Versatile and Sustainable Timed-Release Encryption and Sequential Time-Lock Puzzles (Extended Abstract)

Peter Chvojka[1(✉)], Tibor Jager[1], Daniel Slamanig[2], and Christoph Striecks[2]

[1] University of Wuppertal, Wuppertal, Germany
{chvojka,tibor.jager}@uni-wuppertal.de
[2] AIT Austrian Institute of Technology, Vienna, Austria
{daniel.slamanig,christoph.striecks}@ait.ac.at

Abstract. Timed-release encryption (TRE) makes it possible to send messages "into the future" such that a pre-determined amount of time needs to pass before a message can be accessed. Malavolta and Thyagarajan (CRYPTO'19) recently introduced an interesting variant of TRE called homomorphic time-lock puzzles (HTLPs), making TRE more versatile and greatly extending its applications. Here one considers multiple independently generated puzzles and the homomorphic evaluation of a circuit over these puzzles. Solving the so obtained puzzle yields the output of a circuit evaluated on the messages locked by the original puzzles.

We observe that viewing HTLPs more abstractly gives rise to a simple generic construction of homomorphic TRE (HTRE) that is not necessarily based on sequential squaring, but can be instantiated based on any TLP, e.g., from the LWE assumption (via randomized encodings). This construction has slightly different properties, but provides essentially the same functionality for applications. It makes TRE versatile and can be used beyond HTRE, for instance to construct timed-release functional encryption. Interestingly, it achieves a new *"solve one, get many for free"* property, which supports that an arbitrary number of independently time-locked (homomorphically evaluated) messages can all be obtained simultaneously after solving only a single puzzle. This puzzle is independent of the number of time-locked messages and thus achieves optimal amortized cost.

Moreover, we define and construct *sequential* TLPs as a particularly useful generalization of TLPs and TRE. Such puzzles can be solved sequentially in a way that solving a puzzle additionally considers the previous solution and the time required to solve the puzzle is determined

This work was supported by the German Federal Ministry of Education and Research (BMBF) project REZEIVER, the European Research Council (ERC) under the European Union's Horizon 2020 research and innovation programme, grant agreement n°802823, the EU's Horizon 2020 ECSEL Joint Undertaking under grant agreement n°783119 (SECREDAS) and by the Austrian Science Fund (FWF) and netidee SCIENCE under grant agreement P31621-N38 (PROFET). This is an extended abstract, the full version of this paper is available at [11].

E. Bertino et al. (Eds.): ESORICS 2021, LNCS 12973, pp. 64–85, 2021.
https://doi.org/10.1007/978-3-030-88428-4_4

by the difference in the time parameters. When instantiated from sequential squaring, this allows to realize public "sequential squaring services", where everyone can time-lock messages, but only one entity needs to perform the computations required to solve puzzles. Thus, this removes the burden of wasting computational resources by every receiver and makes TRE economically and ecologically more sustainable.

1 Introduction

Timed-release encryption (TRE) has the goal of sending information into the future in a way that the sender can be sure that a pre-determined amount of time needs to pass before the information can be decrypted. This idea was firstly discussed by May [25], who introduced this notion and proposed a solution based on trusted agents, which after the pre-determined time has passed releases some secret which allows to obtain the information (see also [10,12]). In this work we will focus on an alternative idea proposed by Rivest *et al.* in [29] and called time-lock puzzles (TLPs), which does not require trusted agents. TLPs allow to seal messages in such a way that one is able to obtain the sealed message only by executing an expensive *sequential* computation. The amount of time required to perform this sequential computation is determined by a hardness parameter T of the TLP, which can be freely chosen. Here, a sender can just publish a puzzle whose solution hides the message until enough time has elapsed for the puzzle to be solved. TLPs have found numerous applications such as sealed-bid auctions [29], fair contract signing [6], zero-knowledge arguments [13], or non-malleable commitments [20].

TLP Constructions. The TLP proposed by Rivest *et al.* in [29] uses sequential squaring in an RSA group $\mathbb{Z}_N^*$, i.e., for hardness T compute $s = x^{2^T}$. An interesting feature of this TLP construction is that creating a puzzle, i.e., knowing the factorization of N, is much faster than the expensive sequential computation to solve the puzzle. This is an important property of TLPs when the required amount of time until the puzzle should be solved is very large. Interestingly, TLPs with this property seem hard to find. In [23] Mahmoody *et al.* show that in the random-oracle model it is impossible to construct TLPs from one-way permutations and collision-resistant hash-functions that require more parallel time to solve than the total work required to generate a puzzle and thus ruling out black-box constructions of such TLPs. On the positive side, Bitansky *et al.* [4] show how to construct TLPs with the aforementioned property from randomized encodings [2,18] relying on indistinguishability obfuscation. Interestingly, when slightly relaxing the requirements and allowing efficient parallel computation in the generation of the puzzles or a solution independent preprocessing (so-called *weak* TLPs), then such TLPs can be constructed generically from one-way functions and directly from the learning with errors (LWE) assumption, respectively, via randomized encodings.

Homomorphic TLPs. Recently, Malavolta and Thyagarajan [24] (MT19 henceforth) proposed an interesting variant called *homomorphic* TLPs (HTLPs).

Here one considers multiple puzzles $(Z_1, \ldots, Z_n)$ with hardness parameter T, which can be independently generated by different entities. Without knowing the corresponding solutions $(s_1, \ldots, s_n)$ one can homomorphically evaluate a circuit C over these puzzles to obtain as result a puzzle $\widehat{Z}$ with solution $C(s_1, \ldots, s_n)$, where the hardness of this resulting puzzle does not depend on the size of the circuit C that was evaluated (which is called compactness). Consequently, this allows to aggregate a potentially large number of puzzles in a way that only a single puzzle needs to be solved. While this concept is interesting on its own, MT19 also shows that it extends the applications of TLPs and in particular present applications to e-voting, multi-party coin flipping as well as multi-party contract signing, or more recently verifiable timed signatures [31], again yielding a number of interesting further applications.

MT19 conjecture that any application that involves a large number of users and thus the constraint of requiring to solve multiple puzzles (in parallel) constitute one of the main obstacles that so far prevented the large scale adoption of TLPs. As already, mentioned, this can be partly mitigated via HTLPs by MT19 and in particular if one is only interested in homomorphic evaluations over multiple messages. We additionally stress that applications requiring to solve multiple puzzles will also represent a huge waste of resources and are thus problematic from an economic and ecological perspective. Moreover, even the requirement for a receiver to only solve a single puzzle on its own may already prevent the application of TRE, e.g., for resource constrained receivers of messages as omnipresent within the Internet of Things (IoT).

Motivation for Our Work. The motivation of our work is twofold. Firstly, our goal is to make TRE more versatile in order to improve on existing applications and to broaden the scope of applications even further. For instance, when we look at the HTLPs in MT19, they construct a linearly homomorphic TLP (LHTLP) from the sequential squaring TLP and Paillier encryption [28] which is linearly homomorphic and the evaluation is independent of the hardness T (and one can also turn this into a multiplicatively homomorphic TLP). In order to extend this to fully HTLPs (FHTLPs), MT19 requires sub-exponentially hard indistinguishability obfuscation, whereas in follow-up work Brakerski *et al.* in [8] proposed FHTLPs from standard assumption which itself requires an LHTLP (where to the best of our knowledge the aforementioned is the only known construction). In particular, the idea in [8] is to use a multi-key fully-homomorphic encryption (MK-FHE) scheme [22] to encrypt every message with a fresh key and an LHTP to lock the respective MK-FHE secret keys. To be compatible with the LHTLP, Brakerski *et al.* in particular require a MK-FHE scheme with a linear decryption algorithm.

Unfortunately, all these constructions are not generic as they rely on a single particular construction of an HTLP from sequential squaring (and additionally a very specific MK-FHE scheme in [8]). Moreover, for every such puzzle including the one obtained from homomorphically evaluating on many such puzzles, one can only start to solve it after having produced it. Consequently, although the homomorphic property makes it scalable in a setting where one is only inter-

ested in the homomorphic evaluation over all encrypted messages, it would be convenient to have an approach that also supports a *"solve one, get many for free"* property. And this even if one wants to obtain *all* encrypted messages in full, instead of only the result of a homomorphic evaluation. Note that, if in contrast to the homomorphically evaluated function over all the messages, one wants to unlock n of the input messages with the approaches in [8,24], it requires to solve n puzzles. Consequently, we ask whether it is possible to come up with an approach that provides the *"solve one, get many for free"* property on an arbitrary number of independently time-locked messages such that it is possible to decrypt all single messages and at the same time. So essentially having a solution that *can be* homomorphic but does not need to be if one is only interested in the single messages and all with only solving a single puzzle. Ideally this approach is generic in nature and thus would allow to construct (homomorphic) timed-release encryption (TRE) generically from *any* TLP.

Secondly, a central drawback of TRE is that it puts considerable computational overhead on the message receiver, i.e., the receiver has to invest lots of computational resources to solve the puzzle to obtain the time-locked message. This makes it undesirable for real-life scenarios from an economic as well as ecological perspective. While HTLPs of MT19 address this problem from a different angle, i.e., homomorphically combine many TLPs such that only one puzzle needs to be solved, this will not reveal the individual messages without solving all puzzles. And while this functionality is a helpful feature in certain applications, it can not be considered a general purpose solution, because the amount of recoverable data is bounded by the amount of data that can be encapsulated in a single ciphertext. Moreover, it still requires the receiver to waste potentially significant resources for solving a new puzzle. Consequently, we ask whether this can be avoided.

Due to the lack of space we defer a discussion on recent concurrent and independent work on TLPs to Appendix A.

2 Technical Overview and Contributions

Before we discuss our contributions we stress, that the terms time-lock puzzle (TLP) [29], timed-release encryption (TRE) [25], and time-lock encryption (TLE) [21] are often used interchangeably in the literature. For our constructions we need to distinguish between them. In particular, TRE will denote an encryption scheme which allows us send messages "into the future". A TLP provides the core functionality of a puzzle that needs a certain amount of time to be solved, without considering any messages. TLPs will be used as a building block for TRE. Now, we are ready to discuss our contributions.

Versatile Timed-Release Encryption. We introduce a generic approach to construct TRE. The basic and indeed very simple idea is that given *any* TLP we can use it to generate a puzzle Z and its solution s, and we can use s as the random coins for the key generation algorithm $\mathsf{Gen}(1^\lambda; s)$ of a public key encryption (PKE) scheme. Then, we provide the respective public key pk as

parameters of the TRE and solving the puzzle Z reveals s and thus sk allowing to decrypt *all* of the ciphertexts computed with respect to pk. Note that when using s as the random coins for a partially homomorphic encryption scheme, e.g., ElGamal [14], or a fully homomorphic encryption scheme, e.g., BGV [9], this immediately yields (fully) homomorphic TRE. Interestingly, this approach then allows us 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. Consequently, solving one puzzle allows to decrypt all ciphertexts associated to a hardness parameter (generated by many potentially independent entities). We note that our approach to HTRE satisfies the basic definition of HTLPs from MT19, where the time required to solve the puzzles starts with the generation of the parameters. In contrast, MT19 also provides the notion of a reusable setup for their LHTLP, where one can use the same parameters to generate 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 [8]). However, we observe that for all the applications discussed in [24] it seems sufficient, and in some applications even more desirable, when the runtime of the puzzle is counted from the point of running the puzzle setup algorithm. For instance, MT19 discuss an application to e-voting, where it rather seems to complicate issues when one can only start solving the puzzle after the last voter cast its vote. It seems more practical to set-up the puzzle such that the solution can be made available at a certain predefined point in time. And even if this is not required, it might be easy to adjust the setup in a way that it outputs a set of public parameters, and a user can choose which public parameters to use when computing a puzzle. We defer a detailed discussion of the applications to Appendix B.

Moreover, we demonstrate that our TRE framework can be used to obtain other variants of TRE in a generic way. We showcase this using the regime of functional encryption. In particular, we introduce timed-release functional encryption (TRFE) which allows to time-lock a function f. After a certain time has passed everyone can learn the function $f(x)$ of any ever encrypted message x. As an application we discuss identity-based encryption (IBE) [5] with locked keys, where the key generator at registration gives locked IBE secret keys for various validity periods (e.g., each for a month) to the user and the respective secret keys then unlock over time.

Sustainable Timed-Release Encryption. We introduce the notion of sequential TLPs as a particularly useful generalization of TLPs, which yields practical and particularly sustainable TRE schemes (from the perspective of consumption of computational resources). The basic idea is that the puzzle generation takes a sequence of hardness parameters $T_1, \ldots, T_n$ (where we assume that $T_i < T_{i+1}$ for all $i \in [n-1]$) and outputs a sequence of puzzles and solutions $(Z_i, s_i)_{i \in [n]}$. Now the distinguishing feature is that puzzles can be solved sequentially in a way that solving Z_i additionally considers solution s_{i-1} and the time required to solve puzzle Z_i is determined by the hardness $T_i - T_{i-1}$ (note that having $n = 1$ this yields a conventional TLP). From this, we then build a sequential TRE scheme,

where security is based on the security of a sequential TLP. Unfortunately, it turns out that such a construction is non-trivial. For a TRE scheme to be secure, we require that any adversary that runs in time $T < T_i$ is not able to break the security of an encryption with respect to time slot T_i. For such an adversary, we need to simulate all values up to T_{i-1}, in particular all TLP solutions $s_1, \ldots, s_{i-1}$ up to T_{i-1}, properly, as otherwise the reduction would not simulate the security experiment properly for an adversary running in time $T = T_{i-1} < T_i$, for instance. However, it is not possible to build a reduction which receives as input $s_1, \ldots, s_{i-1}$ as part of the TLP instance, because then the reduction would only be able to break the assumption that the puzzle is hard if it runs in time less than $T_i - T_{i-1}$. Our solution to overcome this difficulty is to construct a TRE scheme which does not directly use the real solutions s_i, but instead $\mathsf{F}(T_i, s_i)$, 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", and thus is able to provide a proper simulation. At the same time it is reasonable to assume that no adversary 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 note that MT19 [24, Section 5.2] also proposed a construction that allows to use multiple time slots, by describing a specific construction which is similar to our notion of sequential TRE. The technical difficulty that we encounter should arise in their construction as well. Unfortunately, they do not provide a formal security analysis, so that this is not clarified. We, however, 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.

In order to construct sequential TLPs, we introduce the so called *gap sequential squaring assumption*, which extends the sequential squaring assumption by an oracle which takes as input the hardness parameter T' and a value y' and outputs 1 if and only if $y' = x^{2^{T'}} \bmod N$. This is akin to other gap problems [26] such as the well known gap Diffie-Hellman problem. As evidence for the hardness of this assumption, we provide an analysis in the strong algebraic group model (SAGM) and in particular show that our assumption holds as long as factoring is hard. The SAGM was introduced by Katz *et al.* [19] as a variant of the algebraic group model (AGM) [16] and enables to work with time-sensitive assumption. Finally, when modeling the above mentioned function F as a random oracle, we obtain a provably secure construction of a sequential TLP and finally a sequential TRE.

Summary and Discussion of Properties of Our Approach. There exist different approaches to construct TRE in the literature, which all aim at achieving a similar goal from a high-level perspective, but which provide very different properties from a low-level perspective, with sometimes subtle but crucial differences. We briefly summarize the properties achieved by our (sequential) TRE approach again and discuss how it enables novel applications.

Homomorphic TRE. We recall that the *homomorphic* timed-release encryption from MT19 (called HTLPs in [24]) supports homomorphic evaluation of functions on encrypted messages and avoids expensive parallel computations to solve one puzzle per ciphertext, while it still achieves the desired security against time-bounded adversaries. In some applications it may be desirable to enable the homomorphic evaluation of ciphertexts before decryption. This might be useful to save space, since it does not require storage of n encryptions of $m_1, \ldots, m_n$, but only of their homomorphic evaluation. Also a sufficiently expressive homomorphic encryption scheme that supports the homomorphic evaluation of function f is required in order to take advantage of the *"solve one, get many"* property. Practically efficient instantiations of additively and multiplicatively homomorphic schemes are readily available, but fully-homomorphic schemes [17] are currently still much less practical. So for applications that require a complex function f, the homomorphic approach from [24] is conceptually interesting, but not yet practical.

Our modular TRE costruction follows a different approach, which supports this in a black-box manner by simply replacing the PKE scheme with a *homomorphic* PKE scheme that supports homomorphic evaluation of ciphertexts. Since the existence of an additional homomorphic evaluation algorithm is merely an additional *functional* feature of the encryption scheme, the security analysis carries over without any modifications. In particular, note that our construction readily supports any (additively/multiplicatively/fully) homomorphic encryption scheme in a modular way. It thus can be based on arbitrary hardness assumptions and without introducing further requirements, such as the need for indistinguishability obfuscation for the fully-homomorphic TLP construction in [24], or the need for a specific *multi-key* fully-homomorphic encryption scheme [22] as in [8]. We note also that our construction of sequential TRE equally provides homomorphic computations *within* a single time period. Homomorphic computations *across* different time slots can easily be realized using any *multi-key* homomorphic encryption scheme [22].

Optimal Amortized Costs. Note that while MT19 [24] achieve the "solve one, get many" property only for homomorphic evaluations over many time-locked messages and thus only for functions evaluated over the time-locked messages, our TRE construction achieves this even without requiring an underlying homomorphic encryption scheme, but for all the original ciphertexts. This is because solving a puzzle yields the randomness to generate the secret key sk of the PKE scheme, which makes it possible to decrypt all ciphertexts efficiently without requiring to solve many puzzles in parallel. Note that the approach of MT19 is thus limited with respect to applicability. The homomorphic evaluation of ciphertexts is only useful when an application needs to decrypt only a function $f(m_1, \ldots, m_n)$ of all encrypted messages $m_1, \ldots, m_n$. However, if it needs to learn all n messages $m_1, \ldots, m_n$ explicitly, then MT19 still requires to solve n puzzles in parallel.

With our TRE approach it is possible to achieve the "solve one, get many" property even for applications that require the full decryption of all

independently encrypted messages. Note that for a number n of independently time-locked messages $m_1, \ldots, m_n$ our scheme is thus the first one to achieve an optimal amortized cost of decryption per ciphertext of $(n \cdot T_{\mathsf{PKE.Dec}} + T_{\mathsf{TLP}})/n$ where $T_{\mathsf{PKE.Dec}}$ is the time required to run the decryption algorithm $\mathtt{PKE.Dec}$ and T_{TLP} is the time required to solve the puzzle. Note that this approaches $T_{\mathsf{PKE.Dec}}$ with increasing n. We note that this equally applies to our sequential TRE approach.

Public Verifiability. In [15], Ephraim *et al.* recently introduced the notion of *public verifiability* for TLPs, meaning that after a party solves the puzzle, they can publish the underlying solution together with a proof which can be later used by anyone to quickly verify the correctness of the solution. Ephraim et al. require this property to hold even if the puzzle is maliciously generated and might have no valid solution. We briefly discuss how our TRE construction provides a public verifiability property, but since in our TRE the puzzles are honestly generated and part of the public parameters, we do not consider malicious puzzle generation. Note that in our TRE construction from the generated puzzle $(Z, s) \leftarrow \mathtt{TLP.Gen}(T)$ the solution s is used as the random coins to obtain $(\mathsf{pk}, \mathsf{sk}) \leftarrow \mathtt{PKE.Gen}(1^\lambda; s)$. Now, our public TRE parameters include $\mathsf{pp}_e := \mathsf{pk}$ and $\mathsf{pp}_d = Z$ and given a potential solution s' one wants to guarantee that $(\mathsf{pk}, \mathsf{sk}) \leftarrow \mathtt{PKE.Gen}(1^\lambda; s')$ generates the same public key and an equivalent secret key. Therefore, if the used PKE scheme $\mathtt{PKE} = (\mathtt{PKE.Gen}, \mathtt{PKE.Enc}, \mathtt{PKE.Dec})$ provides perfect correctness, this public verifiability property is perfectly satisfied, i.e.,. for one pk there cannot be different secret keys output by $\mathtt{PKE.Gen}$ that behave differently in their decryption behavior. In particular, the publicly verifiable proof is then simply the solution s' and the verification is to check whether $s' \in \mathcal{R}$, to run $(\mathsf{pk}', \mathsf{sk}') \leftarrow \mathtt{PKE.Gen}(1^\lambda; s')$ and to check whether $\mathsf{pk}' = \mathsf{pp}_e$, which represents an efficient check.

Sequential TRE with Public Servers. One particularly interesting feature of our notion of sequential TRE is that one can use a *single* centralized server that continuously computes and publishes solutions $s_i = \mathsf{Solve}(s_{i-1})$ to decrypt an arbitrary number of ciphertexts. Most importantly, the server would be *independent* of these ciphertexts, which is not achieved by prior constructions. This yields TRE where the decrypting parties would not have to solve any puzzle, but merely would have to wait until the server publishes a solution. Note that here the fact that the amortized complexity of decrypting n ciphertexts approaches the complexity of running $\mathtt{PKE.Dec}$ with increasing n is particularly useful.

We stress that this must not be confused with TRE schemes in a trusted-agent based setting. Loosely speaking, in such schemes a so called time server publishes a single time-dependent trapdoor that then allows decryption of ciphertexts. As shown in [10], this concept is essentially equivalent to identity-based encryption (IBE) [5]. Most importantly, in any such agent-based TRE scheme there is a trusted party which is not only involved in running the setup, but this party then also needs to be online and, in particular, needs to be trusted to keep the secret keys that are supposed to be released at a later point in time confidential until the time has passed. In our approach of sequential TRE with public servers, however, only the setup needs to be trusted. Even for the service that

actually performs the squaring there are no shortcuts to revealing the decryption keys before the respective time has passed.

If one is worried about a trusted setup performed by a third-party server, or about the fact that one server might run out of service, then one could use $N > 1$ servers. The public parameters of each server would be used to encrypt a share of the message, using an (K, N)-threshold secret sharing scheme (e.g., [30]). Even with $K - 1$ colluding servers, the message would remain hidden. Even if up to $N - K$ servers go out of service, messages would still be recoverable using the K shares obtained from the remaining servers.

3 Definitions and Constructions of Time Lock-Puzzles

Simple Time-Lock Puzzles. In this section we give a new definition for time-lock puzzles (TLPs) and explain how it relates to the old definition.

Definition 1. *A time-lock puzzle is pair of algorithms* $\mathtt{TLP} = (\mathsf{Gen}, \mathsf{Solve})$ *with the following syntax.*

- $(Z, s) \leftarrow \mathsf{Gen}(T)$ *is a probabilistic algorithm which takes as input a hardness parameter* $T \in \mathbb{N}$ *and outputs a puzzle* Z *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 which 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 will also be a lower bound on the running time, which is part of the security definition.*

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

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

Relation to Prior Definitions. In the definitions of TLPs from Bitansky *et al.* [4] and Malavolta and Thyagarajan [24] algorithm Gen receives s as an additional input and output 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:

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

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

Fig. 1. Security experiment for time-lock puzzles.

Security. 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. The following definition is inspired by those from Bitansky *et al.* [4] and Malavolta and Thyagarajan [24], but adopted to our slightly modified definition of the Gen algorithm.

Definition 2. *Consider the security experiment* $\mathsf{ExpTLP}^b_{\mathcal{A}}(\lambda)$ *in Fig. 1. We say that a time-lock puzzle* TLP *is* secure with gap $\epsilon < 1$*, if there exists a polynomial* $\tilde{T}(\cdot)$ *such that for all polynomials* $T(\cdot) \geq \tilde{T}(\cdot)$ *and every polynomial-size adversary* $\mathcal{A} = \{\mathcal{A}_\lambda\}_{\lambda\in\mathbb{N}}$ *of depth* $\leq T^\epsilon(\lambda)$ *there exists a negligible function* $\mathsf{negl}(\cdot)$ *such that for all* $\lambda \in \mathbb{N}$ *it holds*

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

Other Variants of TLPs. In the full version we also discuss weaker forms of TLPs as introduced by Bitansky *et al.* [4]. Moreover, we present instantiations of TLPs based on different variants of randomized encodings [2,18] and in particular the approach of constructing TLPs from them by Bitansky et al. [4]. Furthermore, we discuss how they can be cast into our TLP framework.

Instantiating TLPs from Sequential Squaring. Subsequently, we discuss instantiations of TLPs based on the sequential squaring. Therefore, we recall a definition of the sequential squaring assumption which was implicitly introduced by Rivest et al. [29]. 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 $\varphi(\cdot)$ denotes Euler's totient function. We denote by $\mathbb{QR}_N$ the cyclic group of quadratic residues which has order $|\mathbb{QR}_N| = \frac{\varphi(N)}{4} = \frac{(p-1)(q-1)}{4}$.

Definition 3 (The Sequential Squaring Assumption). *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* $\lambda \in \mathbb{N}$

$$\left|\Pr\left[b = b' \;:\; \begin{array}{r}(p,q,N) \leftarrow \mathsf{GenMod}(1^\lambda)\\ x \xleftarrow{\$} \mathbb{QR}_N, b \xleftarrow{\$} \{0,1\}\\ \textit{if } b = 0 : y := x^{2^{T(\lambda)}} \bmod N\\ \textit{if } b = 1 : y \xleftarrow{\$} \mathbb{QR}_N\\ b' \leftarrow \mathcal{A}_\lambda(N, T(\lambda), x, y)\end{array}\right] - \frac{1}{2}\right| \leq \mathsf{negl}(\lambda).$$

The instantiation of TLP from the sequential squaring assumption is straightforward:

- $\mathsf{Gen}(T)$: Run $(p,q,N) \leftarrow \mathsf{GenMod}(1^\lambda)$. Randomly sample $x \xleftarrow{\$} \mathbb{QR}_N$ and compute the value $s := x^{2^T} \bmod N$. Notice that value s can be efficiently computed knowing the values p and q. 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 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 this assumption to hold, assuming factoring is hard, in the strong algebraic group model (SAGM) of Katz *et al.* [19].

Defining Sequential Time-Lock Puzzles. *Sequential time-lock puzzles* are a particularly useful generalization of basic TLPs, which yields particularly practical time-lock encryption schemes. To this end, we generalize Definition 1 by allowing the Gen algorithm to take multiple different time parameters as input, which then produces a corresponding set of puzzles.

Definition 4. *A* sequential time-lock puzzle *is tuple of algorithms* $\mathtt{sTLP} = (\mathsf{Gen}, \mathsf{Solve})$ *with the following syntax.*

- $(Z_i, s_i)_{i\in[n]} \leftarrow \mathsf{Gen}((T_i)_{i\in[n]})$ *is a probabilistic algorithm which takes as input* n *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 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 which takes as input a puzzle* Z_i *and a solution for puzzle* Z_{i-1} *and outputs a solution* s_i*, where we define* $s_0 := \perp$*. 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 $i \in [n]$ and for polynomials T_i in λ such that $T_i < T_{i+1}$ it holds:*

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

Security. 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 $\mathsf{F} : \mathbb{N} \times S \to Y$, which takes as input a pair a hardness parameter $T \in \mathbb{N}$ together with 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.

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

Fig. 2. Security experiment for sequential time-lock puzzles.

Definition 5. *Consider the security experiment* $\mathsf{ExpsTLP}^b_{\mathcal{A}_i}(\lambda)$ *in Fig. 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*

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

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 takes as input hardness parameter T' and value $y' \in \mathbb{QR}_N$ and outputs 1 if $y' = x^{2^{T'}} \bmod N$, otherwise it outputs 0. The values x and N are defined in security experiment. The assumption essentially states that computational sequential squaring assumption remains hard, even if the adversary is given access to DSSvf, akin to other gap assumptions [26]. We discuss the necessity of this assumption in Appendix C.

Definition 6 (The Gap Sequential Squaring (GGS) 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:

- $\mathsf{Gen}((T_i)_{i\in[n]})$: Run $(p,q,N) \leftarrow \mathsf{GenMod}(1^\lambda)$. Randomly sample $x \xleftarrow{\$} \mathbb{QR}_N$ and compute values $s_i := x^{2^{T_i}} \bmod N$ for all $i \in [n]$. Value s_i can be efficiently computed knowing the values p and q. 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 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* $\mathtt{sTLP} = (\mathsf{Gen}, \mathsf{Solve})$ *defined above is a secure sequential time-lock puzzle with gap* $\underline{\epsilon}$ *and with respect to the function* F*.*

In the full version we prove that the gap sequential squaring problem is at least as hard as factoring N in the Strong Algebraic Group Model (SAGM), which was introduced by Katz *et al.* [19] to consider time-sensitive assumptions.

Theorem 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* $0 < \epsilon < 1$*.*

5 (Sequential) Timed-Release Encryption

In this section we give generic constructions of (sequential) timed-release encryption (TRE) schemes based on (sequential) TLPs. There exist several definitions for TRE and we base ours on that of Unruh [32]. However, we introduce two additional algorithms Setup and Solve which leads to better modularity and applicability of TRE, as we will illustrate in Supplementary Material B.

Definition 7. *A sequential timed-release encryption scheme with message space* $\mathcal{M}$ *is 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 which takes as input a security parameter* 1^λ *and a set of time hardness parameters* $(T_i)_{i\in[n]}$ *with* $T_i < T_{i+1}$ *for all* $i \in [n-1]$*, and outputs set of public encryption parameters and public decryption parameters* $\mathsf{PP} := (\mathsf{pp}_{e,i}, \mathsf{pp}_{d,i})_{i\in[n]}$*. 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 which takes as input public decryption parameters* $\mathsf{pp}_{d,i}$ *and a solution from a previous iteration* s_{i-1}*, where* $s_0 := \perp$*, 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 which 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:*

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

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 Dec algorithm, however, for sequential TRE, this value is necessary. For ease of the notation, it is unified.

Definition 8. *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* $\mathcal{A} = \{(\mathcal{A}_{1,\lambda}, \mathcal{A}_{2,\lambda})\}_{\lambda\in\mathbb{N}}$ *there exists a negligible function* $\mathsf{negl}(\cdot)$ *such that for all* $\lambda \in \mathbb{N}$ *it holds*

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

We require that $|m_0| = |m_1|$ *and an adversary* $\mathcal{A}_\lambda = (\mathcal{A}_{1,\lambda}, \mathcal{A}_{2,\lambda})$ *where* $\mathcal{A}_{1,\lambda}$ *outputs* i *in the second step of the above security experiment consists of two circuits with* total *depth at most* $T^\epsilon_i(\lambda)$ *(i.e., the total depth is the sum of the depth of* $\mathcal{A}_{1,\lambda}$ *and* $\mathcal{A}_{2,,\lambda}$*).*

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

Fig. 3. Construction of TRE

5.1 Basic TRE Construction

Building Blocks. Our construction combines a time-lock puzzle (TLP) with a CPA secure public-key encryption (PKE) scheme. We refer to [11] for standard formal syntactical and security definitions of PKE. We require standard CPA security of the PKE scheme, since this is sufficient to construct a TRE scheme achieving Definition 8[1].

Construction. Let $\mathtt{TLP} = (\mathtt{TLP}.\mathsf{Gen}, \mathtt{TLP}.\mathsf{Solve})$ be a TLP with solution space S and let $\mathtt{PKE} = (\mathtt{PKE}.\mathsf{Gen}, \mathtt{PKE}.\mathsf{Enc}, \mathtt{PKE}.\mathsf{Dec})$ be a PKE scheme. Figure 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 in the construction. Observe that correctness is directly implied by correctness of the PKE scheme and the TLP.

Theorem 3. *If* $\mathtt{TLP} = (\mathtt{TLP}.\mathsf{Gen}, \mathtt{TLP}.\mathsf{Solve})$ *is secure time-lock puzzle with gap* ϵ *in the sense of Definition 2 and* $\mathtt{PKE} = (\mathtt{PKE}.\mathsf{Gen}, \mathtt{PKE}.\mathsf{Enc}, \mathtt{PKE}.\mathsf{Dec})$ *is a CPA secure encryption scheme, then* $\mathtt{TRE} = (\mathsf{Setup}, \mathsf{Solve}, \mathsf{Enc}, \mathsf{Dec})$ *defined in Fig. 3 is a secure timed-release encryption scheme with gap* $\underline{\epsilon} < \epsilon$ *in the sense of Definition 8.*

5.2 Sequential TRE

In the sequel let $\mathtt{sTLP} = (\mathtt{sTLP}.\mathsf{Gen}, \mathtt{sTLP}.\mathsf{Solve})$ be a sequential TLP in the sense of Definition 4 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 $\mathtt{sTLP}$ 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 Fig. 4. Note that correctness of the scheme is directly implied by correctness of the PKE scheme and the sequential TLP.

[1] We note that by replacing the PKE with a CCA secure one, we can straightforwardly obtain a CCA secure TRE. This is easily achieved as our puzzles are included in the public parameters and thus do not need to be non-malleable and we only require non-malleability on the ciphertexts.

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

$\underline{\mathsf{Enc}(\mathsf{pp}_{e,i}, m)}$
return $c \leftarrow \mathtt{PKE}.\mathsf{Enc}(\mathsf{pp}_{e,i}, m)$

$\underline{\mathsf{Solve}(\mathsf{pp}_{d,i}, s_{i-1})}$
$s_i \leftarrow \mathtt{sTLP}.\mathsf{Solve}(\mathsf{pp}_{d,i}, s_{i-1})$
return s_i

$\underline{\mathsf{Dec}(T_i, s_i, c)}$
$(\mathsf{pk}_i, \mathsf{sk}_i) \leftarrow \mathtt{PKE}.\mathsf{Gen}(1^\lambda; \mathsf{F}(T_i, s_i))$
return $m \leftarrow \mathtt{PKE}.\mathsf{Dec}(\mathsf{sk}_i, c)$

Fig. 4. Construction of sequential TRE

$\underline{\mathsf{Gen}(1^\lambda, \mathcal{F}, (T_j)_{j\in[n]})}$
$(\mathsf{pk}, \mathsf{msk}) \leftarrow \mathtt{FE}.\mathsf{Gen}(1^\lambda, \mathcal{F})$
$(\mathsf{pp}_{e,j}, \mathsf{pp}_{d,j})_{j\in[n]} \leftarrow \mathtt{TRE}.\mathsf{Setup}(1^\lambda, (T_j)_{j\in[n]})$
return $(\mathsf{pk}, \mathsf{msk}, (\mathsf{pp}_{e,j}, \mathsf{pp}_{d,j})_{j\in[n]})$

$\underline{\mathsf{KeyGen}(\mathsf{msk}, (\mathsf{pp}_{e,j})_{j\in[n]}, f, i)}$
$\mathsf{sk}_f \leftarrow \mathtt{FE}.\mathsf{KeyGen}(\mathsf{msk}, f)$
$c_i \leftarrow \mathtt{TRE}.\mathsf{Enc}(\mathsf{pp}_{e,i}, \mathsf{sk}_f)$
return $\mathsf{dk}_i := c_i$

$\underline{\mathsf{Solve}(\mathsf{pp}_{d,i}, s_{i-1})}$
return $s_i := \mathtt{TRE}.\mathsf{Solve}(\mathsf{pp}_{d,i}, s_{i-1})$

$\underline{\mathsf{Enc}(\mathsf{pk}, x)}$
return $c \leftarrow \mathtt{FE}.\mathsf{Enc}(\mathsf{pk}, x)$

$\underline{\mathsf{Dec}(\mathsf{dk}_i, T_i, s_i, c)}$
$c_i := \mathsf{dk}_i$
$\mathsf{sk}_f := \mathtt{TRE}.\mathsf{Dec}(T_i, s_i, c_i)$
return $f(x) := \mathtt{FE}.\mathsf{Dec}(\mathsf{sk}_f, c)$

Fig. 5. Construction of TRFE.

Theorem 4. *If* $\mathtt{sTLP} = (\mathtt{sTLP.Gen}, \mathtt{sTLP.Solve})$ *is a secure sequential time-lock puzzle with gap* ϵ *w.r.t. function* F *and* $\mathtt{PKE} = (\mathtt{PKE.Gen}, \mathtt{PKE.Enc}, \mathtt{PKE.Dec})$ *is a CPA secure encryption scheme, then* $\mathtt{TRE} = (\mathsf{Setup}, \mathsf{Enc}, \mathsf{Solve}, \mathsf{Dec})$ *defined in Fig. 4 is a secure sequential timed-release encryption with gap* $\underline{\epsilon} < \epsilon$.

5.3 Integrating Timed-Release Features into Functional Encryption

In this section, we connect sequential timed-release features with functional encryption (FE) [7,27] and introduce the notion of a (sequential) timed-release functional encryption (TRFE) scheme. The basic idea is that in such a scheme, similarly to an FE scheme, there is a public key pk used for encryption of any message x and a main secret key msk which is associated to a class of functions $\mathcal{F} : \mathcal{X} \to \mathcal{Y}$. In contrast to an FE scheme, however, in a TRFE scheme, msk can be used to generate decryption keys for a function $f \in \mathcal{F}$ which is associated to a time hardness parameter T_i (and, hence, to its solution s_i). Decryption takes the associated decryption key dk_i, the solution s_i, a function $f \in \mathcal{F}$, and a ciphertext to message x and outputs $f(x)$. Security-wise, an adversary is allowed to query *any* secret function key for any public encryption parameter associated

to T_i as long as its solution s_i is not retrievable. Due to limited space, we present the formal definitions of FE, the formal framework of timed-release functional encryption (TRFE) as well as the proof that the construction in Fig. 5 is a secure TRFE in the full version.

Construction of TRFE. Let $\mathtt{TRE} = (\mathtt{TRE.Setup}, \mathtt{TRE.Solve}, \mathtt{TRE.Enc}, \mathtt{TRE.Dec})$ be a (sequential) TRE scheme and $\mathtt{FE} = (\mathtt{FE.Gen}, \mathtt{FE.KeyGen}, \mathtt{FE.Enc}, \mathtt{FE.Dec})$ be an FE scheme. We construct a TRFE scheme $\mathsf{TRFE} = (\mathsf{Setup}, \mathsf{KeyGen}, \mathsf{Enc}, \mathsf{Solve}, \mathsf{Dec})$ as given in Fig. 5. Let the message space of TRE be the functional-secret-key space of FE which is the output of FE.KeyGen and all functional secret keys for function $f \in \mathcal{F}$ and any main secret key of FE are of equal length.

Application to Locked-Key IBE. With TRFE, we are able to lock secret keys of an IBE scheme with a sequential timed-release feature. When the central authority in an IBE scheme generates the identity-based secret keys, it can attach hardness parameters to it such that those keys only become usable sequentially. This, for example, enables an IBE central authority to produce all secret keys in the beginning and afterwards go off-line.

A Concurrent and Independent Work

Recently, there have been some independent and concurrent works investigating different aspects of TLPs, which we want to briefly discuss. Most closely related to our work is the one of Katz *et al.* [19] who show that sequential squaring is as hard as factoring in the strong algebraic group model (SAGM) and construct non-malleable timed commitments based upon a novel building block called timed public-key encryption (TPKE). The similarities are that we will also rely on the SAGM to prove the generic hardness of our new gap sequential squaring assumption. However, their TRPKE approach is different to our TRE approach. Firstly, they support a fast and a slow decryption, where former uses the secret key and latter requires solving a TLP. Secondly, while in our setting encryption is efficient, in their TPKE which is constructed from sequential squaring and the Naor-Yung double encryption paradigm one has to compute twice a T-times sequential squaring. This construction achieves CCA security, but they also discuss a CPA secure version where encryption is equivalently expensive. Note that in contrast to our TRE, in their TPKE time starts running with encryption and not with parameter generation.

In [15] Ephraim *et al.* investigate efficient constructions of concurrent non-malleable TLPs in the auxiliary-input random oracle model (whereas previous constructions in the plain model [3] are not practically efficient). The idea, which is similar to our idea, is essentially to evaluate random oracle on hardness parameter T and solution s of a puzzle Z and use the output of the oracle as a randomness for Gen algorithm of any TLP. An interesting property introduced and investigated in [15] is public verifiability of TLPs. As we have already discussed we can achieve public verifiability for our generic TRE when basing them on

perfectly correct PKE schemes. We note however that while Ephraim et al. consider this notion in a setting with malicious puzzle generation, we consider a weaker notion with honest puzzle generation which is sufficient for our TRE.

Abadi and Kiayias [1] construct so-called Multi-Instance Time-Lock Puzzles (MITLPs) which are similar to our notion of sequential TRE. The crucial difference is that MILTPs allow to encrypt messages with respect to consecutive multiples of one hardness parameter by chaining TLPs which requires that all messages of interest must be known at the time when MITLP is generated.

B Applications: Simpler and More Efficient Instantiations

Subsequently, we discuss the applications in [24] when we use our (homomorphic) TRE approach in contrast to HTLPs of MT19. All the following application have in common that they require decrypting a set of encrypted messages at some required time. Our approach to TRE allows to decrypt arbitrary number of messages at the specified time by solving one puzzle. In [24] this is achieved by 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 homomorphic evaluation and only after that the resulting puzzles can be solved. Our scheme allows to start to solve the puzzle immediately after Setup is run. In all of this 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 trusted party, where voters are able to cast their preference without any bias. Similarly to [24], we do not consider privacy nor authenticity of the votes. The crucial property of our $\mathtt{TRE}$ is that 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 to obtain the secret key which is then used to decrypt all ciphertexts that have been produced using 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 final result by combining decrypted plaintexts. 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 [24], we have to wait until the voting phase is finished and then we can combine puzzles from 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, e.g., $i \in [m]$ 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 [24] relies on HTLPs and their protocol consist 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 consist of two phases - the bidding phase and the opening phase. Bids should be kept secret during the bidding phase and later revealed in 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 use only standard time-lock puzzles, then the number of puzzles which has to be solved in the opening phase is equal to number of bidders who went offline. In [24] 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 [24] 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 multiple hardness parameters. We remark that this type of time-lock puzzles are in some sense equivalent to our sequential timed-release encryption.[2] 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 decrease 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 into account time for homomorphic evaluation, 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 are able to obtain 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.

C On the Necessity of the Gap Sequential Squaring Assumption

One might ask why the following seemingly simple solution does not yield a secure sequential TLP:

[2] Though they only discuss them informally in [24] and as mentioned in Sect. 1 it seems that it is not possible to prove it secure as it is proposed.

– Generate a set of (non-sequential) puzzles $(Z_1, s_1), ..., (Z_n, s_n)$, such that the delay parameter for puzzle i is $T_i - T_{i-1}$.
– Let $(\mathsf{Enc}, \mathsf{Dec})$ be some CPA-secure symmetric encryption scheme.
– Publish $(Z_1, (\mathsf{Enc}_{s_{i-1}}(Z_i))_{i=2}^n)$.

Unfortunately, this approach does not work (see also page 16 of the full version [11]). Concretely, suppose we have an adversary which has sufficient running time to solve $n-1$ puzzles, and then successfully attacks the n-th puzzle (say, with success probability 1, for instance). Now note that we cannot use the CPA security of any of the first $n-1$ encryptions to "hide" any intermediate puzzle, because the adversary has enough time to notice this (as it has enough running time to solve all the first $n-1$ puzzles). However, then, since we cannot use the CPA security as an argument in the proof, we can equivalently consider the first $n-1$ encryption as completely insecure.

But if we have no security guarantees for the first $n-1$ encryptions, this means that we also cannot argue that the adversary cannot obtain the n-th puzzle instance Z_n quickly, without solving $n-1$ prior instances, because it could simply "break" the $(n-1)$-th encryption to obtain Z_n (which might be very quick, e.g., in just a few computational steps, because we cannot argue that the scheme is secure in the sense of CPA or some other notion). And then an adversary with sufficient running time to solve $n-1$ puzzles can simply solve only the last puzzle using the standard Solve algorithm.

So in conclusion, we do not claim that the construction is insecure, but only that CPA security or a similar notion seems not sufficient, as we cannot perform a reduction to the CPA security in the usual way. It might be possible to prove security under a non-standard assumption (e.g., by essentially assuming that the proposed construction is secure), however, this would also be an additional assumption, which we want to avoid.

References

1. Abadi, A., Kiayias, A.: Multi-instance publicly verifiable time-lock puzzle and its applications. In: Financial Cryptography and Data Security (2021)
2. Applebaum, B., Ishai, Y., Kushilevitz, E.: Computationally private randomizing polynomials and their applications. Comput. Complex **15**, 115–162 (2006)
3. Ball, M., Dachman-Soled, D., Kulkarni, M., Lin, H., Malkin, T.: Non-malleable codes against bounded polynomial time tampering. Cryptology ePrint Archive, Report 2018/1015. https://eprint.iacr.org/2018/1015
4. Bitansky, N., Goldwasser, S., Jain, A., Paneth, O., Vaikuntanathan, V., Waters, B.: Time-lock puzzles from randomized encodings. In: 7th Conference on Innovations in Theoretical Computer Science, ITCS 2016 (2016)
5. Boneh, D., Franklin, M.: Identity-based encryption from the Weil pairing. In: Kilian, J. (ed.) CRYPTO 2001. LNCS, vol. 2139, pp. 213–229. Springer, Heidelberg (2001). https://doi.org/10.1007/3-540-44647-8_13
6. Boneh, D., Naor, M.: Timed commitments. In: Bellare, M. (ed.) CRYPTO 2000. LNCS, vol. 1880, pp. 236–254. Springer, Heidelberg (2000). https://doi.org/10.1007/3-540-44598-6_15

7. Boneh, D., Sahai, A., Waters, B.: Functional encryption: definitions and challenges. In: Ishai, Y. (ed.) TCC 2011. LNCS, vol. 6597, pp. 253–273. Springer, Heidelberg (2011). https://doi.org/10.1007/978-3-642-19571-6_16
8. Brakerski, Z., Döttling, N., Garg, S., Malavolta, G.: Leveraging linear decryption: rate-1 fully-homomorphic encryption and time-lock puzzles. In: Hofheinz, D., Rosen, A. (eds.) TCC 2019, Part II. LNCS, vol. 11892, pp. 407–437. Springer, Cham (2019). https://doi.org/10.1007/978-3-030-36033-7_16
9. Brakerski, Z., Gentry, C., Vaikuntanathan, V.: (Leveled) fully homomorphic encryption without bootstrapping. In: 3rd Innovations in Theoretical Computer Science, ITCS 2012 (2012)
10. Cheon, J.H., Hopper, N., Kim, Y., Osipkov, I.: Provably secure timed-release public key encryption. ACM Trans. Inf. Syst. Secur. **11**, 1–44 (2008)
11. Chvojka, P., Jager, T., Slamanig, D., Striecks, C.: Versatile and sustainable timed-release encryption and sequential time-lock puzzles. Cryptology ePrint Archive, Report 2020/739. https://eprint.iacr.org/2020/739
12. Di Crescenzo, G., Ostrovsky, R., Rajagopalan, S.: Conditional oblivious transfer and timed-release encryption. In: Stern, J. (ed.) EUROCRYPT 1999. LNCS, vol. 1592, pp. 74–89. Springer, Heidelberg (1999). https://doi.org/10.1007/3-540-48910-X_6
13. Dwork, C., Naor, M.: Zaps and their applications. In: 41st Annual Symposium on Foundations of Computer Science (2000)
14. ElGamal, T.: A public key cryptosystem and a signature scheme based on discrete logarithms. In: Blakley, G.R., Chaum, D. (eds.) CRYPTO 1984. LNCS, vol. 196, pp. 10–18. Springer, Heidelberg (1985). https://doi.org/10.1007/3-540-39568-7_2
15. Ephraim, N., Freitag, C., Komargodski, I., Pass, R.: Non-malleable time-lock puzzles and applications. Cryptology ePrint Archive, Report 2020/779. https://eprint.iacr.org/2020/779
16. Fuchsbauer, G., Kiltz, E., Loss, J.: The algebraic group model and its applications. In: Shacham, H., Boldyreva, A. (eds.) CRYPTO 2018, Part II. LNCS, vol. 10992, pp. 33–62. Springer, Cham (2018). https://doi.org/10.1007/978-3-319-96881-0_2
17. Gentry, C.: Fully homomorphic encryption using ideal lattices. In: 41st Annual ACM Symposium on Theory of Computing (2009)
18. Ishai, Y., Kushilevitz, E.: Randomizing polynomials: a new representation with applications to round-efficient secure computation. In: 41st Annual Symposium on Foundations of Computer Science (2000)
19. Katz, J., Loss, J., Xu, J.: On the security of time-lock puzzles and timed commitments. In: Pass, R., Pietrzak, K. (eds.) TCC 2020, Part III. LNCS, vol. 12552, pp. 390–413. Springer, Cham (2020). https://doi.org/10.1007/978-3-030-64381-2_14
20. Lin, H., Pass, R., Soni, P.: Two-round and non-interactive concurrent non-malleable commitments from time-lock puzzles. In: 58th Annual Symposium on Foundations of Computer Science (2017)
21. Liu, J., Jager, T., Kakvi, S.A., Warinschi, B.: How to build time-lock encryption. Des. Codes Cryptogr. **86**(11), 2549–2586 (2018). https://doi.org/10.1007/s10623-018-0461-x
22. López-Alt, A., Tromer, E., Vaikuntanathan, V.: On-the-fly multiparty computation on the cloud via multikey fully homomorphic encryption. In: 44th Annual ACM Symposium on Theory of Computing (2012)
23. Mahmoody, M., Moran, T., Vadhan, S.: Time-lock puzzles in the random oracle model. In: Rogaway, P. (ed.) CRYPTO 2011. LNCS, vol. 6841, pp. 39–50. Springer, Heidelberg (2011). https://doi.org/10.1007/978-3-642-22792-9_3

24. Malavolta, G., Thyagarajan, S.A.K.: Homomorphic time-lock puzzles and applications. In: Boldyreva, A., Micciancio, D. (eds.) CRYPTO 2019, Part I. LNCS, vol. 11692, pp. 620–649. Springer, Cham (2019). https://doi.org/10.1007/978-3-030-26948-7_22
25. May, T.C.: Timed-release crypto. Technical report (1993)
26. Okamoto, T., Pointcheval, D.: The gap-problems: a new class of problems for the security of cryptographic schemes. In: Kim, K. (ed.) PKC 2001. LNCS, vol. 1992, pp. 104–118. Springer, Heidelberg (2001). https://doi.org/10.1007/3-540-44586-2_8
27. O'Neill, A.: Definitional issues in functional encryption. Cryptology ePrint Archive, Report 2010/556. https://eprint.iacr.org/2010/556
28. Paillier, P.: Public-key cryptosystems based on composite degree residuosity classes. In: Stern, J. (ed.) EUROCRYPT 1999. LNCS, vol. 1592, pp. 223–238. Springer, Heidelberg (1999). https://doi.org/10.1007/3-540-48910-X_16
29. Rivest, R.L., Shamir, A., Wagner, D.A.: Time-lock puzzles and timed-release crypto. Technical report (1996)
30. Shamir, A.: How to share a secret. Commun. ACM (1979)
31. Thyagarajan, S.A.K., Bhat, A., Malavolta, G., Döttling, N., Kate, A., Schröder, D.: Verifiable timed signatures made practical. In: ACM CCS 2020 (2020, to appear). https://verifiable-timed-signatures.github.io/web/assets/paper.pdf
32. Unruh, D.: Revocable quantum timed-release encryption. In: Nguyen, P.Q., Oswald, E. (eds.) EUROCRYPT 2014. LNCS, vol. 8441, pp. 129–146. Springer, Heidelberg (2014). https://doi.org/10.1007/978-3-642-55220-5_8