Leveraging Linear Decryption: Rate-1 Fully-Homomorphic Encryption and Time-Lock Puzzles

Zvika Brakerski*
Weizmann Institute of Science

Nico Döttling
CISPA Helmholtz Center for Information Security

Sanjam Garg†
University of California, Berkeley

Giulio Malavolta‡
Carnegie Mellon University

Abstract

We show how to combine a fully-homomorphic encryption scheme with linear decryption and a linearly-homomorphic encryption schemes to obtain constructions with new properties. Specifically, we present the following new results.

(1) Rate-1 Fully-Homomorphic Encryption: We construct the first scheme with message-to-ciphertext length ratio (i.e., rate) $1 - \sigma$ for $\sigma = o(1)$. Our scheme is based on the hardness of the Learning with Errors (LWE) problem and σ is proportional to the noise-to-modulus ratio of the assumption. Our building block is a construction of a new high-rate linearly-homomorphic encryption.

One application of this result is the first general-purpose secure function evaluation protocol in the preprocessing model where the communication complexity is within *additive* factor of the optimal *insecure* protocol.

(2) Fully-Homomorphic Time-Lock Puzzles: We construct the first time-lock puzzle where one can evaluate any function over a set of puzzles without solving them, from standard assumptions. Prior work required the existence of sub-exponentially hard indistinguishability obfuscation.

1 Introduction

Fully-homomorphic encryption (FHE) allows one to evaluate any function over encrypted data. Since the breakthrough result of Gentry [Gen09], the development of FHE schemes has seen a rapid surge [vGHV10, BV11, BGV12, GSW13, BV14, AP14] and by now FHE has become a well-established cryptographic primitive. An FHE scheme gives an elegant solution to the problem of secure function evaluation: One party publishes the encryption of its input under its own public key $\mathsf{Enc}(\mathsf{pk}, x)$ while the other evaluates some function f homomorphically, returning $c = \mathsf{Enc}(\mathsf{pk}, f(x))$. The first party can recover the output by simply decrypting c. The crucial property of this approach is that its communication complexity is proportional to the size of the input and of the output, but does not otherwise depend on the size of f. This distinguishing feature is essential for certain applications, such as private information retrieval [CGKS95], and has motivated a large body of work on understanding FHE and related notions [BGI16, QWW18].

Unfortunately, our understanding in secure computation protocol with *optimal* communication complexity is much more limited. Typically, FHE schemes introduce a polynomial blowup factor (in the security

*Supported by the Israel Science Foundation (Grant No. 468/14), Binational Science Foundation (Grants No. 2016726, 2014276) and European Union Horizon 2020 Research and Innovation Program via ERC Project REACT (Grant 756482) and via Project PROMETHEUS (Grant 780701).

†Supported in part from DARPA/ARL SAFEWARE Award W911NF15C0210, AFOSR Award FA9550-15-1-0274, AFOSR YIP Award, DARPA and SPAWAR under contract N66001-15-C-4065, a Hellman Award and research grants by the Okawa Foundation, Visa Inc., and Center for LongTerm Cybersecurity (CLTC, UC Berkeley).

‡Supported in part by a gift from Ripple, a gift from DoS Networks, a grant from Northrop Grumman, a Cylab seed funding award, and a JP Morgan Faculty Fellowship.

parameter) to the cipheretext size, thereby affecting the overall communication rate of the protocol. Given the current state-of-the-art FHE schemes, the only class of functions we can evaluate without communication blowup are linear functions [DJ01]. An FHE scheme with optimal rate, i.e., with a message-to-ciphertext ratio approaching 1, would immediately give us a general-purpose tool to securely evaluate any function (with sufficiently large inputs and outputs) with asymptotically optimal communication complexity. Motivated by this objective, this work seeks to answer the following question:

Can we construct an FHE scheme with rate 1 from standard assumptions?

We also consider the related problem of constructing fully-homomorphic time-lock puzzles (FH-TLP), a primitive recently introduced in [MT19] to address the computational burden of classical time-lock puzzles [RSW96]. Time-lock puzzles encapsulate secrets for a pre-determined amount of time, and FH-TLP allow one to evaluate functions over independently generated puzzles. The key feature of FH-TLPs is that after a function has been homomorphically evaluated on a (possibly large) number of input TLPs, only a single output TLP has to be solved to recover the function result. Consquently, FH-TLP can be used in the very same way as TLPs, but the solver is spared from solving a large number of TLPs (in parallel) and only needs to solve a single TLP which encapsulates the function result.

FH-TLP have been shown to be a very versatile tool and have several applications, ranging from coin-flipping to fair contract signing [MT19]. In [MT19] FH-TLPs were constructed from probabilistic iO [CLTV15] and scheme from standard assumptions were limited to restricted classes of functions (e.g., linear functions). Motivated by this gap, the second question that we ask is:

Can we construct an FH-TLP scheme (ideally with rate 1) from standard assumptions?

1.1 Our Results

In this work, we answer both questions in the affirmative. Specifically, we present the following new results:

(1) Our main result is the construction of an FHE which allows compressing many ciphertexts into a *compressed ciphertext* which has rate 1 $1/\lambda$. In fact, we show that for any a-priori block size $\ell = \mathsf{poly}(\lambda)$, we can construct a scheme where the ciphertext length is at most $\ell + \tau(\lambda)$, where τ is a fixed polynomial (which does not depend on ℓ). Setting $\ell = \lambda \ \tau(\lambda)$, the rate claim follows.

To prove security of this scheme, we only need to assume the hardness of the Learning With Errors (LWE) [Reg05] problem with polynomial modulus-to-noise ratio.[1]

(2) We provide a construction of a fully-homomorphic time-lock puzzle from multi-key FHE and linearly homomorphic time-lock puzzles. The security of the former can be based on the hardness of LWE with superpolynomial modulus-to-noise ratio, whereas the latter can be constructed from the sequential squaring assumption [RSW96] in groups of unknown order.

On a technical level, both of our main results are tied together by the common idea of combining an FHE with a linear decryption algorithm with a linearly-homomorphic encryption (time-lock puzzle, respectively) of optimal rate. The hybrid scheme inherits the best of both worlds and gives us a rate-optimal FHE scheme or an FH-TLP from standard assumptions, depending on the building block that we use. Our techniques are reminiscent of the chimeric scheme of Gentry and Halevi [GH11], with a new twist to how to encode information without inflating the size of the ciphertexts. Somewhat interestingly, our construction of rate-1 linearly homomorphic encryption from LWE leverages ideas which were originally conceived in the context spooky FHE [DHRW16], homomorphic secret sharing [BKS19] and private-information retrieval [DGI+19].

Concurrent Work. In a concurrent work, Gentry and Halevi [GH19] constructed rate-1 FHE schemes using similar ideas as in our work. While the goal of their work is realizing practically efficient high-rate private information retrieval protocols, our constructions are more general and designed to achieve the best possible asymptotic rate.

[1] We note that the modulus-to-noise ratio does depend (linearly) on ℓ.

1.2 Applications

We outline a few interesting implications of our results. We stress that the tools that we develop in this work are of general purpose and we expect them to find more (possibly indirect) applications in the near future.

(1) Secure Function Evaluation: FHE yields a very natural protocol for secure function evaluation (SFE) where one party encrypts its input and the other computes the function homomorphically. Given that the input and the output are sufficiently large, rate-1 FHE yields a (semi-honest) SFE scheme where the communication complexity is within *additive* factor from that of the best possible insecure protocol.

(2) Encrypted Databases with Updates: Using rate-1 FHE, it is possible to outsource an encrypted database to an untrusted (semi-honest) cloud provider, without suffering additional storage overhead due to ciphertext expansion. While FHE hybrid encryption (using a non-rate-1 FHE) allows to store a static database without additional storage requirements, as soon as database entries are homomorphically updated they become *FHE-ciphertexts* and consequently their size grows substantially. Keeping the database encrypted under a rate-1 FHE scheme enables the cloud provider to perform updates on the database, while not increasing the size of the encrypted data.

(3) Malicious Circuit Privacy: Instantiating the generic compiler of Ostrovsky et al. [OPP14] with our rate-1 FHE scheme gives the first maliciously circuit-private FHE scheme with rate-1. A maliciously circuit-private scheme does not leak any information to the decrypter about the homomorphically evaluated functions (beyond the function output) for *any choice* of the public parameters. Among others, a rate-1 scheme implies a maliciously statistically sender-private oblivious transfer [BD18] with the same rate. Previous works [DGI+19] were able to achieve rate 1 only for oblivious transfer and only in the semi-honest setting. The prior best known rate in the malicious setting was $1/2$.

(4) Sealed Bid Auctions: One of the motivating applications of time-lock puzzles is to construct fair sealed bid auctions, where each bid is encrypted in a time-lock puzzle whose opening can be forced by the auctioneer in case the bidder refuses to disclose it. This however involves a computational effort proportional to the number of unopened bids, which can be used as a vector for denial-of-service attacks. Homomorphic time-lock puzzles solve this problem by allowing the auctioneer to homomorphically compute the winner of the auction and only solve a single puzzle. Since this computation cannot be expressed as a linear function, our work provides the first solution from standard assumptions.

1.3 Technical Outline

We present a detailed technical outline of our results in the following. As far as rate-1 FHE is concerned, our focus is on techniques to compress post-evaluation ciphertexts. Compressed ciphertexts can be further expanded (and homomorphically evaluated) via standard bootstrapping techniques.

Schematically, our method for achieving rate-1 FHE is as follows. We consider the "batched-Regev" LWE based encryption scheme (which appears explicitly in the literature, e.g., in [PVW08, BGH13]). This scheme has much better rate than "plain" Regev, but the rate is still asymptotically 0 (i.e., $o(1)$). It can be shown that it is possible to convert plain-Regev ciphertexts into batched-Regev, essentially using the key-switching technique that is frequently used in the FHE literature (see, e.g., [BV11]). We then show that batched-Regev ciphertexts can be compressed in a way that increases the rate to $1 - o(1)$, but maintains (perfect) decryptability. We do this by combining rounding techniques that appeared previously in the literature [DHRW16, BKS19, DGI+19] with new techniques that we develop and allow to maintain high rate, perfect correctness, and modest LWE modulus simultaneously. We note that in order to apply key-switching, we need to use batched-Regev in its non-compressed form, and only apply the compression after the switching is complete. This transformation, maintains decryptability but homomorphic capabilities are lost. As mentioned above, these can be restored using bootstrapping in a generic way.

Leveraging Linear Decryption. Our starting point is the observation that, for essentially any FHE construction in literature, decryption (or rather *noisy decryption*) is a linear function in the secret key. More specifically, we can write the decryption operation as a function $L_{\mathsf{c}}(\mathsf{s})$, which is linear in s, the secret key. Typically things are set up in a way such that it holds for correctly formed ciphertexts c that $L_{\mathsf{c}}(\mathsf{s}) = \frac{q}{2}\ \mathsf{m}+e$, where m is the plaintext and e is a small noise term. We can then recover m from $L_{\mathsf{c}}(\mathsf{s})$ via rounding.

For many FHE schemes, the choice of the factor $q/2$ is not hardwired into the scheme, but can be provided as an explicit input to the decryption function. More specifically, it holds that

$$L\ _{;\mathsf{c}}(\mathsf{s}) = \alpha\ \ \mathsf{m} + e,$$

where $L\ _{;\mathsf{c}}(\ )$ is a linear function and e is a small noise term. Assume in the following that $jej < B$ for some bound B. We refer to this operation as *linear decrypt-and-multiply*. In fact, Micciancio [Mic19] observed that any FHE scheme with linear decryption can be transformed into a scheme which supports linear decrypt-and-multiply.

Equipped with a linear decrypt-and-multiply FHE, our main idea to construct a rate-1 FHE scheme is to run the linear decrypt-and-multiply operation of the FHE scheme inside a high rate linearly homomorphic scheme. Consider an FHE scheme whose secret keys are vectors over Z_q, and a rate-1 linearly homomorphic scheme HE with plaintext space Z_q. Assume we are given as "compression key" the encryption $\mathsf{ck} = \mathsf{Enc}(\mathsf{pk}, \mathsf{s})$ of the FHE secret key s under the linearly homomorphic scheme HE. Given an FHE ciphertext c encrypting a message $\mathsf{m}\ 2\ f0, 1g$, we can transform c into an encryption of m under the linearly homomorphic scheme by homomorphically evaluating the linear function $L\ _{;\mathsf{c}}(\ )$ on ck, i.e. we compute $\mathsf{HE.Eval}(L\ _{;\mathsf{c}}(\ ), \mathsf{ck})$. By homomorphic correctness, this results in an encryption of $\alpha\ \ \mathsf{m} + e$ under the linearly homomorphic scheme HE.

So far, we have not gained anything in terms of rate, as we still have a large ciphertext encrypting only a single bit m. However, we have not yet taken advantage of the fact that we can choose α freely and that the scheme HE has rate 1. Our idea to increase the rate is to *pack* many FHE ciphertexts $(\mathsf{c}_1, \ldots, \mathsf{c}_\ell)$, each encrypting a single bit m_i, into a single ciphertext of the high-rate linearly homomorphic scheme HE. More specifically, for given FHE ciphertexts $(\mathsf{c}_1, \ldots, \mathsf{c}_\ell)$ and a parameter t, consider the function $L\ (x)$ defined as

$$L\ (x) = \dot{\sum_{i=1}} L_{2^{t+i};\mathsf{c}_i}(x).$$

Note that, although we define L as a sum of functions, this is not how we compute it. Since L is a linear function, we can obtain a matrix-representation of it by, e.g., evaluating it on a basis and then later use the matrix representation to compute the function. By correctness of the FHE scheme it holds that

$$\begin{aligned} L\ (\mathsf{s}) &= \dot{\sum_{i=1}} L_{2^{t+i};\mathsf{c}_i}(\mathsf{s}) \\ &= \dot{\sum_{i=1}} 2^{t+i}\ \ \mathsf{m}_i + e, \end{aligned}$$

where $e = \bar{\sum_{i=1}} e_i$ is an ℓB-bounded noise term. Consequently, by homomorphically evaluating L on ck, we obtain an encryption $\tilde{\mathsf{c}}$ of $\bar{\sum_{i=1}} 2^{t+i}\ \ \mathsf{m}_i + e$ under the high-rate scheme HE. Given that $2^t > \ell B$, the noise e does not interfer with the encodings of the message bits m_i and they can be recovered during decryption.

The main effect that works in our favor here is that we can *distribute* the message bits m_i into the high order bits by multiplying them with appropriate powers of 2, whereas the decryption noise *piles up in the low order bits*. Consequently, the noise occupies only the lower $\log(\ell)+\log(B)$ bits, whereas the remaining bits of the message space can be packed with message bits. Choosing q as $q\ \ (\ell B)^{1=}$ for a parameter $\epsilon > 0$ we achieve an encoding rate of $\frac{\log(q)\ \ \log(\ell B)}{\log(q)} = 1\ \ \epsilon$. Given that the linearly homomorphic encryption scheme has a similarly high rate, we obtain an overall rate of $1\ \ O(\epsilon)$. Consequently, this construction yields an FHE scheme with rate $1\ \ O(1/\lambda)$ using, e.g., the Damgård-Jurik cryptosystem or a variant of Regev encryption as linearly homomorphic scheme, where the LWE modulus-to-noise ratio is with (sub-)exponential [PVW08].

Towards a Scheme from Standard LWE. Our next goal is to achieve the same (asymptotic) rate assuming only LWE with polynomial modulus-to-noise ratio. Recall that our packing strategy consisted in encoding the message vector $\mathsf{m} = (\mathsf{m}_1, \ldots, \mathsf{m}_\ell)$ into the high-order bits of a Z_q-element by homomorphically computing $\mathsf{t}^\top \cdot \mathsf{m}$, where $\mathsf{t}^\top = (2^{t+1}, \ldots, 2^{t+\ell})$. However, this is not the only possible strategy. More generally, linear decrypt-and-multiply enables us to homomorphically pack messages $(\mathsf{m}_1, \ldots, \mathsf{m}_\ell)$ into an encoded vector $\mathsf{T} \cdot \mathsf{m}$ for some packing matrix $\mathsf{T} \in \mathsf{Z}_q^{k \times \ell}$. Since linear decryption is inherently noisy, we will require some error correcting properties from such an encoding, i.e., we need to be able to reconstruct m from $\mathsf{T} \cdot \mathsf{m} + \mathsf{e}$, for short noise terms e. With this observation in mind, our next step will be to construct an ad-hoc high-rate linearly homomorphic encryption and pair it with an appropriate packing strategy.

Linearly Homomorphic Encryption with Ciphertext Shrinking. We now discuss new constructions of linearly homomorphic encryption schemes from LWE which allow asymptotically optimal ciphertext sizes. To avoid confusion with our FHE ciphertext compression technique, we will refer to this technique as *ciphertext shrinking*. Our starting point is Regev encryption and its variants. Let q be a modulus. In Regev encryption a ciphertext c consists of two parts, a vector $\mathsf{c}_1 \in \mathsf{Z}_q^n$ and a scalar $\mathsf{c}_2 \in \mathsf{Z}_q$. The secret key is a vector $\mathsf{s} \in \mathsf{Z}_q^n$. Decryption for this scheme is linear, and it holds that

$$\mathsf{c}_2 - \mathsf{s}^\top \cdot \mathsf{c}_1 = \underbrace{\frac{q}{2} \cdot \mathsf{m}}_{\hat{\mathsf{m}}} + e,$$

where e with $|e| < B$ for some bound B is a decryption noise term. We obtain the plaintext m by rounding $\hat{\mathsf{m}}$, i.e., by computing

$$\begin{aligned} \lfloor \hat{\mathsf{m}} \cdot 2/q \rceil &= \left\lfloor \left(\frac{q}{2} \cdot \mathsf{m} + e \right) \cdot 2/q \right\rceil \\ &= \lfloor \mathsf{m} + 2e/q \rceil = \mathsf{m}, \end{aligned}$$

given that $q > 4B$. We first show how to shrink the component c_2 of the ciphertext into a single bit at the expense of including an additional ring element $r \in \mathsf{Z}_q$ in the ciphertext. Although this procedure does not actually shrink the ciphertext (in fact it increases its size by one bit), we will later amortize the cost of r across multiple components. The main idea is to *delegate* a part of the rounding operation from the decrypter to a public operation Shrink and it is inspired by recent works on spooky encryption [DHRW16], homomorphic secret sharing [BKS19], and private-information retrieval [DGI+19].

The algorithm Shrink takes as input the ciphertext $\mathsf{c} = (\mathsf{c}_1, \mathsf{c}_2)$ where $\mathsf{c}_2 \in \mathsf{Z}_q$ and proceeds as follows. It first chooses an $r \in \mathsf{Z}_q$ such that $\mathsf{c}_2 + r \notin [q/4 - B, q/4 + B] \cup [3/4 \cdot q - B, 3/4 \cdot q + B]$, then it computes $w = \lfloor \mathsf{c}_2 + r\mathsf{c}_2 \rceil$ and outputs a compressed ciphertext $\tilde{\mathsf{c}} = (\mathsf{c}_1, r, w)$. Given a shrunk ciphertext $\tilde{\mathsf{c}} = (\mathsf{c}_1, r, w)$ and the secret key s, the decrypter computes

$$\mathsf{m}' = (w - \lfloor \mathsf{s}^\top \mathsf{c}_2 + r\mathsf{c}_2 \rceil) \mod 2.$$

We claim that m' is identical to $\mathsf{Dec}(\mathsf{s}, \mathsf{c}) = \lfloor \mathsf{c}_2 - \mathsf{s}^\top \cdot \mathsf{c}_1 \mathsf{c}_2 \rceil$. To see this, note that since $\mathsf{c}_2 - \mathsf{s}^\top \cdot \mathsf{c}_1 = \frac{q}{2}\mathsf{m} + e$, we can write

$$\mathsf{c}_2 - e = \mathsf{s}^\top \cdot \mathsf{c}_1 + \frac{q}{2} \cdot \mathsf{m}.$$

Now, since r is chosen such that $\mathsf{c}_2 + r \notin [q/4 - B, q/4 + B] \cup [3/4 \cdot q - B, 3/4 \cdot q + B]$ and $e \in [-B, B]$, it holds that

$$\lfloor \mathsf{c}_2 + r\mathsf{c}_2 \rceil = \lfloor \mathsf{c}_2 + r \cdot e\mathsf{c}_2 \rceil.$$

Using the above this implies that

$$w = \lfloor \mathsf{c}_2 + r\mathsf{c}_2 \rceil = \lfloor \mathsf{c}_2 + r \cdot e\mathsf{c}_2 \rceil = \lfloor \mathsf{s}^\top \cdot \mathsf{c}_1 + r + \frac{q}{2} \cdot \mathsf{m}\mathsf{c}_2 \rceil = (\lfloor \mathsf{s}^\top \cdot \mathsf{c}_1 + r\mathsf{c}_2 + \mathsf{m} \rceil) \mod 2.$$

It follows that $\mathsf{m} = (w \quad d\mathsf{s}^{>}\mathsf{c}_2 + r\mathsf{c}_2) \mod 2$. Note that after shrinking ciphertexts, we can no longer perform homomorphic operations (unless one is willing to run a bootstrapped ciphertext expansion). As a consequence, in our applications we will only perform the shrinking operation after all homomorphic operations have been computed.

What is left to be shown is how to amortize the cost of including r by shrinking many c_2 components for the same c_1. To achieve this, instead of using basic Regev encryption, we use *batched* Regev encryption. In batched Regev encryption, ciphertexts consist of a vector $\mathsf{c}_1 \in \mathbb{Z}_q^n$ and ring elements $\mathsf{c}_{2;i} \in \mathbb{Z}_q$ for $i \in [\ell]$. To decrypt the i-th message component m_i, we compute

$$\mathsf{m}_i = d\mathsf{c}_{2;i} \quad \mathsf{s}_i^{>} \quad \mathsf{c}_1\mathsf{c}_2.$$

where s_i is the secret key for the i-th component. Consequently, we can use the same shrinking strategy as above for every $\mathsf{c}_{2;i}$. However, now each $\mathsf{c}_{2;i}$ imposes a constraint on r, namely that $\mathsf{c}_{2;i} + r \notin [q/4 \quad B, q/4 + B] \cup [3/4 \quad q \quad B, 3/4 \quad q + B]$.

Fortunately, given that q is sufficiently large, namely $q > 4\ell B$, there exists an r which fulfills all constraints simultaneously. To find such an r, we compute a union of all *forbidden intervals* modulo q, and pick an r outside of this set. Notice that this procedure can be efficiently implemented even if q is super-polynomially large. The rate of the resulting scheme is

$$\frac{\ell}{(n+1)\log(q) + \ell} = 1 \quad \frac{(n+1)\log(q)}{(n+1)\log(q) + \ell}.$$

For $q \quad 4\ell B$ and a sufficiently large $\ell = \Omega(\lambda \quad (n+1)\log(q)) = \mathsf{poly}(\lambda)$, we achieve rate $1 \quad O(1/\lambda)$.

Notice that while basic Regev encryption is only additively homomorphic, we need a scheme that supports homomorphic evaluation of linear functions. Fortunately, this can be achieved by a very simple modification. Instead of encrypting a message m, encrypt the messages $2^i \quad \mathsf{m}$ for all $i \in [k]$. Further details are deferred to the main body (Section 3.3).

Back to Rate-1 FHE. Returning to our main objective of rate-1 FHE, if we instantiate our generic construction from above with the packed Regev scheme that allows ciphertext shrinking, note that there is a slight mismatch. Recall that our rate 1 FHE construction assumed a linearly homomorphic encryption scheme with plaintext space $\mathbb{Z}_q$ or $\mathbb{Z}_q^k$, whereas our Regev scheme with shrinking has a plaintext space $\{0,1\}^*$.

Towards resolving this issue, it is instructive to consider Regev encryption without message encoding and decryption without rounding. That is, we consider only the linear part of decryption where a ciphertext $\mathsf{c} = (\mathsf{c}_1, \mathsf{c}_2)$ decrypts to

$$\mathsf{Dec}(\mathsf{s}, \mathsf{c}) = \mathsf{c}_2 \quad \mathsf{s}^{>} \quad \mathsf{c}_1 = \mathsf{m} \quad + e'$$

where s is the secret key and the message m is an element of $\mathbb{Z}_q$. The important observation is that in the construction above the message m is the result of a linear decrypt-and-multiply operation. This means that m already contains a certain amount of decryption noise and the actual message contained in m has already been encoded by the linear decrypt-and-multiply operation.

Assuming for simplicity that $\mathsf{m} = L_{\frac{q}{2};\mathsf{c}}(\mathsf{s})$, where c is an FHE ciphertext encrypting a message m and s the corresponding FHE secret key, we have that

$$\begin{aligned}\mathsf{Dec}(\mathsf{s}, \mathsf{c}) = \mathsf{c}_2 \quad \mathsf{s}^{>} \quad \mathsf{c}_1 &= L_{\frac{q}{2};\mathsf{c}}(\mathsf{s}) + e' \\ &= \frac{q}{2} \quad \mathsf{m} + e' + e'',\end{aligned}$$

where e'' is a small noise term which is introduced by the inner FHE decryption. Note that above we only had to deal with noise e'' coming from the inner FHE decryption, whereas now we have an additional noise term e' coming from the decryption of the linearly homomorphic scheme. Given that the *compound noise* $e = e' + e''$ is sufficiently small, our shrinking technique for the ciphertext $(\mathsf{c}_1, \mathsf{c}_2)$ still works. The only condition we need for the shrinking technique to work is that $\mathsf{c}_2 \quad \mathsf{s}^{>} \quad \mathsf{c}_1$ is of the form $\frac{q}{2} \quad \mathsf{m} + e$ for a B-bounded error e.

To sum things up, all we need to ensure is that the encrypted message is well-formed before ciphertext shrinking via the Shrink procedure. To stay with the notation from above, for this scheme the packing matrix T which is used to encode plaintexts during the homomorphic decrypt-and-multiply step will be $\frac{q}{2}\ \mathsf{I}$, where I is the identity matrix.

Fully Homomorphic Time-Lock Puzzles. We finally show how ideas from our rate-1 FHE construction can be used to obtain fully homomorphic time-lock puzzles (FH-TLP) from standard assumptions. Very recently, Malavolta and Thyargarajan [MT19] introduced the notion of homomorphic time-lock puzzles and proposed an efficient construction of linearly homomorphic timelock puzzles (LH-TLP) from the sequential squaring assumption [RSW96]. An LH-TLP allows for evaluations of linear functions on messages encrypted in time-lock puzzles. A key aspect here is that the time-lock puzzles may be independently generated by different players.

The basic idea underlying our construction of FH-TLP is to replace the linearly homomorphic encryption scheme in our rate-1 FHE construction above by an LH-TLP. More concretely, fix and LH-TLP scheme where the message-space is Z_q and an FHE scheme for which the secret keys are Z_q^n vectors. We will describe how to generate a puzzle for a message m and time parameter T. First, generate an FHE public key pk together with a secret key $\mathsf{s}\ 2\ \mathsf{Z}_q^n$. Next, create a puzzle Z with time parameter T for the LH-TLP scheme encrypting the FHE secret key s. Finally, encrypt the message m under the FHE public key pk obtaining a ciphertext c. The time-lock puzzle consists of $(\mathsf{pk}, \mathsf{c}, \mathsf{Z})$ and can be solved by recovering the secret key s and then decrypting the message m.

While this simple idea allows us to perform homomorphic computations on a single message m, it fails at our actual goal of allowing homomorphic computations on puzzles generated by different puzzle generators. The reason being that every time we generate a new puzzle, we generate a fresh FHE key, and generally homomorphic computations across different keys are not possible. To overcome this issue, we instead use a multi-key FHE scheme, which enables homomorphic computations across different public keys. More specifically, given ℓ puzzles $(\mathsf{pk}_1, \mathsf{c}_1, \mathsf{Z}_1), \ldots, (\mathsf{pk}_\ell, \mathsf{c}_\ell, \mathsf{Z}_\ell)$, encrypting messages $(m_1, \ldots, m_\ell)$, and an ℓ input function f, we can homomorpically compute a ciphertext $\mathsf{c}\ = \mathsf{Eval}(\mathsf{pk}_1, \ldots, \mathsf{pk}_\ell, f, (\mathsf{c}_1, \ldots, \mathsf{c}_\ell))$ which encrypts the message $\mathsf{m}\ = f(\mathsf{m}_1, \ldots, \mathsf{m}_\ell)$.

We have, however, still not solved the main problem. In order to recover $f(\mathsf{m}_1, \ldots, \mathsf{m}_\ell)$ from c, we first have to recover all secret keys $(\mathsf{s}_1, \ldots, \mathsf{s}_\ell)$ from the LH-TLPs $(\mathsf{Z}_1, \ldots, \mathsf{Z}_\ell)$. Thus, the workload is proportional to that of solving ℓ time-lock puzzles, which is identical to the trivial construction. The final idea is to use a multi-key FHE scheme with linear decryption: If c is a (homomorphically evaluated) ciphertext which encrypts a message m under public keys $\mathsf{pk}_1, \ldots, \mathsf{pk}_\ell$, we can decrypt c using a function $L_{\mathsf{c}}\ (\mathsf{s}_1, \ldots, \mathsf{s}_\ell)$ which is linear in the secret keys $\mathsf{s}_1, \ldots, \mathsf{s}_\ell$. As before, this decryption operation is noisy, i.e.,

$$L_{\mathsf{c}}\ (\mathsf{s}_1, \ldots, \mathsf{s}_\ell) = \frac{q}{2}\ \ \mathsf{m}\ \ + e,$$

where e with $|e| < B$ is a small noise term. This allows us to homomorphically evaluate the linear function L_{c} over the time-lock puzzles $(\mathsf{Z}_1, \ldots, \mathsf{Z}_\ell)$ (recall the Z_i encrypts the secret key s_i) and obtain a time-lock puzzle $\mathsf{Z}\ = \mathsf{Eval}(L_{\mathsf{c}}\ , (\mathsf{Z}_1, \ldots, \mathsf{Z}_\ell))$ encrypting $L_{\mathsf{c}}\ (\mathsf{s}_1, \ldots, \mathsf{s}_\ell) = \frac{q}{2}\ \ \mathsf{m}\ \ + e$. To recover the computation result m we only have to solve Z. Note that the final puzzle Z is a single compact puzzle for the LH-TLP scheme, thus the overhead to solve this puzzle is that of solving a single LH-TLP and therefore independent of ℓ.

We remark that both multi-key FHE from standard assumptions [CM15, MW16] and LH-TLP from standard assumptions [MT19] need a setup. Consequently, our FH-TLP construction inherits this property. Finally, techniques that we develop to construct rate-1 FHE also apply to our FH-TLP construction.

2 Preliminaries

We denote by $\lambda\ 2\ \mathsf{N}$ the security parameter. We we say that a function $\mathsf{negl}(\ )$ is negligible if it vanishes faster than any polynomial. Given a set S, we denote by $s\ \ _{\$}S$ the uniform sampling from S. We say that an algorithm is PPT if it can be implemented by a probabilistic machine running in time $\mathsf{poly}(\lambda)$. We

abbreviate the set $f1, \ldots, ng$ as $[n]$. Matrices are denoted by M and vectors are denoted by v. We use the infinity norm of a vector $k\mathsf{v}k_1$, since it behaves conveniently with rounding. For a given modulus q, we define the rounding function $d x c_2 = d x \ \ 2/q c \mod 2$.

2.1 Learning with Errors

The (decisional) learning with errors (LWE) problem was introduced by Regev [Reg05]. The LWE problem is parametrized by a modulus q, positive integers n, m and an error distribution χ. An adversary is either given $(\mathsf{A}, \mathsf{s}^> \ \mathsf{A} + \mathsf{e})$ or (A, u) and has to decide which is the case. Here, A is chosen uniformly from $\mathsf{Z}_q^{n \ m}$, s is chosen uniformly from Z_q^n, u is chosen uniformly from Z_q^m and e is chosen from χ^m. The matrix version of this problem asks to distinguish $(\mathsf{A}, \mathsf{S} \ \ \mathsf{A} + \mathsf{E})$ from (A, U), where the dimensions are accordingly. It follows from a simple hybrid argument that the matrix version is as hard as the standard version.

As shown in [Reg05, PRS17], for *any* sufficiently large modulus q the LWE problem where χ is a discrete Gaussian distribution with parameter $\sigma = \alpha q \quad 2\sqrt{n}$ (i.e. the distribution over Z where the probability of x is proportional to $e^{\ (jxj= \ )^2}$), is at least as hard as approximating the shortest independent vector problem (SIVP) to within a factor of $\gamma = \tilde{O}(n/\alpha)$ in *worst case* dimension n lattices. We refer to $\alpha = \sigma/q$ as the *modulus-to-noise* ratio, and by the above this quantity controls the hardness of the LWE instantiation. Hereby, LWE with polynomial α is (presumably) harder than LWE with super-polynomial or sub-exponential α. We can truncate the discrete gaussian distribution χ to $\sigma \ \omega(\sqrt{\log(\lambda)})$ while only introducing a negligible error. Consequently, we omit the actual distribution χ but only use the fact that it can be appropriately bounded by a (small) bound B.

2.2 Homomorphic Encryption

We recall the definition of homomorphic encryption in the following.

Definition 2.1 (Homomorphic Encryption). *A homomorphic encryption scheme consists of the following efficient algorithms.*

$\mathsf{KeyGen}(1^\lambda)$: *On input the security parameter* 1^λ*, the key generation algorithm returns a key pair* $(\mathsf{sk}, \mathsf{pk})$.

$\mathsf{Enc}(\mathsf{pk}, \mathsf{m})$: *On input a public key* pk *and a message* m*, the encryption algorithm returns a ciphertext* c.

$\mathsf{Eval}(\mathsf{pk}, f, (\mathsf{c}_1, \ldots, \mathsf{c}_\ell))$: *On input the public key* pk*, an* ℓ*-argument function* f*, and a vector of ciphertexts* $(\mathsf{c}_1, \ldots, \mathsf{c}_\ell)$*, the evaluation algorithm returns an evaluated ciphertext* c.

$\mathsf{Dec}(\mathsf{sk}, \mathsf{c})$: *On input the secret key* sk *and a ciphertext* c*, the decryption algorithm returns a message* m.

We say that a scheme is fully-homomorphic (FHE) if it is homomorphic for all polynomial-size circuits. We also consider a restricted class of homomorphism that supports linear functions and we refer to such a scheme as linearly-homomorphic encryption. We characterize correctness of a single function evaluation. This can be extended to the more general notion of multi-hop correctness [GHV10] if the condition specified below is required to hold for arbitrary compositions of functions.

Definition 2.2 (Correctness). *A homomorphic encryption scheme* $(\mathsf{KeyGen}, \mathsf{Enc}, \mathsf{Eval}, \mathsf{Dec})$ *is correct if for all* $\lambda \ 2 \ \mathsf{N}$*, all* ℓ*-argument functions* f *in the supported family, all inputs* $(\mathsf{m}_1, \ldots, \mathsf{m}_\ell)$*, all* $(\mathsf{sk}, \mathsf{pk})$ *in the support of* $\mathsf{KeyGen}(1^\lambda)$*, and all* c_i *in the support of* $\mathsf{Enc}(\mathsf{pk}, \mathsf{m}_i)$ *there exists a negligible function* $\mathsf{negl}(\ )$ *such that*

$$\Pr\left[\mathsf{Dec}(\mathsf{sk}, \mathsf{Eval}(\mathsf{pk}, f, (\mathsf{c}_1, \ldots, \mathsf{c}_\ell))) = f(\mathsf{m}_1, \ldots, \mathsf{m}_\ell)\right] \quad 1 \quad \mathsf{negl}(\lambda)\,.$$

We require a scheme to be compact in the sense that the size of the ciphertext should not grow with the size of the evaluated function.

Definition 2.3 (Compactness). *A homomorphic encryption scheme* $(\mathsf{KeyGen}, \mathsf{Enc}, \mathsf{Eval}, \mathsf{Dec})$ *is compact if there exists a polynomial* $\mathsf{poly}(\cdot)$ *such that for all* $\lambda \in \mathbb{N}$*, all* ℓ*-argument functions* f *in the supported family, all inputs* $(\mathsf{m}_1, \ldots, \mathsf{m}_\ell)$*, all* $(\mathsf{sk}, \mathsf{pk})$ *in the support of* $\mathsf{KeyGen}(1^\lambda)$*, and all* c_i *in the support of* $\mathsf{Enc}(\mathsf{pk}, \mathsf{m}_i)$ *it holds that*

$$|\mathsf{Eval}(\mathsf{pk}, f, (\mathsf{c}_1, \ldots, \mathsf{c}_\ell))| = \mathsf{poly}(\lambda, |f(\mathsf{m}_1, \ldots, \mathsf{m}_\ell)|)\,.$$

The notion of security is standard for public-key encryption [GM82].

Definition 2.4 (Semantic Security). *A homomorphic encryption scheme* $(\mathsf{KeyGen}, \mathsf{Enc}, \mathsf{Eval}, \mathsf{Dec})$ *is semantically secure if for all* $\lambda \in \mathbb{N}$ *and for all PPT adversaries* $\mathcal{A} = (\mathcal{A}_0, \mathcal{A}_1)$ *there exists a negligible function* $\mathsf{negl}(\cdot)$ *such that*

$$\Pr\left[b = \mathcal{A}_1(\mathsf{c}, \mathsf{st}) \;\middle|\; \begin{array}{l} (\mathsf{sk}, \mathsf{pk}) \leftarrow \mathsf{KeyGen}(1^\lambda) \\ (\mathsf{m}_0, \mathsf{m}_1, \mathsf{st}) \leftarrow \mathcal{A}_0(\mathsf{pk}) \\ b \xleftarrow{\$} \{0,1\} \\ \mathsf{c} \leftarrow \mathsf{Enc}(\mathsf{pk}, \mathsf{m}_b) \end{array}\right] = \frac{1}{2} + \mathsf{negl}(\lambda)\,.$$

Finally we define the rate of an encryption scheme as the asymptotic message-to-ciphertext size ratio.

Definition 2.5 (Rate). *We say that a homomorphic encryption scheme* $(\mathsf{KeyGen}, \mathsf{Enc}, \mathsf{Eval}, \mathsf{Dec})$ *has rate* $\rho = \rho(\lambda)$*, if it holds for all* pk *in the support of* $\mathsf{KeyGen}(1^\lambda)$*, all supported functions* f *with sufficiently large output size, all messages* $(\mathsf{m}_1, \ldots, \mathsf{m}_\ell)$ *in the message space, and all* c_i *in the support of* $\mathsf{Enc}(\mathsf{pk}, \mathsf{m}_i)$ *that*

$$\frac{|f(\mathsf{m}_1, \ldots, \mathsf{m}_\ell)|}{|\mathsf{Eval}(\mathsf{pk}, f, (\mathsf{c}_1, \ldots, \mathsf{c}_\ell))|} \geq \rho.$$

We also say that a scheme has rate 1, if it holds that

$$\liminf_{\lambda \in \mathbb{N}} \rho(\lambda) = 1.$$

Note that in Definition 2.5 we need to restrict ourselves to a class of supported functions for which the output size $|f(\mathsf{m}_1, \ldots, \mathsf{m}_\ell)|$ is sufficiently large. E.g., if a function output $f(\mathsf{m}_1, \ldots, \mathsf{m}_\ell)$ is just one bit, we cannot hope to achieve a good rate. Consequently we will only consider functions with a large output domain.

2.2.1 Multi-Key Homomorphic Encryption

A multi-key homomorphic encryption supports the evaluation of functions over ciphertexts computed under different (possibly independently sampled) keys. The result of the computation can then be decrypted using all of the corresponding secret keys. Formally, this introduces a few syntactical modifications. Most notably and in contrast with the single-key variant, multi-key schemes might need a setup which generates public parameters shared across all users.

Definition 2.6 (Multi-Key Homomorphic Encryption). *A multi-key homomorphic encryption scheme consists of the following efficient algorithms.*

$\mathsf{Setup}(1^\lambda)$: *On input the security parameter* 1^λ*, the setup algorithm returns the public parameters* pp*.*

$\mathsf{KeyGen}(\mathsf{pp})$: *On input the public parameters* pp*, the key generation algorithm returns a key pair* $(\mathsf{sk}, \mathsf{pk})$*.*

$\mathsf{Enc}(\mathsf{pk}, \mathsf{m})$: *On input a public key* pk *and a message* m*, the encryption algorithm returns a ciphertext* c*.*

$\mathsf{Eval}((\mathsf{pk}_1, \ldots, \mathsf{pk}_\ell), f, (\mathsf{c}_1, \ldots, \mathsf{c}_\ell))$: *On input a vector of public keys* $(\mathsf{pk}_1, \ldots, \mathsf{pk}_\ell)$*, an* ℓ*-argument function* f*, and a vector of ciphertexts* $(\mathsf{c}_1, \ldots, \mathsf{c}_\ell)$*, the evaluation algorithm returns an evaluated ciphertext* c*.*

$\mathsf{Dec}((\mathsf{sk}_1, \ldots, \mathsf{sk}_\ell), \mathsf{c})$: *On input a vector of secret keys* $(\mathsf{sk}_1, \ldots, \mathsf{sk}_\ell)$ *and a ciphertext* c*, the decryption algorithm returns a message* m*.*

As before, we say that the scheme is fully-homomorphic (MK-FHE) if it is homomorphic for P/poly. The definition of correctness is adapted to the multi-key settings.

Definition 2.7 (Multi-Key Correctness). *A multi-key homomorphic encryption scheme* (Setup, KeyGen, Enc, Eval, Dec) *is correct if for all* $\lambda \in \mathbb{N}$*, all* ℓ *polynomial in* λ*, all* ℓ*-argument functions* f *in the supported family, all inputs* $(\mathsf{m}_1, \ldots, \mathsf{m}_\ell)$*, all* pp *in the support of* Setup*, all* $(\mathsf{sk}_i, \mathsf{pk}_i)$ *in the support of* $\mathsf{KeyGen}(\mathsf{pp})$*, and all* c_i *in the support of* $\mathsf{Enc}(\mathsf{pk}_i, \mathsf{m}_i)$ *there exists a negligible function* $\mathsf{negl}(\cdot)$ *such that*

$$\Pr\left[\mathsf{Dec}((\mathsf{sk}_1, \ldots, \mathsf{sk}_\ell), \mathsf{Eval}((\mathsf{pk}_1, \ldots, \mathsf{pk}_\ell), f, (\mathsf{c}_1, \ldots, \mathsf{c}_\ell))) = f(\mathsf{m}_1, \ldots, \mathsf{m}_\ell)\right] \quad 1 \quad \mathsf{negl}(\lambda)\,.$$

Compactness is unchanged except that the ciphertext may grow with the number of keys.

Definition 2.8 (Multi-Key Compactness). *A multi-key homomorphic encryption scheme* (Setup, KeyGen, Enc, Eval, Dec) *is compact if there exists a polynomial* $\mathsf{poly}(\cdot)$ *such that for all* $\lambda \in \mathbb{N}$*, all* ℓ *polynomial in* λ*, all* ℓ*-argument functions* f *in the supported family, all inputs* $(\mathsf{m}_1, \ldots, \mathsf{m}_\ell)$*, all* $(\mathsf{sk}_i, \mathsf{pk}_i)$ *in the support of* $\mathsf{KeyGen}(1^\lambda)$*, and all* c_i *in the support of* $\mathsf{Enc}(\mathsf{pk}_i, \mathsf{m}_i)$ *it holds that*

$$|\mathsf{Eval}((\mathsf{pk}_1, \ldots, \mathsf{pk}_\ell), f, (\mathsf{c}_1, \ldots, \mathsf{c}_\ell))| = \mathsf{poly}(\lambda, \ell, |f(\mathsf{m}_1, \ldots, \mathsf{m}_\ell)|)\,.$$

The definition of semantic security is identical to that of single-key schemes.

2.2.2 Linear Decrypt-and-Multiply

To construct our schemes we will need FHE schemes with a more fine-grained correctness property. More specifically, we will require an FHE scheme where for which decryption is a linear function in the secret key. Furthermore, we require that this linear decryption function outputs a the product of the plaintext whith a constant ω (which is provided as input to the decryption algorithm). We will refer to such schemes as linear decrypt-and-multiply schemes.

The output of this function may contain some (short) noise, thus we also need an upper bound on amount of noise linear decrypt-and-multiply introduces. This property was explicitly characterized in an oral presentation of Micciancio [Mic19] where he showed that schemes from the literature already satisfy this notion [AP14, GSW13] and discussed some applications. A formal definition is given in the following.

Definition 2.9 (Decrypt-and-Multiply). *We call a homomorphic encryption scheme* (KeyGen, Enc, Eval, Dec) *a decrypt-and-multiply scheme, if there exists bounds* $B = B(\lambda)$ *and* $Q = Q(\lambda)$ *and an algorithm* Dec&Mult *such that the following holds. For every* $q \quad Q$*, all* $(\mathsf{sk}, \mathsf{pk})$ *in the support of* $\mathsf{KeyGen}(1^\lambda, q)$*, every* ℓ*-argument functions* f *(in the class supported by the scheme), all inputs* $(\mathsf{m}_1, \ldots, \mathsf{m}_\ell)$*, all* c_i *in the support of* $\mathsf{Enc}(\mathsf{pk}, \mathsf{m}_i)$ *and every* $\omega \in \mathbb{Z}_q$ *that*

$$\mathsf{Dec\&Mult}(\mathsf{sk}, \mathsf{Eval}(\mathsf{pk}, f, (\mathsf{c}_1, \ldots, \mathsf{c}_\ell)), \omega) = \omega \quad f(\mathsf{m}_1, \ldots, \mathsf{m}_\ell) + e \mod q$$

where Dec&Mult *is a linear function in* sk *over* $\mathbb{Z}_q$ *and* $|e| \quad B$ *with all but negligible probability.*

We also consider decrypt-and-multiply for multi-key schemes and we extend the definition below. We note that schemes with such a property were previously considered in the context of Spooky Encryption [DHRW16].

Definition 2.10 (Multi-Key Decrypt-and-Multiply). *We call a multi-key homomorphic encryption scheme* (Setup, KeyGen, Enc, Eval, Dec) *a decrypt-and-multiply scheme, if there exists bounds* $B = B(\lambda)$ *and* $Q = Q(\lambda)$ *and an algorithm* Dec&Mult *such that the following holds. For every* $q \quad Q$*, all* pp *in the support of* $\mathsf{Setup}(1^\lambda; q)$*, all* $(\mathsf{sk}_i, \mathsf{pk}_i)$ *in the support of* $\mathsf{KeyGen}(1^\lambda)$*, every* ℓ*-argument functions* f *(in the class supported by the scheme), all inputs* $(\mathsf{m}_1, \ldots, \mathsf{m}_\ell)$*, all* c_i *in the support of* $\mathsf{Enc}(\mathsf{pk}_i, \mathsf{m}_i)$ *and every* $\omega \in \mathbb{Z}_q$ *that*

$$\mathsf{Dec\&Mult}((\mathsf{sk}_1, \ldots, \mathsf{sk}_\ell), \mathsf{Eval}((\mathsf{pk}_1, \ldots, \mathsf{pk}_\ell), f, (\mathsf{c}_1, \ldots, \mathsf{c}_\ell)), \omega) = \omega \quad f(\mathsf{m}_1, \ldots, \mathsf{m}_\ell) + e \mod q$$

where Dec&Mult *is a linear function in the vector* $(\mathsf{sk}_1, \ldots, \mathsf{sk}_\ell)$ *over* $\mathbb{Z}_q$ *and* $|e| \quad B$ *with all but negligible probability.*

An aspect we have omitted so far is to specify over which domain we require decryption to be linear. For essentially all FHE schemes in the literature, decryption is a linear function over a ring Z_q, which also requires that secret keys are vectors over Z_q. As mentioned before, the main idea behind our constructions will be to perform linear decrypt-and-multiply under a linearly homomorphic encryption scheme. Consequently, we need to match the plaintext space of the linearly homomorphic scheme with the secret key-space of the fully homomorphic scheme. As for some linearly homomorphic schemes we consider, such as the Damgård-Jurik scheme the plaintext space will depend on the public key, we will need a way to connect the two. Luckily, for essentially all FHE schemes in the literature, the modulus q does not depend on any secret but depends only on the security parameter. Moreover, LWE-based FHE schemes can be instantiated with any (sufficiently large) modulus q without affecting the worst-case hardness of the underlying LWE problem [PRS17].

Consequently, we can consider the modulus q as a system parameter for the underlying FHE scheme. In abuse of notation, we will provide the modulus q as an explicit input to the FHE key generation algorithm.

2.2.3 Schemes with Linear Decrypt-and-Multiply

Micciancio [Mic19] has recently shown that any FHE scheme with linear decryption always admits an efficient linear decrypt-and-multiply algorithm. Notable examples of constructions that support linear decrypt-and-multiply right away are GSW-based schemes [GSW13], e.g., [AP14, BV14, CM15, MW16, DHRW16].

In these schemes, ciphertexts are of the form $\mathsf{C} = \mathsf{A}\ \ \mathsf{R} + \mathsf{m}\ \ \mathsf{G}$, where $\mathsf{A} \in \mathsf{Z}_q^{n\ \ m}$ is a matrix specified in the public key, R is a matrix with *small* entries and G is the so-called gadget matrix. The secret key is a vector s, for which the last component $s_n = 1$, which has the property that $\mathsf{s}^{>}\ \ \mathsf{A} = \mathsf{e}^{>}$, for a vector $\mathsf{e}^{>}$ with small entries. For a vector v let $\mathsf{G}\ \ ^{1}(\mathsf{v})$ be a binary vector with the property that $\mathsf{G}\ \ \mathsf{G}\ \ ^{1}(\mathsf{v}) = \mathsf{v}$ ($\mathsf{G}\ \ ^{1}(\ )$ is a non-linear function). For an $\omega \in \mathsf{Z}_q$ let $\ \ \in \mathsf{Z}_q^n$ be a vector which is 0 everywhere but ω in the last component. We can perform the linear decrypt-and-multiply operation by computing

$$\begin{aligned}
\mathsf{s}^{>}\ \ \mathsf{C}\ \ \mathsf{G}\ \ ^{1}(\ ) &= \mathsf{s}^{>}\ \ \mathsf{A}\ \ \mathsf{R}\ \ \mathsf{G}\ \ ^{1}(\ ) + \mathsf{m}\ \ \mathsf{s}^{>}\ \ \mathsf{G}\ \ \mathsf{G}\ \ ^{1}(\ ) \\
&= \mathsf{e}^{>}\ \ \mathsf{R}\ \ \mathsf{G}\ \ ^{1}(\ ) + \mathsf{m}\ \ \mathsf{s}^{>}\ \ \ \\
&= \omega\ \ \mathsf{m} + e^{0},
\end{aligned}$$

where $e^{0} = \mathsf{e}^{>}\ \ \mathsf{R}\ \ \mathsf{G}\ \ ^{1}(\ )$ is a short noise vector. The second equality holds as $\mathsf{s}^{>}\ \ \mathsf{A} = \mathsf{e}^{>}$, and the third one holds as $\mathsf{s}^{>}\ \ \ = \omega$. We remark that the scheme of Brakerski and Vaikunthanatan [BV14] satisfies these constraints with a polynomial modulus-to-noise ratio, by exploiting the asymmetric noise growth in the GSW scheme and a specific way to homomorphically evaluate functions.

Since we need a multi-key FHE scheme in our construction of fully homomorphic time-lock puzzles, we briefly discuss a linear decrypt-and-multiply procedure for the MK-FHE construction of Mukherjee and Wichs [MW16], which in turn is a simplified version of the scheme from Clear and McGoldrick [CM15]. Recall that the scheme shown in [CM15, MW16] is secure against the Learning with Errors problem (with super-polynomial modulo-to-noise ratio) and satisfies the following properties:

(1) The construction is in the common random string model and all parties have access to a uniform matrix $\mathsf{A}\ \ \ _{\$}\mathsf{Z}_q^{(n\ \ 1)\ \ m}$.

(2) For any fixed depth parameter d, the scheme supports multi-key evaluation of depth-d circuits using public keys of size $d\ \ \mathsf{poly}(\lambda)$, while secret keys are vectors $\mathsf{s}\ \ \ _{\$}\mathsf{Z}_q^n$, regardless of the depth parameter. More concretely, there exists an efficient algorithm $\mathsf{MK\text{-}FHE.Eval}$ that is given as input:

   (a) Parameters $(\ell, d) \in \mathsf{N}$, where ℓ is the number of public keys that perform depth-d computation.

   (b) A depth-d circuit that computes an ℓ-argument Boolean function $f : \{0,1\}\ \ \ \ \{0,1\}$.

   (c) A vector of public keys $(\mathsf{pk}_1, \ldots, \mathsf{pk}_\ell)$ and a fresh (bit-by-bit) encryption of each argument x_i under pk_i, denoted by $\mathsf{c}_i\ \ \ \mathsf{MK\text{-}FHE.Enc}(\mathsf{pk}_i, x_i)$.

Then MK-FHE.Eval outputs a matrix $\mathbf{C} \in \mathbb{Z}_q^{n\ell \times m\ell}$ such that

$$\tilde{\mathbf{s}} \cdot \mathbf{C} \cdot \mathbf{G}^{-1}(\mathbf{I}) = \omega \cdot f(x_1, \ldots, x_\ell) + e \pmod{q}$$

where $\tilde{\mathbf{s}}$ is the row concatenation of $(\mathbf{s}_1, \ldots, \mathbf{s}_\ell)$, $\mathbf{I}$ is the row vector $(0, \ldots, 0, \omega) \in \mathbb{Z}_q^{n\ell}$, and $\mathbf{G}^{-1}$ is the bit-decomposition operator. Furthermore, it holds that

$$\|e\| \leq \beta \cdot (m^4 + m)(m\ell + 1)^d = \beta \cdot 2^{O(d \log(\lambda))}$$

where β is a bound on the absolute value of the noise of fresh ciphertexts.

(3) By further making a circular-security assumption, MK-FHE.Eval supports the evaluation of circuits of any depth without increasing the size of the public keys. In this case the bound on the noise is $\|e\| \leq \beta \cdot 2^{O(d_{\mathsf{Dec}} \log(\lambda))}$, where d_{Dec} is the depth of the decryption circuit, which is poly-logarithmic in λ.

Note that that by setting $\ell = 1$ we recover the FHE scheme of [GSW13] except that for the latter we can give a slightly better bound for the noise, namely $\|e\| \leq \beta \cdot m^2(m+1)^d$. The important observation here is that $\mathbf{C} \cdot \mathbf{G}^{-1}(\mathbf{I})$ does not depend on the secret key and therefore defining

$$\mathsf{Dec\&Mult}(\tilde{\mathbf{s}}, \mathbf{C}, \omega) = \tilde{\mathbf{s}} \cdot \mathbf{C} \cdot \mathbf{G}^{-1}(\mathbf{I})$$

gives a syntactically correct linear decrypt-and-multiply algorithm and $B = \|e\|$ is the corresponding noise bound. Finally we remark that the MK-FHE scheme does not impose any restriction on the choice of q (except for its size) so we can freely adjust it to match the modulus of the companion time-lock puzzle.

2.3 Homomorphic Time-Lock Puzzles

Homomorphic time-lock puzzles generalize the classical notion of time-lock puzzles [RSW96] by allowing one to publicly manipulate puzzles to evaluate functions over the secrets. They were introduced in a recent work [MT19] and we recall the definition in the following.

Definition 2.11 (Homomorphic Time-Lock Puzzles). *A homomorphic time-lock puzzle consists of the following efficient algorithms.*

$\mathsf{Setup}(1^\lambda, T)$: *On input the security parameter 1^λ and a time parameter T, the setup algorithm returns the public parameters* pp.

$\mathsf{PuzGen}(\mathsf{pp}, \mathsf{s})$: *On input the public parameters* pp *and a secret* s*, the puzzle generation algorithm returns a puzzle* Z.

$\mathsf{Eval}(\mathsf{pp}, f, (\mathsf{Z}_1, \ldots, \mathsf{Z}_\ell))$: *On input the public parameters* pp*, an ℓ-argument function f, and a vector of puzzles $(\mathsf{Z}_1, \ldots, \mathsf{Z}_\ell)$, the evaluation algorithm returns an evaluated puzzle* Z.

$\mathsf{Solve}(\mathsf{pp}, \mathsf{Z})$: *On input the public parameters* pp *and a puzzle* Z*, the solving algorithm returns a secret* s.

By convention, we refer to a puzzle as fully-homomorphic (FHTLP) if it is homomorphic for all circuits. We now give the definition of (single-hop) correctness.

Definition 2.12 (Correctness). *A homomorphic time-lock puzzle* $(\mathsf{Setup}, \mathsf{PuzGen}, \mathsf{Eval}, \mathsf{Solve})$ *is correct if for all $\lambda \in \mathbb{N}$, all $T \in \mathbb{N}$, all ℓ-argument functions f in the supported family, all inputs $(\mathsf{s}_1, \ldots, \mathsf{s}_\ell)$, all* pp *in the support of* $\mathsf{Setup}(1^\lambda, T)$*, and all Z_i in the support of* $\mathsf{PuzGen}(\mathsf{pp}, \mathsf{s}_i)$ *the following two conditions are satisfied:*

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

$$\Pr\left[\mathsf{Solve}(\mathsf{pp}, \mathsf{Eval}(\mathsf{pp}, f, (\mathsf{Z}_1, \ldots, \mathsf{Z}_\ell))) = f(\mathsf{s}_1, \ldots, \mathsf{s}_\ell)\right] = 1 - \mathsf{negl}(\lambda).$$

(2) The runtime of $\mathsf{Solve}(\mathsf{pp}, \mathsf{Z})$*, where* $\mathsf{Z} \quad \mathsf{Eval}(\mathsf{pp}, f, (\mathsf{Z}_1, \ldots, \mathsf{Z}_\ell))$*, is bounded by* $\mathsf{poly}(\lambda, T)$*, for some fixed polynomial* $\mathsf{poly}(\ )$*.*

Homomorphic time-lock puzzles should also satisfy the notion of compactness. To rule out trivial schemes, we specifically require that the running time of the evaluation algorithm does not depend on the hardness of the puzzle.

De nition 2.13 (Compactness). *A homomorphic time-lock puzzle* $(\mathsf{Setup}, \mathsf{PuzGen}, \mathsf{Eval}, \mathsf{Solve})$ *is compact if there exist two polynomials* $\mathsf{poly}(\ )$ *and* $\tilde{\mathsf{poly}}(\ )$ *such that for all* $\lambda \ 2 \ \mathsf{N}$*, all* $T \ 2 \ \mathsf{N}$*, all* ℓ*-argument functions* f *in the supported family, all inputs* $(\mathsf{s}_1, \ldots, \mathsf{s}_\ell)$*, all* pp *in the support of* $\mathsf{Setup}(1 \ , T)$*, and all* Z_i *in the support of* $\mathsf{PuzGen}(\mathsf{pp}, \mathsf{s}_i)$ *the following two conditions are satisfied:*

(1) $j\mathsf{Eval}(\mathsf{pp}, f, (\mathsf{Z}_1, \ldots, \mathsf{Z}_\ell))j = \mathsf{poly}(\lambda, jf(\mathsf{s}_1, \ldots, \mathsf{s}_\ell)j)$*.*

(2) The runtime of $\mathsf{Eval}(\mathsf{pp}, f, (\mathsf{Z}_1, \ldots, \mathsf{Z}_\ell))$ *is bounded by* $\tilde{\mathsf{poly}}(\lambda, jfj)$*, where* jfj *is the size of the circuit representation of* f*.*

In this work we consider the stronger notion of security where time is counted starting from the moment the puzzle is generated (as opposed to the moment where the public parameters of the scheme are generated). This is termed security with reusable setup in [MT19] and we henceforth refer to it simply as security.

De nition 2.14 (Security). *A homomorphic time-lock puzzle* $(\mathsf{Setup}, \mathsf{PuzGen}, \mathsf{Eval}, \mathsf{Solve})$ *is secure if for all* $\lambda \ 2 \ \mathsf{N}$*, all* $T \ 2 \ \mathsf{N}$*, all PPT adversaries* $\mathsf{A} = (\mathsf{A}_0, \mathsf{A}_1)$ *such that the depth of* A_1 *is bounded by* T*, there exists a negligible function* $\mathsf{negl}(\ )$ *such that*

$$\Pr\left[\begin{array}{l} 2 \\ 6 \\ 4 \end{array} b = \mathsf{A}_1(\mathsf{Z}, \mathsf{st}) \;\middle|\; \begin{array}{l} \mathsf{pp} \quad \mathsf{Setup}(1 \ , T) \\ (\mathsf{s}_0, \mathsf{s}_1, \mathsf{st}) \quad \mathsf{A}_0(\mathsf{pp}) \\ b \quad {}_{\$} \ f0, 1g \\ \mathsf{Z} \quad \mathsf{PuzGen}(\mathsf{pp}, \mathsf{s}_b) \end{array} \;\middle|\; \begin{array}{l} 3 \\ 7 \\ 7 \\ 5 \end{array} = \frac{1}{2} + \mathsf{negl}(\lambda)\,.\right.$$

3 Shrinking Linearly Homomorphic Encryption

In the following section we introduce the useful abstraction of linearly homomorphic encryption with compressing ciphertexts and we discuss several concrete instantiations.

3.1 Definitions

We start by providing relaxed correctness definitions for linearly homomorphic encryption. As discussed before, for Regev-like encryption schemes decryption is a linear operation which, unavoidably, introduces noise. This noise is dealt with by encoding the message accordingly and decoding the result of linear decryption, usually by applying a rounding function. In this section we provide definitions for linearly homomorphic encryption which account for noise, and allow to treat encoding and decoding of the message separately. We assume that a linearly homomorphic encryption scheme is described by four algorithms $(\mathsf{KeyGen}, \mathsf{Enc}, \mathsf{Dec}, \mathsf{Eval})$ with the usual syntax. We further assume that each public key pk specifies a message space of the form Z_q^k.

De nition 3.1 (Relaxed Correctness). *Let* $\mathsf{HE} = (\mathsf{KeyGen}, \mathsf{Enc}, \mathsf{Dec}, \mathsf{Eval})$ *be a linearly homomorphic encryption scheme. Let* $B = B(\lambda)$ *and* $\ell = \mathsf{poly}(\lambda)$*. We say that* HE *is correct with* B*-noise, if it holds for every* $(\mathsf{pk}, \mathsf{sk})$ *in the support of* $\mathsf{KeyGen}(1 \ )$*, where* pk *specifies a message space* Z_q^k*, every linear function* $f : (\mathsf{Z}_q^k)^{\ell} \ ! \ \mathsf{Z}_q^k$*, all messages* $(\mathsf{m}_1, \ldots, \mathsf{m}_\ell) \ 2 \ \mathsf{Z}_q^k$ *that*

$$\mathsf{Dec}(\mathsf{sk}, \mathsf{Eval}(\mathsf{pk}, f, (\mathsf{Enc}(\mathsf{pk}, \mathsf{m}_1), \ldots, \mathsf{Enc}(\mathsf{pk}, \mathsf{m}_\ell)))) = f(\mathsf{m}_1, \ldots, \mathsf{m}_\ell) + \mathsf{e},$$

where $\mathsf{e} \ 2 \ \mathsf{Z}^k$ *is a noise term with* $\mathsf{kek}_1 \quad \ell B$*.*

Notice that we allow the amount of noise to depend linearly on the parameter ℓ. We also consider linearly homomorphic encryption schemes which allow for shrinking post-evaluation ciphertexts. Such schemes will have two additional algorithms Shrink and ShrinkDec defined below.

Shrink$(\mathsf{pk}, \mathsf{c})$: Takes as input a public key pk and an evaluated ciphertext c and outputs a shrunk ciphertext $\tilde{\mathsf{c}}$.

ShrinkDec$(\mathsf{sk}, \tilde{\mathsf{c}})$: Takes as input a secret key sk and a shrunk ciphertext $\tilde{\mathsf{c}}$ and outputs a message m.

Furthermore, for such schemes we assume that the public key pk contains an *encoding matrix* $\mathsf{T} \in \mathbb{Z}_q^{k \times \ell}$. The encoding matrix T will specifies how *binary messages are supposed to be encoded in the message space* $\mathbb{Z}_q^k$. We can now define the notion of shrinking correctness for a homomorphic encryption scheme HE.

Definition 3.2 (Shrinking Correctness). *Let* $\mathsf{HE} = (\mathsf{KeyGen}, \mathsf{Enc}, \mathsf{Dec}, \mathsf{Eval})$ *be a linearly homomorphic encryption scheme with additional algorithms* $(\mathsf{Shrink}, \mathsf{ShrinkDec})$. *Let* $K = K(\lambda)$. *We say that* HE *is correct up to* K*-noise, if the following holds. For every* $(\mathsf{pk}, \mathsf{sk})$ *in the support of* $\mathsf{KeyGen}(1^\lambda)$*, where* pk *specifies a message space* $\mathbb{Z}_q^k$ *and an encoding matrix* $\mathsf{T} \in \mathbb{Z}_q^{k \times \ell}$*, and every* c *with*

$$\mathsf{Dec}(\mathsf{sk}, \mathsf{c}) = \mathsf{T}\,\mathsf{m} + \mathsf{e},$$

where $\mathsf{m} \in \{0,1\}^\ell$ *and* $\|\mathsf{e}\| < K$*, it holds that*

$$\mathsf{ShrinkDec}(\mathsf{sk}, \mathsf{Shrink}(\mathsf{pk}, \mathsf{c})) = \mathsf{m}.$$

In our main construction, we will set the bounds B (in the Definition 3.1) and K (in Definition 3.2) in such a way that the amount of noise K *tolerated* by shrinking correctness is substantially higher than the noise B introduced by decryption. Finally, we remark that the notion of shrinking correctness also applies to non-homomorphic encryption, albeit it seems not very useful in this context, as optimal rate can be achieved via hybrid encryption.

3.2 A Ciphertext Shrinking Algorithm

We discuss a ciphertext shrinking technique which applies to a broad class of encryption schemes. Let $(\mathsf{KeyGen}, \mathsf{Enc}, \mathsf{Dec}, \mathsf{Eval})$ be an encryption scheme where the public key specifies a message space $\mathbb{Z}_q^\ell$, the secret key S is a matrix in $\mathbb{Z}_q^{\ell \times n}$, (evaluated) ciphertexts are of the form $(\mathsf{c}_1, \mathsf{c}_2)$, and (noisy) decryption computes

$$\mathsf{Dec}(\mathsf{S}, (\mathsf{c}_1, \mathsf{c}_2)) = F(\mathsf{c}_2) \quad \mathsf{S} \quad H(\mathsf{c}_1),$$

where $F(\mathsf{c}_2) \in \mathbb{Z}_q^\ell$, $H(\mathsf{c}_1) \in \mathbb{Z}_q^n$. Here the two functions F and H are part of the description of the scheme and publicly known. Assume in the following that q is even. We describe a general method to shrink ciphertexts of schemes that satisfy these conditions. Consider the following algorithms Shrink and ShrinkDec.

Shrink$(\mathsf{pk}, (\mathsf{c}_1, \mathsf{c}_2))$: Compute $F(\mathsf{c}_2)$ and parse it as $(y_1, \ldots, y_\ell) \in \mathbb{Z}_q^\ell$. Compute the union of intervals

$$U = \bigcup_{i=1}^{\ell} ([q/4 \quad y_i \quad B, q/4 \quad y_i + B] \cup [\ q/4 \quad y_i \quad B, \quad q/4 \quad y_i + B]) \quad \mathbb{Z}_q.$$

Pick any $r \in \mathbb{Z}_q \setminus U$. For $i = 1, \ldots, \ell$ compute $w_i = d y_i + r\mathsf{c}_2$. Output $\tilde{c} = (r, \mathsf{c}_1, w_1, \ldots, w_\ell)$.

ShrinkDec$(\mathsf{S}, \tilde{c} = (r, \mathsf{c}_1, w_1, \ldots, w_\ell))$: Compute $\mathsf{v} = \mathsf{S} \quad H(\mathsf{c}_1)$ and parse $\mathsf{v} = (v_1, \ldots, v_\ell)$. For $i = 1, \ldots, \ell$ set $m_i' = (w_i \quad d v_i \mathsf{c}_2) \mod 2$. Output $\mathsf{m}' = (\mathsf{m}_1', \ldots, \mathsf{m}_\ell')$.

The encoding matrix T for this scheme is defined to be $\mathsf{T} = \frac{q}{2} \quad \mathsf{I}$, where $\mathsf{I} \in \mathbb{Z}_q^{\ell \times \ell}$ is the identity matrix. We now state the conditions under which the modified scheme has shrinking correctness.

Lemma 1. *Let* $\mathsf{HE} = (\mathsf{KeyGen}, \mathsf{Enc}, \mathsf{Dec}, \mathsf{Eval})$ *be an encryption scheme as above, let* $(\mathsf{Shrink}, \mathsf{ShrinkDec})$ *be as above and let* $K = K(\lambda)$*. Let* pk *be a public key for* HE *specifying a message space* Z_q^{ℓ} *with a corresponding secret key* $\mathsf{S} \in \mathsf{Z}_q^{\ell\, n}$*. Then given that* $q > 4\ell K$ *the scheme has shrinking correctness up to noise* K*.*

Proof. Let $(\mathsf{c}_1, \mathsf{c}_2)$ be a ciphertext under pk for which it holds that $F(\mathsf{c}_2) - \mathsf{S} \cdot H(\mathsf{c}_1) = \frac{q}{2} \cdot \mathsf{m} + \mathsf{z}$ for a z with $\|\mathsf{z}\| \leq K$. Let $\mathsf{y} = F(\mathsf{c}_2)$, $\mathsf{v} = \mathsf{S} \cdot H(\mathsf{c}_1)$ and parse $\mathsf{y} = (y_1, \ldots, y_\ell)$ and $\mathsf{v} = (v_1, \ldots, v_\ell)$. I.e., it holds that $\mathsf{y} - \mathsf{v} = \frac{q}{2} \cdot \mathsf{m} + \mathsf{z}$. Fix an index $i \in [\ell]$ and write $y_i - v_i = \frac{q}{2} \cdot \mathsf{m}_i + z_i$, for a $z_i \in [-K, K]$. This implies that $y_i = v_i + z_i + \frac{q}{2} \cdot \mathsf{m}_i$. Note that given that $y_i + r \notin [q/4 - B, q/4 + B]$, $y_i + r \notin [-q/4 - B, -q/4 + B]$ and $z_i \in [-B, B]$, it holds that

$$\begin{aligned} \lfloor y_i + r\mathsf{c}_2 \rceil &= \lfloor y_i + r \rceil - z_i\mathsf{c}_2 \\ &= \lfloor v_i + r + \frac{q}{2} \cdot \mathsf{m}_i\mathsf{c}_2 \rceil \\ &= (\lfloor v_i + r\mathsf{c}_2 + \mathsf{m}_i \rceil \mod 2. \end{aligned}$$

Consequently, it holds that $(\lfloor y_i + r\mathsf{c}_2 \rceil - \lfloor v_i + r\mathsf{c}_2 \rceil) \mod 2 = \mathsf{m}_i$.

Thus, given that it holds for all $i \in [\ell]$ that $y_i + r \notin [q/4 - B, q/4 + B]$ and $y_i + r \notin [-q/4 - B, -q/4 + B]$ then decryption of all m_i will succeed. We will now argue that under the given parameter choice such an r always exists. For every index $i \in [\ell]$ it holds that $y_i + r \notin [q/4 - B, q/4 + B]$ and $y_i + r \notin [-q/4 - B, -q/4 + B]$, if and only if $r \notin [q/4 - y_i - B, q/4 - y_i + B]$ and $r \notin [-q/4 - y_i - B, -q/4 - y_i + B]$. I.e., for every index i there are two intervals $[q/4 - y_i - B, q/4 - y_i + B]$ and $[-q/4 - y_i - B, -q/4 - y_i + B]$ of forbidden choices of r. Given that the set of all forbidden choices

$$U = \bigcup_{i=1}^{\ell} ([q/4 - y_i - B, q/4 - y_i + B] \cup [-q/4 - y_i - B, -q/4 - y_i + B])$$

has less than q elements, we can find an $r \in \mathsf{Z}_q$ which satisfies all constraints. By a union bound it holds that $|U| \leq \ell \cdot 4B$. Consequently, since $q > 4\ell B$, it holds that $\mathsf{Z}_q \cap U \neq \emptyset$, and the compression algorithm will find an r such that decryption will recover every m_i correctly. □

3.3 Packed Regev Encryption

We briefly recall the linearly homomorphic packed Regev encryption and augment it with the shrinking procedures provided in the last section. This will give use a linearly homomorphic scheme with rate $1 - O(1/\lambda)$. Let $q = 2^q$ be a k-bit modulus, let (n, m, ℓ) be positive integers and let χ be a B-bounded error distribution defined on Z. Let $\mathsf{G}_i \in \mathsf{Z}_q^{\ell\, k}$ be a matrix which is zero everywhere, but its i-th row is $\mathsf{g}^\top = (1, 2, \ldots, 2^i, \ldots, 2^k)$. For a $y \in \mathsf{Z}_q$ let $\mathsf{g}^{-1}(y) \in \{0,1\}^k$ be the binary expansion of y, i.e., it holds that $\mathsf{g}^\top \cdot \mathsf{g}^{-1}(y) = y$.

$\mathsf{KeyGen}(1^\lambda)$: Choose $\mathsf{A} \leftarrow_\$ \mathsf{Z}_q^{n\, m}$ uniformly at random. Choose $\mathsf{S} \leftarrow_\$ \mathsf{Z}_q^{\ell\, n}$ uniformly at random and sample $\mathsf{E} \leftarrow_\$ \chi^{\ell\, m}$. Set $\mathsf{B} = \mathsf{S} \cdot \mathsf{A} + \mathsf{E}$. Set $\mathsf{pk} = (\mathsf{A}, \mathsf{B})$ and $\mathsf{sk} = \mathsf{S}$.

$\mathsf{Enc}(\mathsf{pk} = (\mathsf{A}, \mathsf{B}), (m_1, \ldots, m_\ell))$: Choose a random matrix $\mathsf{R} \leftarrow_\$ \{0,1\}^{m\, k}$ and set $\mathsf{C}_1 = \mathsf{A} \cdot \mathsf{R}$ and. $\mathsf{C}_2 = \mathsf{B} \cdot \mathsf{R} + \sum_{i=1}^{\ell} m_i \cdot \mathsf{G}_i$. Output $c = (\mathsf{C}_1, \mathsf{C}_2)$.

$\mathsf{Eval}((f_1, \ldots, f_t), (c_1, \ldots, c_t))$: Parse $c_i = (\mathsf{C}_{1;i}, \mathsf{C}_{2;i})$. Compute $\mathsf{c}_1 = \sum_{i=1}^{t} \mathsf{C}_{1;i} \cdot \mathsf{g}^{-1}(f_i)$ and $\mathsf{c}_2 = \sum_{i=1}^{t} \mathsf{C}_{2;i} \cdot \mathsf{g}^{-1}(f_i)$. Output $c = (\mathsf{c}_1, \mathsf{c}_2)$.

$\mathsf{Dec}(\mathsf{sk} = \mathsf{S}, c = (\mathsf{c}_1, \mathsf{c}_2))$: Compute and output $\lfloor \mathsf{c}_2 - \mathsf{S} \cdot \mathsf{c}_1 \rceil$.

First notice that under the LWE assumption, the matrix B in the public key is pseudorandom. Consequently, given that $m > (n+\ell) \log(q) + \omega(\log(\lambda))$, we can call the leftover-hash lemma [ILL89] to argue that ciphertexts $(\mathsf{C}_1, \mathsf{C}_2)$ are statistically close to uniform [Reg05] and we obtain semantic security.