined on domain $[\lfloor N/4 \rfloor]$ as follows:

$$\Pr[X = r] = 1/\lfloor N/4 \rfloor \ \ \forall r \in [\lfloor N/4 \rfloor] \text{ and } \Pr[Y = r] = \begin{cases} 4/\varphi(N) & \forall r \in [\varphi(N)/4] \\ 0 & \text{otherwise} \end{cases}$$

Clearly, X has the same distribution as a sampling in G_0 and Y has the same distribution as a sampling in G_1. The following computation proves the lemma:

$$\mathbb{SD}(X,Y) = \frac{1}{2}\sum_{r\in[\lfloor N/4\rfloor]} |\Pr[X=r]-\Pr[Y=r]| =$$

$$\frac{1}{2}\left(\sum_{r=1}^{\varphi(N)/4} |\Pr[X=r]-\Pr[Y=r]| + \sum_{r=\varphi(N)/4+1}^{\lfloor N/4\rfloor} |\Pr[X=r]-\Pr[Y=r]|\right) =$$

$$\frac{1}{2}\left(\sum_{r=1}^{\varphi(N)/4} \left|\frac{1}{\lfloor N/4\rfloor}-\frac{4}{\varphi(N)}\right| + \sum_{r=\varphi(N)/4+1}^{\lfloor N/4\rfloor} \left|\frac{1}{\lfloor N/4\rfloor}-0\right|\right) \leq$$

$$\frac{1}{2}\left(\sum_{r=1}^{\varphi(N)/4} \left|\frac{4}{N}-\frac{4}{\varphi(N)}\right| + \sum_{r=\varphi(N)/4+1}^{\lfloor N/4\rfloor} \left|\frac{1}{\lfloor N/4\rfloor}-0\right|\right) =$$

$$\frac{1}{2}\left(\varphi(N)/4\left|\frac{4(\varphi(N)-N)}{\varphi(N)N}\right| + (\lfloor N/4\rfloor-\varphi(N)/4)\frac{1}{\lfloor N/4\rfloor}\right) =$$

$$\frac{1}{2}\left(\frac{(N-\varphi(N))}{N}+1-\frac{\varphi(N)/4}{\lfloor N/4\rfloor}\right) \leq \frac{1}{2}\left(\frac{(N-\varphi(N))}{N}+1-\frac{\varphi(N)/4}{N/4}\right) =$$

$$\frac{1}{2}\frac{2(N-\varphi(N))}{N} = \frac{(N-(N-p-q+1))}{N} = \frac{p+q-1}{N} = \frac{1}{p}+\frac{1}{q}-\frac{1}{N}.$$

Game 2. In G_2 we sample y uniformly at random from $\mathbb{QR}_N$.

Let $\tilde{T}_{\mathsf{SSS}}(\lambda)$ be the polynomial whose existence is guaranteed by the SSS assumption. Let $\mathsf{poly}_{\mathcal{B}}(\lambda)$ be the fixed polynomial which bounds the time required to execute Steps 1–3 and answer decryption queries in Step 4 of the adversary $\mathcal{B}_{2,\lambda}$ defined below. Set $\underline{T} := (\mathsf{poly}_{\mathcal{B}}(\lambda))^{1/\epsilon}$. Set $\tilde{T}_{\mathtt{TPKE}} := \max(\tilde{T}_{\mathsf{SSS}}, \underline{T})$.

Lemma 4.5.8. *From any polynomial-size adversary $\mathcal{A} = \{(\mathcal{A}_{1,\lambda}, \mathcal{A}_{2,\lambda})\}_{\lambda\in\mathbb{N}}$, where depth of $\mathcal{A}_{2,\lambda}$ is at most $T^{\epsilon}(\lambda)$ for some $T(\cdot) \geq \underline{T}(\cdot)$ we can construct a polynomial-size adversary $\mathcal{B} = \{(\mathcal{B}_{1,\lambda}, \mathcal{B}_{2,\lambda})\}_{\lambda\in\mathbb{N}}$ where the depth of $\mathcal{B}_{2,\lambda}$ is at most $T^{\epsilon}(\lambda)$ with*

$$\mathbf{Adv}^{\mathsf{SSS}}_{\mathcal{B}} = |\Pr[\mathsf{G}_1 = 1|\mathsf{GNR}] - \Pr[\mathsf{G}_2 = 1|\mathsf{GNR}]|\,.$$

We construct an adversary $\mathcal{B} = \{(\mathcal{B}_{1,\lambda}, \mathcal{B}_{2,\lambda})\}_{\lambda\in\mathbb{N}}$ as follows.
The adversary $\mathcal{B}_{1,\lambda}(N,T)$:

1. Runs $(\mathsf{pk}_1, \mathsf{sk}_1) \leftarrow \mathtt{TBE.Gen}(1^\lambda)$.
2. Samples $g \xleftarrow{\$} \mathbb{QR}_N$ and computes $h := g^{2^T} \bmod N$ by repeated squaring.
3. Sets $pk := (\mathsf{pk}_1, N, T, g, h)$, runs $(m_0, m_1, \mathsf{st}) \leftarrow \mathcal{A}_{1,\lambda}(\mathsf{pk})$ and answers decryption queries exactly as in Dec_f using the key sk_1.
4. Outputs $\mathsf{st}' = (\mathsf{pk}_1, \mathsf{sk}_1, m_0, m_1, \mathsf{st})$.

The adversary $\mathcal{B}_{2,\lambda}(x, y, \mathsf{st}')$:

1. Parses st' as $(\mathsf{pk}_1, \mathsf{sk}_1, m_0, m_1, \mathsf{st})$.
2. Randomly picks $b \xleftarrow{\$} \{0,1\}$.
3. Computes $(\mathsf{sk}_2, \mathsf{pk}_2) \leftarrow \mathtt{PKE}.\mathsf{Gen}(1^\lambda; y), (\mathsf{vk}_{\mathsf{OT}}, \mathsf{sk}_{\mathsf{OT}}) \leftarrow \mathtt{OTS}.\mathsf{Gen}(1^\lambda), c_1 \leftarrow \mathtt{TBE}.\mathsf{Enc}(\mathsf{pk}_1, \mathsf{vk}_{\mathsf{OT}}, m_b), c_2 \leftarrow \mathtt{PKE}.\mathsf{Enc}(\mathsf{pk}_2, m_b), \sigma \leftarrow \mathtt{OTS}.\mathsf{Sign}(\mathsf{vk}_{\mathsf{OT}}, (x, c_1, c_2))$.
4. Runs $b' \leftarrow \mathcal{A}_{1,\lambda}((\mathsf{vk}_{\mathsf{OT}}, x, c_1, c_2, \sigma), \mathsf{st})$. Decryption queries are answered as before.
5. Returns the truth value of $b = b'$.

If the event GNR happens, then g must be generator of $\mathbb{QR}_N$ and therefore there must exist some r such that $x = g^r \mod N$. Hence, if $y = x^{2^T} \mod N$, then $\mathcal{B}$ perfectly simulates G_1, otherwise y is a random value and $\mathcal{B}$ perfectly simulates G_2. Therefore, we obtain

$$\mathbf{Adv}^{\mathsf{SSS}}_{\mathcal{B}} = |\Pr[\mathsf{G}_1 = 1|\mathsf{GNR}] - \Pr[\mathsf{G}_2 = 1|\mathsf{GNR}]|\,.$$

Adversary $\mathcal{B}_1$ computes h by T consecutive squarings and because T is polynomial in λ, $\mathcal{B}_1$ is efficient. Moreover, $\mathcal{B}_2$ fulfils the depth constraint:

$$\mathsf{depth}(\mathcal{B}_{2,\lambda}) = \mathsf{poly}_{\mathcal{B}}(\lambda) + \mathsf{depth}(\mathcal{A}_{2,\lambda}) \leq \underline{T}^{\epsilon}(\lambda) + T^{\epsilon}(\lambda) \leq 2T^{\epsilon}(\lambda) = o(T^{\epsilon}(\lambda)).$$

Also $T(\cdot) \geq \tilde{T}_{\mathtt{TPKE}}(\cdot) \geq \tilde{T}_{\mathsf{SSS}}(\cdot)$ as required.

Game 3. In G_3 instead of encrypting the given message twice, we compute c_2 as an encryption of a random message.

Lemma 4.5.9. *From any polynomial-size adversary $\mathcal{A} = \{(\mathcal{A}_{1,\lambda}, \mathcal{A}_{2,\lambda})\}_{\lambda\in\mathbb{N}}$ we can construct a polynomial size adversary $\mathcal{B}' = \{\mathcal{B}'_\lambda\}_{\lambda\in\mathbb{N}}$ such that*

$$|\Pr[\mathsf{G}_2 = 1|\mathsf{GNR}] - \Pr[\mathsf{G}_3 = 1|\mathsf{GNR}]| = \mathbf{Adv}^{\mathsf{PKE},\mathsf{RoR}}_{\mathcal{B}'}.$$

We construct an adversary $\mathcal{B}'$ as follows.
The adversary $\mathcal{B}'_\lambda(\mathsf{pk}_2)$:

1. Runs $(\mathsf{pk}, \mathsf{sk}) \leftarrow \mathsf{KGen}(1^\lambda)$ as described in the construction.
2. Runs $(m_0, m_1, \mathsf{st}) \leftarrow \mathcal{A}_{1,\lambda}(\mathsf{pk})$ and answers decryption queries exactly as in Dec_f using the key sk.
3. Randomly samples $b \xleftarrow{\$} \{0,1\}$, $x \xleftarrow{\$} \mathbb{QR}_N$, and computes $(\mathsf{vk}_{\mathsf{OT}}, \mathsf{sk}_{\mathsf{OT}}) \leftarrow \mathtt{OTS}.\mathsf{Gen}(1^\lambda), c_1 \leftarrow \mathtt{TBE}.\mathsf{Enc}(\mathsf{pk}, \mathsf{vk}_{\mathsf{OT}}, m_b)$.

4. Sends m_b to challenger and obtain ciphertext c_2 as a response.

5. Signs the ciphertext $\sigma \leftarrow \mathtt{OTS}.\mathsf{Sign}(\mathsf{sk}_{\mathsf{OT}}, (x, c_1, c_2))$.

6. Runs $b' \leftarrow \mathcal{A}_{1,\lambda}((\mathsf{vk}_{\mathsf{OT}}, x, c_1, c_2, \sigma), \mathsf{st})$. Decryption queries are answered as before.

7. Returns the truth value of $b = b'$.

If c_2 is an encryption of the value m_b, then $\mathcal{B}'$ perfectly simulates G_2, otherwise c_2 is an encryption of a random value and $\mathcal{B}'$ perfectly simulates G_3. Hence, we obtain

$$\mathbf{Adv}^{\mathtt{PKE},\mathsf{RoR}}_{\mathcal{B}'} = |\Pr[\mathsf{G}_2 = 1|\mathsf{GNR}] - \Pr[\mathsf{G}_3 = 1|\mathsf{GNR}]|\,.$$

Lemma 4.5.10. *From any polynomial-size adversary $\mathcal{A} = \{(\mathcal{A}_{1,\lambda}, \mathcal{A}_{2,\lambda})\}_{\lambda\in\mathbb{N}}$ we can construct polynomial-size adversaries $\mathcal{F} = \{\mathcal{F}_\lambda\}_{\lambda\in\mathbb{N}}$ and $\mathcal{B}'' = \{\mathcal{B}''_\lambda\}_{\lambda\in\mathbb{N}}$ such that*

$$|\Pr[\mathsf{G}_3 = 1|\mathsf{GNR}] - 1/2| \leq \mathbf{Adv}^{\mathtt{OTS}}_{\mathcal{F}} + \mathbf{Adv}^{\mathtt{TBE}}_{\mathcal{B}''}.$$

Let $\mathcal{A} = \{(\mathcal{A}_{1,\lambda}, \mathcal{A}_{2,\lambda})\}_{\lambda\in\mathbb{N}}$ be an adversary in game G_3. Let $(\mathsf{vk}^*_{\mathsf{OT}}, x^*, c_1^*, c_2^*, \sigma^*)$ be the challenge ciphertext and let FRG denote the event that $\mathcal{A}$ submits a valid ciphertext $(\mathsf{vk}^*_{\mathsf{OT}}, x, c_1, c_2, \sigma)$ to the decryption oracle in the game G_3, where $(x^*, c_1^*, c_2^*, \sigma^*) \neq (x, c_1, c_2, \sigma)$. We prove the lemma in two steps. At first we prove that

$$\Pr[\mathsf{FRG}|\mathsf{GNR}] \leq \mathbf{Adv}^{\mathtt{OTS}}_{\mathcal{F}}. \tag{4.3}$$

In the second step we prove the following:

$$\left|\Pr[\mathsf{G}_3 = 1 \wedge \overline{\mathsf{FRG}}|\mathsf{GNR}] + \frac{1}{2}\Pr[\mathsf{FRG}|\mathsf{GNR}] - \frac{1}{2}\right| \leq \mathbf{Adv}^{\mathtt{TBE}}_{\mathcal{B}''}. \tag{4.4}$$

Notice that

$$\begin{aligned}\left|\Pr[\mathsf{G}_3 = 1|\mathsf{GNR}] - \frac{1}{2}\right| &\leq \left|\Pr[\mathsf{G}_3 = 1 \wedge \mathsf{FRG}|\mathsf{GNR}] - \frac{1}{2}\Pr[\mathsf{FRG}|\mathsf{GNR}]\right| + \\ &\qquad \left|\Pr[\mathsf{G}_3 = 1 \wedge \overline{\mathsf{FRG}}|\mathsf{GNR}] + \frac{1}{2}\Pr[\mathsf{FRG}|\mathsf{GNR}] - \frac{1}{2}\right| \\ &\leq \frac{1}{2}\Pr[\mathsf{FRG}|\mathsf{GNR}] + \left|\Pr[\mathsf{G}_3 = 1 \wedge \overline{\mathsf{FRG}}|\mathsf{GNR}] + \frac{1}{2}\Pr[\mathsf{FRG}|\mathsf{GNR}] - \frac{1}{2}\right|.\end{aligned}$$

We start with the first inequality. Assuming that the event FRG has happened, we construct a forger $\mathcal{F}$ against the strong one-time signature scheme.

The adversary $\mathcal{F}_\lambda(\mathsf{vk}^*_{\mathsf{OT}})$:

1. Runs $(\mathsf{pk}, \mathsf{sk}) \leftarrow \mathsf{KGen}(1^\lambda)$ as described in the construction.

2. Runs $(m_0, m_1, \mathsf{st}) \leftarrow \mathcal{A}_{1,\lambda}(\mathsf{pk})$ and answers decryption queries exactly as in Dec_f using the key sk. If $\mathcal{A}_{1,\lambda}$ submits a valid ciphertext $(\mathsf{vk}^*_{\mathsf{OT}}, x, c_1, c_2, \sigma)$ to its decryption oracle before requesting the challenge ciphertext, then outputs the forgery $((x, c_1, c_2), \sigma)$ and stops.

3. Otherwise, samples $b \xleftarrow{\$} \{0,1\}, x \xleftarrow{\$} \mathbb{QR}_N$ and runs $(\mathsf{sk}_2, \mathsf{pk}_2) \leftarrow \mathtt{PKE}.\mathsf{Gen}(1^\lambda)$, $c_1^* \leftarrow \mathtt{TBE}.\mathsf{Enc}(\mathsf{pk}, \mathsf{vk}^*_{\mathsf{OT}}, m_b)$, $c_2^* \xleftarrow{\$} \mathtt{PKE}.\mathsf{Enc}(\mathsf{pk}_2, m')$, where $m' \xleftarrow{\$} \mathcal{M}$ is a random message such that $|m'| = |m_0|$.

4. Obtains signature σ^* on (x^*, c_1^*, c_2^*) from $\mathsf{SIGN}(\cdot)$.

5. Runs $b' \leftarrow \mathcal{A}_{2,\lambda}((\mathsf{vk}^*_{\mathsf{OT}}, x^*, c_1^*, c_2^*, \sigma^*), \mathsf{st})$ and answers decryption queries as before. If $\mathcal{A}_{2,\lambda}$ submits a valid ciphertext $(\mathsf{vk}^*_{\mathsf{OT}}, x, c_1, c_2, \sigma)$ to its decryption oracle, then we must have $(x^*, c_1^*, c_2^*, \sigma^*) \neq (x, c_1, c_2, \sigma)$. Hence, it outputs $((x, c_1, c_2), \sigma)$ as a forgery.

It is easy to see that $\Pr[\mathsf{FRG}|\mathsf{GNR}] \leq \mathbf{Adv}^{\mathtt{OTS}}_{\mathcal{F}}$.

Now we prove the second inequality. We use $\mathcal{A}$ to construct $\mathcal{B}''$ which breaks the security of the tag-based encryption scheme:

The adversary $\mathcal{B}''_\lambda$:

1. Runs $(\mathsf{vk}^*_{\mathsf{OT}}, \mathsf{sk}^*_{\mathsf{OT}}) \leftarrow \mathtt{OTS}.\mathsf{Gen}(1^\lambda)$ and outputs tag $\mathsf{vk}^*_{\mathsf{OT}}$ to its challenger.

2. Obtains pk_1 from the challenger.

3. Runs $(p, q, N) \leftarrow \mathsf{GenMod}(1^\lambda)$, samples $g \xleftarrow{\$} \mathbb{QR}_N$ uniformly at random, and it computes $\varphi(N) := (p-1)(q-1), t := 2^T \bmod \varphi(N)/4, h := g^t \bmod N$. It sets $\mathsf{pk} := (\mathsf{pk}_1, N, T, g, h)$.

4. Runs $(m_0, m_1, \mathsf{st}) \leftarrow \mathcal{A}_{1,\lambda}(\mathsf{pk})$ and answers decryption queries of the form $(\mathsf{vk}_{\mathsf{OT}}, x, c_1, c_2, \sigma)$ as follows:
   - If $\mathsf{vk}_{\mathsf{OT}} = \mathsf{vk}^*_{\mathsf{OT}}$ checks if $\mathtt{OTS}.\mathsf{Vrfy}(\mathsf{vk}^*_{\mathsf{OT}}, (x, c_1, c_2), \sigma) = 1$. If so, it outputs a random bit to its challenger and aborts. Otherwise, it returns $\perp$.
   - If $\mathsf{vk}_{\mathsf{OT}} \neq \mathsf{vk}^*_{\mathsf{OT}}$ and $\mathtt{OTS}.\mathsf{Vrfy}(\mathsf{vk}_{\mathsf{OT}}, (x, c_1, c_2), \sigma) = 0$, then returns $\perp$.
   - If $\mathsf{vk}_{\mathsf{OT}} \neq \mathsf{vk}^*_{\mathsf{OT}}$ and $\mathtt{OTS}.\mathsf{Vrfy}(\mathsf{vk}_{\mathsf{OT}}, (x, c_1, c_2), \sigma) = 1$, then forwards $(c_1, \mathsf{vk}_{\mathsf{OT}})$ to the challenger and returns the response to $\mathcal{A}_{1,\lambda}$.

5. Forwards messages (m_0, m_1) to the challenger and obtains c_1^*. It computes $x^* \xleftarrow{\$} \mathbb{QR}_N, (\mathsf{sk}_2, \mathsf{pk}_2) \leftarrow \mathtt{PKE}.\mathsf{Gen}(1^\lambda)$, randomly samples $m' \xleftarrow{\$} \mathcal{M}$ such that $|m'| = |m_0|$ and runs $c_2^* \leftarrow \mathtt{PKE}.\mathsf{Enc}(\mathsf{pk}_2, m'), \sigma^* \leftarrow \mathtt{OTS}.\mathsf{Sign}(\mathsf{sk}^*_{\mathsf{OT}}, (x^*, c_1^*, c_2^*))$.

6. Outputs $b' \leftarrow \mathcal{A}_{2,\lambda}((\mathsf{vk}^*_{\mathsf{OT}}, x^*, c_1^*, c_2^*, \sigma^*), \mathsf{st})$. Decryption queries are answered as before.

Notice that $\mathcal{B}''$ never requests a decryption for the tag $\mathsf{vk}^*_{\mathsf{OT}}$ and moreover it provides a perfect simulation until the event FRG occurs. It follows that:

$$\mathbf{Adv}^{\mathtt{TBE}}_{\mathcal{B}''} = \left|\Pr[\mathsf{G}_3 = 1 \wedge \overline{\mathsf{FRG}}|\mathsf{GNR}] + \frac{1}{2}\Pr[\mathsf{FRG}|\mathsf{GNR}] - \frac{1}{2}\right|.$$

By combining Lemmas 4.5.6-4.5.10 we obtain following:

$$\begin{aligned}
&\mathbf{Adv}^{\mathtt{TPKE}}_{\mathcal{A}} = \left|\Pr[\mathsf{G}_0 = 1] - \frac{1}{2}\right| \\
&\leq \left|\Pr[\mathsf{GNR}]\Pr[\mathsf{G}_0 = 1|\mathsf{GNR}] + \Pr[\overline{\mathsf{GNR}}] - \frac{1}{2}\right| \\
&\leq \Pr[\mathsf{GNR}]\left(\sum_{i=0}^{2}|\Pr[\mathsf{G}_i = 1|\mathsf{GNR}] - \Pr[\mathsf{G}_{i+1} = 1|\mathsf{GNR}]| + \left|\Pr[\mathsf{G}_3 = 1|\mathsf{GNR}] - \frac{1}{2}\right|\right) \\
&+ \Pr[\overline{\mathsf{GNR}}] \\
&\leq \left(1 - \frac{1}{p'} - \frac{1}{q'} + \frac{1}{p'q'}\right)\left(\frac{1}{p} + \frac{1}{q} - \frac{1}{N} + \mathbf{Adv}^{\mathsf{SSS}}_{\mathcal{B}} + \mathbf{Adv}^{\mathsf{PKE,RoR}}_{\mathcal{B}'} + \mathbf{Adv}^{\mathtt{OTS}}_{\mathcal{F}}\right. \\
&\left. + \mathbf{Adv}^{\mathtt{TBE}}_{\mathcal{B}''}\right) + \frac{1}{p'} + \frac{1}{q'} - \frac{1}{p'q'},
\end{aligned}$$

which concludes the proof. □

4.6 Conclusion and Open Problems

We have proposed a new notion of (sequential) timed-release encryption which helps to overcome the barrier of deployment of time-lock puzzles in real-world applications. Our solution allows encrypting several messages with respect to one puzzle in such a way that by solving the puzzle we can decrypt all corresponding ciphertexts. This is achieved in our work by producing a puzzle in a trusted setup, which is reasonable for many interesting applications like e-voting, sealed-bid auctions, etc. Additionally, we are able to produce a set of puzzles with respect to different hardness parameters in a way that running only one sequential computation is sufficient in order to solve all of them. In this way, we can avoid wasting computational resources and make applications more sustainable. The advantage of our approach is that it is generic and hence can rely on different time-lock puzzles.

Moreover, we have provided more efficient constructions for timed public-key encryption which is used as a building block for non-malleable timed commitments. The state-of-the-art construction suffers from encryption whose runtime is proportional to the hardness parameter. Our constructions either have exponentially faster encryption with respect to the hardness parameter or the runtime of encryption is completely independent of the hardness parameter.

One of the most interesting open problems in the field of time-lock puzzles is to provide new constructions of them. There are currently only two known constructions, one based on succinct randomized encodings which are currently feasible only using iO and therefore this construction can not be considered to be practical yet, and the second one based on sequential squaring. With advancements in building quantum computers, it is highly desirable to design a time-lock puzzle that would even be secure against quantum adversaries. Another possible research direction is homomorphic time-lock puzzles: all known constructions are based on sequential squaring. It would be interesting to investigate the possibility of building HTLP generically from any TLP.

Bibliography

[AFP16] Hamza Abusalah, Georg Fuchsbauer, and Krzysztof Pietrzak. Offline witness encryption. In Mark Manulis, Ahmad-Reza Sadeghi, and Steve Schneider, editors, *ACNS 16: 14th International Conference on Applied Cryptography and Network Security*, volume 9696 of *Lecture Notes in Computer Science*, pages 285–303, Guildford, UK, June 19–22, 2016. Springer, Heidelberg, Germany.

[AIK06] Benny Applebaum, Yuval Ishai, and Eyal Kushilevitz. Computationally private randomizing polynomials and their applications. *Comput. Complex.*, 15(2):115–162, 2006.

[AJN+16] Prabhanjan Ananth, Aayush Jain, Moni Naor, Amit Sahai, and Eylon Yogev. Universal constructions and robust combiners for indistinguishability obfuscation and witness encryption. In Matthew Robshaw and Jonathan Katz, editors, *Advances in Cryptology – CRYPTO 2016, Part II*, volume 9815 of *Lecture Notes in Computer Science*, pages 491–520, Santa Barbara, CA, USA, August 14–18, 2016. Springer, Heidelberg, Germany.

[AK21] Aydin Abadi and Aggelos Kiayias. Multi-instance publicly verifiable time-lock puzzle and its applications. Financial Cryptography and Data Security, 2021.

[And97] Ross Anderson. Two remarks on public key cryptography. Technical Report 549, University of Cambrige Computer Laboratry, 1997. Available at `https://www.cl.cam.ac.uk/techreports/UCAM-CL-TR-549.pdf`.

[App17] Benny Applebaum. Garbled circuits as randomized encodings of functions: a primer. In *Tutorials on the Foundations of Cryptography*, pages 1–44. 2017.

[ASS+16] Nimrod Aviram, Sebastian Schinzel, Juraj Somorovsky, Nadia Heninger, Maik Dankel, Jens Steube, Luke Valenta, David Adrian, J. Alex Halderman, Viktor Dukhovni, Emilia Käsper, Shaanan Cohney, Susanne Engels, Christof Paar, and Yuval Shavitt. DROWN:

Breaking TLS using SSLv2. In Thorsten Holz and Stefan Savage, editors, *USENIX Security 2016: 25th USENIX Security Symposium*, pages 689–706, Austin, TX, USA, August 10–12, 2016. USENIX Association.

[BBS04] Dan Boneh, Xavier Boyen, and Hovav Shacham. Short group signatures. In Matthew Franklin, editor, *Advances in Cryptology – CRYPTO 2004*, volume 3152 of *Lecture Notes in Computer Science*, pages 41–55, Santa Barbara, CA, USA, August 15–19, 2004. Springer, Heidelberg, Germany.

[BCP14] Elette Boyle, Kai-Min Chung, and Rafael Pass. On extractability obfuscation. In Yehuda Lindell, editor, *TCC 2014: 11th Theory of Cryptography Conference*, volume 8349 of *Lecture Notes in Computer Science*, pages 52–73, San Diego, CA, USA, February 24–26, 2014. Springer, Heidelberg, Germany.

[BD20] Jeffrey Burdges and Luca De Feo. Delay encryption. Cryptology ePrint Archive, Report 2020/638, 2020. `https://eprint.iacr.org/2020/638`.

[BDGM19] Zvika Brakerski, Nico Döttling, Sanjam Garg, and Giulio Malavolta. Leveraging linear decryption: Rate-1 fully-homomorphic encryption and time-lock puzzles. In Dennis Hofheinz and Alon Rosen, editors, *TCC 2019: 17th Theory of Cryptography Conference, Part II*, volume 11892 of *Lecture Notes in Computer Science*, pages 407–437, Nuremberg, Germany, December 1–5, 2019. Springer, Heidelberg, Germany.

[BF01] Dan Boneh and Matthew K. Franklin. Identity-based encryption from the Weil pairing. In Joe Kilian, editor, *Advances in Cryptology – CRYPTO 2001*, volume 2139 of *Lecture Notes in Computer Science*, pages 213–229, Santa Barbara, CA, USA, August 19–23, 2001. Springer, Heidelberg, Germany.

[BGI+01] Boaz Barak, Oded Goldreich, Russell Impagliazzo, Steven Rudich, Amit Sahai, Salil P. Vadhan, and Ke Yang. On the (im)possibility of obfuscating programs. In Joe Kilian, editor, *Advances in Cryptology – CRYPTO 2001*, volume 2139 of *Lecture Notes in Computer Science*, pages 1–18, Santa Barbara, CA, USA, August 19–23, 2001. Springer, Heidelberg, Germany.

[BGI14] Elette Boyle, Shafi Goldwasser, and Ioana Ivan. Functional signatures and pseudorandom functions. In Hugo Krawczyk, editor, *PKC 2014:*

17th International Conference on Theory and Practice of Public Key Cryptography, volume 8383 of *Lecture Notes in Computer Science*, pages 501–519, Buenos Aires, Argentina, March 26–28, 2014. Springer, Heidelberg, Germany.

[BGI+17] Saikrishna Badrinarayanan, Sanjam Garg, Yuval Ishai, Amit Sahai, and Akshay Wadia. Two-message witness indistinguishability and secure computation in the plain model from new assumptions. In Tsuyoshi Takagi and Thomas Peyrin, editors, *Advances in Cryptology – ASIACRYPT 2017, Part III*, volume 10626 of *Lecture Notes in Computer Science*, pages 275–303, Hong Kong, China, December 3–7, 2017. Springer, Heidelberg, Germany.

[BGJ+15] Nir Bitansky, Shafi Goldwasser, Abhishek Jain, Omer Paneth, Vinod Vaikuntanathan, and Brent Waters. Time-lock puzzles from randomized encodings. Cryptology ePrint Archive, Report 2015/514, 2015. `http://eprint.iacr.org/2015/514`.

[BGJ+16] Nir Bitansky, Shafi Goldwasser, Abhishek Jain, Omer Paneth, Vinod Vaikuntanathan, and Brent Waters. Time-lock puzzles from randomized encodings. In Madhu Sudan, editor, *ITCS 2016: 7th Conference on Innovations in Theoretical Computer Science*, pages 345–356, Cambridge, MA, USA, January 14–16, 2016. Association for Computing Machinery.

[BGV12] Zvika Brakerski, Craig Gentry, and Vinod Vaikuntanathan. (Leveled) fully homomorphic encryption without bootstrapping. In Shafi Goldwasser, editor, *ITCS 2012: 3rd Innovations in Theoretical Computer Science*, pages 309–325, Cambridge, MA, USA, January 8–10, 2012. Association for Computing Machinery.

[BH15] Mihir Bellare and Viet Tung Hoang. Adaptive witness encryption and asymmetric password-based cryptography. In Jonathan Katz, editor, *PKC 2015: 18th International Conference on Theory and Practice of Public Key Cryptography*, volume 9020 of *Lecture Notes in Computer Science*, pages 308–331, Gaithersburg, MD, USA, March 30 – April 1, 2015. Springer, Heidelberg, Germany.

[BIOW20] Ohad Barta, Yuval Ishai, Rafail Ostrovsky, and David J. Wu. On succinct arguments and witness encryption from groups. In Daniele Micciancio and Thomas Ristenpart, editors, *Advances in Cryptology*

– *CRYPTO 2020, Part I*, volume 12170 of *Lecture Notes in Computer Science*, pages 776–806, Santa Barbara, CA, USA, August 17–21, 2020. Springer, Heidelberg, Germany.

[BISW18] Dan Boneh, Yuval Ishai, Amit Sahai, and David J. Wu. Quasi-optimal SNARGs via linear multi-prover interactive proofs. In Jesper Buus Nielsen and Vincent Rijmen, editors, *Advances in Cryptology – EUROCRYPT 2018, Part III*, volume 10822 of *Lecture Notes in Computer Science*, pages 222–255, Tel Aviv, Israel, April 29 – May 3, 2018. Springer, Heidelberg, Germany.

[BL18] Fabrice Benhamouda and Huijia Lin. k-round multiparty computation from k-round oblivious transfer via garbled interactive circuits. In Jesper Buus Nielsen and Vincent Rijmen, editors, *Advances in Cryptology – EUROCRYPT 2018, Part II*, volume 10821 of *Lecture Notes in Computer Science*, pages 500–532, Tel Aviv, Israel, April 29 – May 3, 2018. Springer, Heidelberg, Germany.

[BMSZ16] Saikrishna Badrinarayanan, Eric Miles, Amit Sahai, and Mark Zhandry. Post-zeroizing obfuscation: New mathematical tools, and the case of evasive circuits. In Marc Fischlin and Jean-Sébastien Coron, editors, *Advances in Cryptology – EUROCRYPT 2016, Part II*, volume 9666 of *Lecture Notes in Computer Science*, pages 764–791, Vienna, Austria, May 8–12, 2016. Springer, Heidelberg, Germany.

[BN00] Dan Boneh and Moni Naor. Timed commitments. In Mihir Bellare, editor, *Advances in Cryptology – CRYPTO 2000*, volume 1880 of *Lecture Notes in Computer Science*, pages 236–254, Santa Barbara, CA, USA, August 20–24, 2000. Springer, Heidelberg, Germany.

[BP15] Elette Boyle and Rafael Pass. Limits of extractability assumptions with distributional auxiliary input. In Tetsu Iwata and Jung Hee Cheon, editors, *Advances in Cryptology – ASIACRYPT 2015, Part II*, volume 9453 of *Lecture Notes in Computer Science*, pages 236–261, Auckland, New Zealand, November 30 – December 3, 2015. Springer, Heidelberg, Germany.

[BR93] Mihir Bellare and Phillip Rogaway. Random oracles are practical: A paradigm for designing efficient protocols. In Dorothy E. Denning, Raymond Pyle, Ravi Ganesan, Ravi S. Sandhu, and Victoria Ashby,

editors, *ACM CCS 93: 1st Conference on Computer and Communications Security*, pages 62–73, Fairfax, Virginia, USA, November 3–5, 1993. ACM Press.

[BS02] Dan Boneh and Alice Silverberg. Applications of multilinear forms to cryptography. Cryptology ePrint Archive, Report 2002/080, 2002. `http://eprint.iacr.org/2002/080`.

[BS03] Dan Boneh and Alice Silverberg. Applications of multilinear forms to cryptography. In *Topics in algebraic and noncommutative geometry (Luminy/Annapolis, MD, 2001)*, volume 324 of *Contemp. Math.*, pages 71–90. Amer. Math. Soc., Providence, RI, 2003.

[BSW11] Dan Boneh, Amit Sahai, and Brent Waters. Functional encryption: Definitions and challenges. In Yuval Ishai, editor, *TCC 2011: 8th Theory of Cryptography Conference*, volume 6597 of *Lecture Notes in Computer Science*, pages 253–273, Providence, RI, USA, March 28–30, 2011. Springer, Heidelberg, Germany.

[BSY17] Hanno Böck, Juraj Somorovsky, and Craig Young. Return of Bleichenbacher's oracle threat (ROBOT). Cryptology ePrint Archive, Report 2017/1189, 2017. `https://eprint.iacr.org/2017/1189`.

[CDG+17] Chongwon Cho, Nico Döttling, Sanjam Garg, Divya Gupta, Peihan Miao, and Antigoni Polychroniadou. Laconic oblivious transfer and its applications. In Jonathan Katz and Hovav Shacham, editors, *Advances in Cryptology – CRYPTO 2017, Part II*, volume 10402 of *Lecture Notes in Computer Science*, pages 33–65, Santa Barbara, CA, USA, August 20–24, 2017. Springer, Heidelberg, Germany.

[CHK04] Ran Canetti, Shai Halevi, and Jonathan Katz. Chosen-ciphertext security from identity-based encryption. In Christian Cachin and Jan Camenisch, editors, *Advances in Cryptology – EUROCRYPT 2004*, volume 3027 of *Lecture Notes in Computer Science*, pages 207–222, Interlaken, Switzerland, May 2–6, 2004. Springer, Heidelberg, Germany.

[CHKO06] Jung Hee Cheon, Nicholas Hopper, Yongdae Kim, and Ivan Osipkov. Timed-release and key-insulated public key encryption. In Giovanni Di Crescenzo and Avi Rubin, editors, *FC 2006: 10th International Conference on Financial Cryptography and Data Security*, volume 4107 of *Lecture Notes in Computer Science*, pages 191–205,

Anguilla, British West Indies, February 27 – March 2, 2006. Springer, Heidelberg, Germany.

[CHS07] Konstantinos Chalkias, Dimitrios Hristu-Varsakelis, and George Stephanides. Improved anonymous timed-release encryption. In Joachim Biskup and Javier López, editors, *ESORICS 2007: 12th European Symposium on Research in Computer Security*, volume 4734 of *Lecture Notes in Computer Science*, pages 311–326, Dresden, Germany, September 24–26, 2007. Springer, Heidelberg, Germany.

[CJK20] Peter Chvojka, Tibor Jager, and Saqib A. Kakvi. Offline witness encryption with semi-adaptive security. In Mauro Conti, Jianying Zhou, Emiliano Casalicchio, and Angelo Spognardi, editors, *ACNS 20: 18th International Conference on Applied Cryptography and Network Security, Part I*, volume 12146 of *Lecture Notes in Computer Science*, pages 231–250, Rome, Italy, October 19–22, 2020. Springer, Heidelberg, Germany.

[CJSS20] Peter Chvojka, Tibor Jager, Daniel Slamanig, and Christoph Striecks. Generic constructions of incremental and homomorphic timed-release encryption. Cryptology ePrint Archive, Report 2020/739, 2020. `https://eprint.iacr.org/2020/739`.

[CLQ05] Julien Cathalo, Benoît Libert, and Jean-Jacques Quisquater. Efficient and non-interactive timed-release encryption. In Sihan Qing, Wenbo Mao, Javier López, and Guilin Wang, editors, *ICICS 05: 7th International Conference on Information and Communication Security*, volume 3783 of *Lecture Notes in Computer Science*, pages 291–303, Beijing, China, December 10–13, 2005. Springer, Heidelberg, Germany.

[CY08] Sherman S. M. Chow and Siu-Ming Yiu. Timed-release encryption revisited. In Joonsang Baek, Feng Bao, Kefei Chen, and Xuejia Lai, editors, *ProvSec 2008: 2nd International Conference on Provable Security*, volume 5324 of *Lecture Notes in Computer Science*, pages 38–51, Shanghai, China, October 31 – November 1, 2008. Springer, Heidelberg, Germany.

[DGJ+18] David Derler, Kai Gellert, Tibor Jager, Daniel Slamanig, and Christoph Striecks. Bloom filter encryption and applications to efficient forward-secret 0-RTT key exchange. Cryptology ePrint Archive, Report 2018/199, 2018. `https://eprint.iacr.org/2018/199`.

[DKP20] Dana Dachman-Soled, Ilan Komargodski, and Rafael Pass. Non-malleable codes for bounded polynomial depth tampering. Cryptology ePrint Archive, Report 2020/776, 2020. https://eprint.iacr.org/2020/776.

[DN00] Cynthia Dwork and Moni Naor. Zaps and their applications. In *41st Annual Symposium on Foundations of Computer Science*, pages 283–293, Redondo Beach, CA, USA, November 12–14, 2000. IEEE Computer Society Press.

[DOR99] Giovanni Di Crescenzo, Rafail Ostrovsky, and Sivaramakrishnan Rajagopalan. Conditional oblivious transfer and timed-release encryption. In Jacques Stern, editor, *Advances in Cryptology – EUROCRYPT'99*, volume 1592 of *Lecture Notes in Computer Science*, pages 74–89, Prague, Czech Republic, May 2–6, 1999. Springer, Heidelberg, Germany.

[EFKP20] Naomi Ephraim, Cody Freitag, Ilan Komargodski, and Rafael Pass. Non-malleable time-lock puzzles and applications. Cryptology ePrint Archive, Report 2020/779, 2020. https://eprint.iacr.org/2020/779.

[ElG84] Taher ElGamal. A public key cryptosystem and a signature scheme based on discrete logarithms. In G. R. Blakley and David Chaum, editors, *Advances in Cryptology – CRYPTO'84*, volume 196 of *Lecture Notes in Computer Science*, pages 10–18, Santa Barbara, CA, USA, August 19–23, 1984. Springer, Heidelberg, Germany.

[FHHL18] Pooya Farshim, Julia Hesse, Dennis Hofheinz, and Enrique Larraia. Graded encoding schemes from obfuscation. In Michel Abdalla and Ricardo Dahab, editors, *PKC 2018: 21st International Conference on Theory and Practice of Public Key Cryptography, Part II*, volume 10770 of *Lecture Notes in Computer Science*, pages 371–400, Rio de Janeiro, Brazil, March 25–29, 2018. Springer, Heidelberg, Germany.

[FKL18] Georg Fuchsbauer, Eike Kiltz, and Julian Loss. The algebraic group model and its applications. In Hovav Shacham and Alexandra Boldyreva, editors, *Advances in Cryptology – CRYPTO 2018, Part II*, volume 10992 of *Lecture Notes in Computer Science*, pages 33–62, Santa Barbara, CA, USA, August 19–23, 2018. Springer, Heidelberg, Germany.

[FNV17] Antonio Faonio, Jesper Buus Nielsen, and Daniele Venturi. Predictable arguments of knowledge. In Serge Fehr, editor, *PKC 2017: 20th International Conference on Theory and Practice of Public Key Cryptography, Part I*, volume 10174 of *Lecture Notes in Computer Science*, pages 121–150, Amsterdam, The Netherlands, March 28–31, 2017. Springer, Heidelberg, Germany.

[GGH12] Sanjam Garg, Craig Gentry, and Shai Halevi. Candidate multilinear maps from ideal lattices. Cryptology ePrint Archive, Report 2012/610, 2012. `http://eprint.iacr.org/2012/610`.

[GGH13] Sanjam Garg, Craig Gentry, and Shai Halevi. Candidate multilinear maps from ideal lattices. In Thomas Johansson and Phong Q. Nguyen, editors, *Advances in Cryptology – EUROCRYPT 2013*, volume 7881 of *Lecture Notes in Computer Science*, pages 1–17, Athens, Greece, May 26–30, 2013. Springer, Heidelberg, Germany.

[GGHW14] Sanjam Garg, Craig Gentry, Shai Halevi, and Daniel Wichs. On the implausibility of differing-inputs obfuscation and extractable witness encryption with auxiliary input. In Juan A. Garay and Rosario Gennaro, editors, *Advances in Cryptology – CRYPTO 2014, Part I*, volume 8616 of *Lecture Notes in Computer Science*, pages 518–535, Santa Barbara, CA, USA, August 17–21, 2014. Springer, Heidelberg, Germany.

[GGSW13] Sanjam Garg, Craig Gentry, Amit Sahai, and Brent Waters. Witness encryption and its applications. In Dan Boneh, Tim Roughgarden, and Joan Feigenbaum, editors, *45th Annual ACM Symposium on Theory of Computing*, pages 467–476, Palo Alto, CA, USA, June 1–4, 2013. ACM Press.

[GHJL17] Felix Günther, Britta Hale, Tibor Jager, and Sebastian Lauer. 0-RTT key exchange with full forward secrecy. In Jean-Sébastien Coron and Jesper Buus Nielsen, editors, *Advances in Cryptology – EUROCRYPT 2017, Part III*, volume 10212 of *Lecture Notes in Computer Science*, pages 519–548, Paris, France, April 30 – May 4, 2017. Springer, Heidelberg, Germany.

[GKP+13a] Shafi Goldwasser, Yael Tauman Kalai, Raluca A. Popa, Vinod Vaikuntanathan, and Nickolai Zeldovich. How to run turing machines on encrypted data. In Ran Canetti and Juan A. Garay, editors, *Advances in Cryptology – CRYPTO 2013, Part II*, volume 8043 of *Lec-*

ture Notes in Computer Science, pages 536–553, Santa Barbara, CA, USA, August 18–22, 2013. Springer, Heidelberg, Germany.

[GKP+13b] Shafi Goldwasser, Yael Tauman Kalai, Raluca A. Popa, Vinod Vaikuntanathan, and Nickolai Zeldovich. Reusable garbled circuits and succinct functional encryption. In Dan Boneh, Tim Roughgarden, and Joan Feigenbaum, editors, *45th Annual ACM Symposium on Theory of Computing*, pages 555–564, Palo Alto, CA, USA, June 1–4, 2013. ACM Press.

[GLS15] S. Dov Gordon, Feng-Hao Liu, and Elaine Shi. Constant-round MPC with fairness and guarantee of output delivery. In Rosario Gennaro and Matthew J. B. Robshaw, editors, *Advances in Cryptology – CRYPTO 2015, Part II*, volume 9216 of *Lecture Notes in Computer Science*, pages 63–82, Santa Barbara, CA, USA, August 16–20, 2015. Springer, Heidelberg, Germany.

[GM15] Matthew D. Green and Ian Miers. Forward secure asynchronous messaging from puncturable encryption. In *2015 IEEE Symposium on Security and Privacy*, pages 305–320, San Jose, CA, USA, May 17–21, 2015. IEEE Computer Society Press.

[GS08] Jens Groth and Amit Sahai. Efficient non-interactive proof systems for bilinear groups. In Nigel P. Smart, editor, *Advances in Cryptology – EUROCRYPT 2008*, volume 4965 of *Lecture Notes in Computer Science*, pages 415–432, Istanbul, Turkey, April 13–17, 2008. Springer, Heidelberg, Germany.

[IK00] Yuval Ishai and Eyal Kushilevitz. Randomizing polynomials: A new representation with applications to round-efficient secure computation. In *41st Annual Symposium on Foundations of Computer Science*, pages 294–304, Redondo Beach, CA, USA, November 12–14, 2000. IEEE Computer Society Press.

[Kah67] David Kahn. *The Code-Breakers: The Story of Secret Writing*. The Macmillan Company, 1967.

[Kil06] Eike Kiltz. Chosen-ciphertext security from tag-based encryption. In Shai Halevi and Tal Rabin, editors, *TCC 2006: 3rd Theory of Cryptography Conference*, volume 3876 of *Lecture Notes in Computer Science*, pages 581–600, New York, NY, USA, March 4–7, 2006. Springer, Heidelberg, Germany.

[KL14] Jonathan Katz and Yehuda Lindell. *Introduction to Modern Cryptography, Second Edition*. Chapman and Hall/CRC Press, 2014.

[KLW15] Venkata Koppula, Allison Bishop Lewko, and Brent Waters. Indistinguishability obfuscation for turing machines with unbounded memory. In Rocco A. Servedio and Ronitt Rubinfeld, editors, *47th Annual ACM Symposium on Theory of Computing*, pages 419–428, Portland, OR, USA, June 14–17, 2015. ACM Press.

[KLX20] Jonathan Katz, Julian Loss, and Jiayu Xu. On the security of time-lock puzzles and timed commitments. In Rafael Pass and Krzysztof Pietrzak, editors, *TCC 2020: 18th Theory of Cryptography Conference, Part III*, volume 12552 of *Lecture Notes in Computer Science*, pages 390–413, Durham, NC, USA, November 16–19, 2020. Springer, Heidelberg, Germany.

[KM15] Neal Koblitz and Alfred Menezes. The random oracle model: A twenty-year retrospective. Cryptology ePrint Archive, Report 2015/140, 2015. `http://eprint.iacr.org/2015/140`.

[KNY14] Ilan Komargodski, Moni Naor, and Eylon Yogev. Secret-sharing for NP. In Palash Sarkar and Tetsu Iwata, editors, *Advances in Cryptology – ASIACRYPT 2014, Part II*, volume 8874 of *Lecture Notes in Computer Science*, pages 254–273, Kaoshiung, Taiwan, R.O.C., December 7–11, 2014. Springer, Heidelberg, Germany.

[LJKW18] Jia Liu, Tibor Jager, Saqib A. Kakvi, and Bogdan Warinschi. How to build time-lock encryption. *Des. Codes Cryptography*, 86(11):2549–2586, 2018.

[LPS17] Huijia Lin, Rafael Pass, and Pratik Soni. Two-round and non-interactive concurrent non-malleable commitments from time-lock puzzles. In Chris Umans, editor, *58th Annual Symposium on Foundations of Computer Science*, pages 576–587, Berkeley, CA, USA, October 15–17, 2017. IEEE Computer Society Press.

[LTV12] Adriana López-Alt, Eran Tromer, and Vinod Vaikuntanathan. On-the-fly multiparty computation on the cloud via multikey fully homomorphic encryption. In Howard J. Karloff and Toniann Pitassi, editors, *44th Annual ACM Symposium on Theory of Computing*, pages 1219–1234, New York, NY, USA, May 19–22, 2012. ACM Press.

[May93] Timothy C. May. Timed-release crypto. Technical report, February 1993.

[MMV11] Mohammad Mahmoody, Tal Moran, and Salil P. Vadhan. Time-lock puzzles in the random oracle model. In Phillip Rogaway, editor, *Advances in Cryptology – CRYPTO 2011*, volume 6841 of *Lecture Notes in Computer Science*, pages 39–50, Santa Barbara, CA, USA, August 14–18, 2011. Springer, Heidelberg, Germany.

[MRY04] Philip D. MacKenzie, Michael K. Reiter, and Ke Yang. Alternatives to non-malleability: Definitions, constructions, and applications (extended abstract). In Moni Naor, editor, *TCC 2004: 1st Theory of Cryptography Conference*, volume 2951 of *Lecture Notes in Computer Science*, pages 171–190, Cambridge, MA, USA, February 19–21, 2004. Springer, Heidelberg, Germany.

[MT19] Giulio Malavolta and Sri Aravinda Krishnan Thyagarajan. Homomorphic time-lock puzzles and applications. In Alexandra Boldyreva and Daniele Micciancio, editors, *Advances in Cryptology – CRYPTO 2019, Part I*, volume 11692 of *Lecture Notes in Computer Science*, pages 620–649, Santa Barbara, CA, USA, August 18–22, 2019. Springer, Heidelberg, Germany.

[NY90] Moni Naor and Moti Yung. Public-key cryptosystems provably secure against chosen ciphertext attacks. In *22nd Annual ACM Symposium on Theory of Computing*, pages 427–437, Baltimore, MD, USA, May 14–16, 1990. ACM Press.

[O'N10] Adam O'Neill. Definitional issues in functional encryption. Cryptology ePrint Archive, Report 2010/556, 2010. `http://eprint.iacr.org/2010/556`.

[OP01] Tatsuaki Okamoto and David Pointcheval. The gap-problems: A new class of problems for the security of cryptographic schemes. In Kwangjo Kim, editor, *PKC 2001: 4th International Workshop on Theory and Practice in Public Key Cryptography*, volume 1992 of *Lecture Notes in Computer Science*, pages 104–118, Cheju Island, South Korea, February 13–15, 2001. Springer, Heidelberg, Germany.

[Pai99] Pascal Paillier. Public-key cryptosystems based on composite degree residuosity classes. In Jacques Stern, editor, *Advances in Cryptology – EUROCRYPT'99*, volume 1592 of *Lecture Notes in Computer Science*, pages 223–238, Prague, Czech Republic, May 2–6, 1999. Springer, Heidelberg, Germany.

[PD19] Tapas Pal and Ratna Dutta. Offline witness encryption from witness PRF and randomized encoding in CRS model. In Julian Jang-Jaccard and Fuchun Guo, editors, *ACISP 19: 24th Australasian Conference on Information Security and Privacy*, volume 11547 of *Lecture Notes in Computer Science*, pages 78–96, Christchurch, New Zealand, July 3–5, 2019. Springer, Heidelberg, Germany.

[PD20] Tapas Pal and Ratna Dutta. Semi-adaptively secure offline witness encryption from puncturable witness PRF. In Khoa Nguyen, Wenling Wu, Kwok-Yan Lam, and Huaxiong Wang, editors, *ProvSec 2020: 14th International Conference on Provable Security*, volume 12505 of *Lecture Notes in Computer Science*, pages 169–189, Singapore, November 29 – December 1, 2020. Springer, Heidelberg, Germany.

[PDM+18] Damian Poddebniak, Christian Dresen, Jens Müller, Fabian Ising, Sebastian Schinzel, Simon Friedberger, Juraj Somorovsky, and Jörg Schwenk. Efail: Breaking S/MIME and OpenPGP email encryption using exfiltration channels. In William Enck and Adrienne Porter Felt, editors, *USENIX Security 2018: 27th USENIX Security Symposium*, pages 549–566, Baltimore, MD, USA, August 15–17, 2018. USENIX Association.

[RSW96] Ronald L. Rivest, Adi Shamir, and David A. Wagner. Time-lock puzzles and timed-release crypto. Technical report, 1996.

[Sha79] Adi Shamir. How to share a secret. *Commun. ACM*, 22(11):612–613, 1979.

[Sho97] Victor Shoup. Lower bounds for discrete logarithms and related problems. In Walter Fumy, editor, *Advances in Cryptology – EUROCRYPT'97*, volume 1233 of *Lecture Notes in Computer Science*, pages 256–266, Konstanz, Germany, May 11–15, 1997. Springer, Heidelberg, Germany.

[Sho04] Victor Shoup. Sequences of games: a tool for taming complexity in security proofs. Cryptology ePrint Archive, Report 2004/332, 2004. `http://eprint.iacr.org/2004/332`.

[TBM+20] Sri Aravinda Krishnan Thyagarajan, Adithya Bhat, Giulio Malavolta, Nico Döttling, Aniket Kate, and Dominique Schröder. Verifiable timed signatures made practical. In *to appear at ACM CCS 2020, https://verifiable-timed-signatures.github.io/web/assets/paper.pdf*, 2020.

[Unr14] Dominique Unruh. Revocable quantum timed-release encryption. In Phong Q. Nguyen and Elisabeth Oswald, editors, *Advances in Cryptology – EUROCRYPT 2014*, volume 8441 of *Lecture Notes in Computer Science*, pages 129–146, Copenhagen, Denmark, May 11–15, 2014. Springer, Heidelberg, Germany.

[Zha16] Mark Zhandry. How to avoid obfuscation using witness PRFs. In Eyal Kushilevitz and Tal Malkin, editors, *TCC 2016-A: 13th Theory of Cryptography Conference, Part II*, volume 9563 of *Lecture Notes in Computer Science*, pages 421–448, Tel Aviv, Israel, January 10–13, 2016. Springer, Heidelberg, Germany.