Bridges connecting Encryption Schemes

Mugurel Barcau[1,2], Cristian Lupaşcu[1,3], Vicenţiu Paşol[1,2], and George C. Ţurcaş[1,4]

[1] certSIGN – Research and Innovation, Bucharest, Romania
[2] Institute of Mathematics "Simion Stoilow" of the Romanian Academy
[3] Ferdinand I Military Technical Academy, Bucharest, Romania
[4] Babeş-Bolyai University, Cluj-Napoca, Romania
{alexandru.barcau,cristian.lupascu,vicentiu.pasol,george.turcas}@certsign.ro

Abstract. The present work investigates morphisms between encryption schemes, called bridges. By associating an encryption scheme to every such bridge, we define and examine their security. Inspired by the bootstrapping procedure used by Gentry to produce fully homomorphic encryption schemes, we exhibit a general recipe for the construction of bridges and we give various examples. We shall also present an example of a bridge that does not fall in this category.

Keywords: Encryption scheme · Homomorphic encryption · IND-CPA

1 Introduction

Homomorphic encryption is a type of encryption which subsequently allows one to perform certain computations on encrypted data without decrypting it. A fully homomorphic encryption scheme, briefly, is an encryption scheme that allows evaluation of any boolean circuit on encrypted data (see [1]). The problem of finding a fully homomorphic encryption scheme was first introduced by Rivest, Adleman and Dertouzos in [26]; however the first candidate was constructed by Gentry thirty years later in [15,16].

Before Gentry's breakthrough, there were several partial results [20,25,5,23]. The authors of [5] propose an interesting idea for the realisation of FHE. Roughly speaking, if one possesses two encryption schemes, each homomorphic with respect to an operation, and a way of moving encryptions from one to another, then one can evaluate any boolean circuit. Unfortunately, the construction proposed in [5] works only for certain arithmetic circuits.

The idea of moving from an encryption scheme to another was remarkably used by Gentry [15,16] in order to transform a somewhat homomorphic encryption scheme into a (leveled) fully homomorphic scheme. To be precise, the *Recrypt* algorithm takes as input a ciphertext and certain encryptions of the secret key under a different key and evaluates homomorphically the decryption algorithm in order to produce an encryption of the same plaintext under the new key. Under the definition we propose, the *Recrypt* algorithm is a bridge from a somewhat homomorphic encryption scheme to itself. The recipe can be extended

to produce a bridge from any encryption scheme to any somewhat homomorphic encryption scheme that can handle (correctly evaluate) the decryption circuit of the former.

Perhaps connected to the same idea is the work in [12], where maps between two encryption schemes are used to construct a 2-party computation protocol, called an Encryption Switching Protocol (ESP). The examples proposed in [12] and [9] consist of two encryptions schemes over the same plaintext, which has a structure of a ring, and switching protocols between them. One of the schemes is homomorphic with respect to addition and the other is homomorphic for the multiplication. An ESP of this form can be used to construct a secure general 2-party computational protocol.

Switching between one encryption scheme to another, in order to securely perform some homomorphic operations, is a recurrent theme in the literature. In this respect, it is important to formally define and analyze the security implications of such protocols, which represents the main goal of the present work. We shall call a map (or a morphism) between encryption schemes satisfying certain properties a *bridge*. The terminology is borrowed from [6], where the expression "bridge between encryption schemes" is briefly used in reference to a hybrid solution for switching between FHE schemes in order to optimize performance of certain homomorphic computations on encrypted data.

Our contribution In this paper, we first propose a general definition for a bridge, formalizing the conditions under which an algorithm that publicly transforms encrypted data from one scheme to another should perform. We provide a general recipe, inspired by Gentry's idea, for the construction of bridges and then apply it to give various examples. This general recipe can be modified in various ways and we demonstrate this by presenting a variant of it. We also present an additional example of a bridge that does not fall in the category of Gentry type bridges. We canonically associate to any bridge an encryption scheme and then define the security of a bridge as being the security of its associated encryption scheme. This association is widely used in mathematics when someone needs to replace a morphism between two objects by an object. More precisely, it consists in substituting a map by its graph, whenever this is possible. We prove a general theorem (Theorem 2) asserting that the security of a bridge reduces to the security of the first encryption scheme together with a technical additional assumption. We show that the latter technicality is in fact a natural condition by proving that bridges obtained using Gentry's *Recrypt* idea satisfy this assumption (Proposition 2). The examples of bridges presented herein were implemented and comments on their performance are to be found in the appendix of this article.

Organization The article is organized as follows. Section 2 consists of some mathematical background and preliminaries about encryption schemes used in the rest of the article. It starts by recalling some terminology and theoretical facts about finite distributions. In the same section, we also give the definition of

a bridge. The contributions in section 3 regard the security of a bridge between two encryption schemes. The main result of our paper (Theorem 2) is proved in this section. In section 4, we show that Gentry's *Recrypt* algorithm gives a general recipe for the construction of bridges. Using the main result from the previous section, we prove that bridges generated using this recipe are secure. By representing the decryption circuit of a specific encryption scheme in four different ways, in section 5 we give four different examples of bridges from the same encryption scheme to various FHE schemes. The article ends with a section in which a bridge between the schemes GM and SYY is exhibited. This bridge is not obtained using the recipe presented in section 4. Its security follows from results in section 3. The homomorphic evaluation of a comparison circuit is presented as an application to the latter bridge. In the appendix of this article, we report on the results of several experiments involving the implementation of the bridges introduced in the last two sections.

Acknowledgment The authors are indebted to George Gugulea and Mihai Togan for helpful discussions and comments during the preparation of this work.

2 Preliminaries

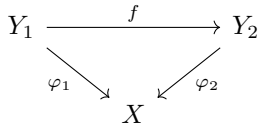
In all our definitions, we denote the security parameter by λ. We say that a function $\mu : \mathbb{N} \to [0, +\infty)$ is a negligible function if for any positive integer c there exists a positive integer N_c, such that $\mu(n) < \dfrac{1}{n^c}$ for all $n \geq N_c$.

2.1 Finite Distributions

A finite probability distribution is a probability distribution with finite support. If X is a finite distribution, we denote by $|X|$ its support. If X and Y are finite distributions, then a morphism $\varphi : Y \to X$ is a map of sets (still denoted by) $\varphi : |Y| \to |X|$ such that

$$\Pr\{X = x\} = \sum_{y \in \varphi^{-1}(x)} \Pr\{Y = y\}.$$

for all $x \in |X|$. Notice that if $\varphi^{-1}(x)$ is empty then $\Pr\{X = x\} = 0$, which means that φ is surjective onto $\{x \in |X| \mid \Pr\{X = x\} \neq 0\}$. The composition of two morphisms is a morphism and the identity map $1_{|X|} : |X| \to |X|$ gives rise to a morphism of distributions $1_X : X \to X$ so that the class of finite distributions together with all morphisms between them forms a category denoted $\mathscr{F}in\mathscr{D}ist$. As usual, two finite distributions are isomorphic if there exist a morphism between them that has an inverse. If X is a finite distribution, then the slice category (cf. [3]) $\mathscr{F}in\mathscr{D}ist_X$ of X-distributions consists of pairs (Y, φ) where Y is a finite distribution and $\varphi : Y \to X$ is a morphism of finite distributions. A morphism of X-distributions $f : (Y_1, \varphi_1) \to (Y_2, \varphi_2)$, consists of a morphism of finite distributions $f : Y_1 \to Y_2$ such that the following diagram



is commutative.

If $x \in |X|$ with $\Pr\{X = x\} \neq 0$ and (Y, φ) is an X-distribution then the *fiber of* Y *over* x is the finite distribution $Y|_{X=x}$ with support $\varphi^{-1}(x)$ and $\Pr\{Y|_{X=x} = y\} = \dfrac{\Pr\{Y = y\}}{\Pr\{X = x\}}$, for all $y \in \varphi^{-1}(x)$.

If (Y_1, φ_1) and (Y_2, φ_2) are two X-distributions we construct the following product $Y_1 \times_X Y_2$. The support of this distribution is

$$|Y_1 \times_X Y_2| := \{(y_1, y_2) | y_1 \in |Y_1|, y_2 \in |Y_2| \text{ such that } \varphi_1(y_1) = \varphi_2(y_2)\}.$$

If $x = \varphi_1(y_1) = \varphi_2(y_2)$ and $\Pr\{X = x\} \neq 0$, then

$$\Pr\{Y_1 \times_X Y_2 = (y_1, y_2)\} := \frac{\Pr\{Y_1 = y_1\} \cdot \Pr\{Y_2 = y_2\}}{\Pr\{X = x\}}.$$

Moreover, when $\Pr\{X = x\} = 0$, then

$$\Pr\{Y_1 \times_X Y_2 = (y_1, y_2)\} := 0.$$

Finally, the structural morphism of $\psi : |Y_1 \times_X Y_2| \to X$ is $\psi := \varphi_1 \circ \mathrm{pr}_1 = \varphi_2 \circ \mathrm{pr}_2$, where $\mathrm{pr}_i : |Y_1 \times_X Y_2| \to |Y_i|, i \in \{1, 2\}$ are the usual projections.

We remark that $|Y_1 \times_X Y_2|$ is the usual fiber product in the category of sets, but $Y_1 \times_X Y_2$ is not a fiber product in the category $\mathscr{F}in\mathscr{D}ist$. However, the distribution $Y_1 \times_X Y_2$ is a product in the following sense. If one constructs the distribution of triples (x, y_1, y_2): x is chosen from $|X|$ according to X, y_1 and y_2 are chosen independently from $\varphi_1^{-1}(x)$ and $\varphi_2^{-1}(x)$ according to Y_1 and Y_2 respectively, then one obtains a distribution isomorphic to $Y_1 \times_X Y_2$.

Any finite distribution whose support is a one-point set is a final object in $\mathscr{F}in\mathscr{D}ist$. We shall denote by $Y_1 \times Y_2$ the product $Y_1 \times_X Y_2$, where X is any of the final objects of $\mathscr{F}in\mathscr{D}ist$.

Notice that if Y is an X-distribution, then the distribution $X \times_X Y$ is isomorphic to Y as X-distributions (here we view X as an X-distribution via the identity map). We will sometimes identify the distribution $X \times_X Y$ with Y without mentioning it, if this is clear from the context. Morally, $X \times_X Y$ is the distribution Y whose associated map φ is known.

If $\{X_\lambda\}_{\lambda \in \mathbb{N}}$, $\{Y_\lambda\}_{\lambda \in \mathbb{N}}$ are ensembles of finite distributions then we define a morphism from the latter to the former as being a set of morphisms of finite distributions $\varphi_\lambda : Y_\lambda \to X_\lambda$ for all λ. One can verify immediately that ensembles of finite distributions together with morphisms form a category. If we fix an ensemble $\{X_\lambda\}_\lambda$, then we obtain the slice category of $\{X_\lambda\}_\lambda$-ensembles of finite distributions. In this category we define, as before, the product of the two ensembles $\{Y_\lambda\}_\lambda$, $\{Z_\lambda\}_\lambda$ as $\{Y_\lambda \times_{X_\lambda} Z_\lambda\}_\lambda$.

The first part of the following statement is Definition 2 from [19].

Definition 1. *An ensemble $\{X_\lambda\}_\lambda$ of finite distributions is polynomial-time constructible if there exists a PPT algorithm A such that $A(1^\lambda) = X_\lambda$, for every λ. An $\{X_\lambda\}_\lambda$-ensemble of finite distributions $\{(Y_\lambda, \varphi_\lambda)\}_\lambda$ is polynomial-time constructible on fibers if there exist a PPT algorithm A, such that for any $x_\lambda \in |X_\lambda|$ we have $A(1^\lambda, x_\lambda) = Y_\lambda|_{X_\lambda = x_\lambda}$.*

We will also use the following notion of computational (or polynomial) indistinguishability from [20] and [19].

Definition 2. *Two ensembles of finite distributions $\{X_\lambda\}_\lambda$ and $\{Y_\lambda\}_\lambda$ are called computationally indistinguishable if for any PPT distinguisher D, the quantity*

$$|\Pr\{D(X_\lambda) = 1\} - \Pr\{D(Y_\lambda) = 1\}|$$

is negligible as a function of λ.

When referring to ensembles of finite distributions, we will leave out the subscript λ if this is clear from the context.

2.2 Encryption Schemes and Bridges

A public key (or asymmetric) encryption scheme

$$\mathscr{S} = (\mathrm{KeyGen}_{\mathscr{S}}, \mathrm{Enc}_{\mathscr{S}}, \mathrm{Dec}_{\mathscr{S}})$$

is a triple of PPT algorithms as follows:

- **Key Generation.** The algorithm $(sk, pk) \leftarrow \mathrm{KeyGen}_{\mathscr{S}}(1^\lambda)$ takes a unary representation of the security parameter λ and outputs a secret decryption key sk and a public encryption key pk;
- **Encryption.** The algorithm $c \leftarrow \mathrm{Enc}_{\mathscr{S}}(pk, m)$ takes the public key pk and a message $m \in \mathscr{P}$ and outputs a ciphertext $c \in \mathscr{C}$;
- **Decryption.** The algorithm $m^\star \leftarrow \mathrm{Dec}_{\mathscr{S}}(sk, c)$ takes the secret key sk and a ciphertext $c \in \mathscr{C}$ and outputs a message $m^\star \in \mathscr{P}$;

where the finite sets $\mathscr{P}$ and $\mathscr{C}$ represent the plaintext space, respectively the ciphertext space. The algorithms above must satisfy the correctness property

$$\Pr\{\mathrm{Dec}_{\mathscr{S}}(sk, \mathrm{Enc}_{\mathscr{S}}(pk, m)) = m\} = 1 - \mathrm{negl}(\lambda),$$

where the probability is taken over the experiment of running the key generation and encryption algorithms and choosing uniformly $m \leftarrow \mathscr{P}$.

A private key (or symmetric) encryption scheme is a public key encryption scheme for which the public and secret keys are equal.

We say that an instance pk of the public key, or an instance sk of the secret key, is of *level* λ_0 if it is outputted by the key generation algorithm whose input is the unary representation of λ_0.

Remark 1. In the language of ensembles of finite distributions, the public keys of an encryption scheme form an SK-ensemble of finite distributions, where SK is the ensemble of secret keys. Moreover, an encryption scheme is just a collection of PK-ensembles of finite distributions indexed by the plaintext space that are polynomial-time constructible on fibers (here PK is the ensemble of public keys).

A homomorphic (public-key) encryption scheme

$$\mathscr{H} = (\mathrm{KeyGen}_{\mathscr{H}}, \mathrm{Enc}_{\mathscr{H}}, \mathrm{Dec}_{\mathscr{H}}, \mathrm{Eval}_{\mathscr{H}})$$

is a quadruple of PPT algorithms such that $(\mathrm{KeyGen}_{\mathscr{H}}, \mathrm{Enc}_{\mathscr{H}}, \mathrm{Dec}_{\mathscr{H}})$ is a public-key encryption scheme and the $\mathrm{KeyGen}_{\mathscr{H}}$ algorithm also outputs an additional evaluation key evk besides sk and pk, where the **Homomorphic Evaluation** algorithm $\mathrm{Eval}_{\mathscr{H}}$ takes the evaluation key evk, a circuit $f : \mathscr{P}^{\ell} \to \mathscr{P}$ and a set of ℓ ciphertexts $c_1, ..., c_{\ell} \in \mathscr{C}$, and outputs a ciphertext c_f.

We say that a homomorphic encryption scheme $\mathscr{H}$ is $\mathcal{C}$-*homomorphic* for a class of functions $\mathcal{C} = \{\mathcal{C}_\lambda\}_{\lambda \in \mathbb{N}}$, if for any sequence of functions $f_\lambda \in \mathcal{C}_\lambda$ and respective inputs $\mu_1, ..., \mu_\ell \in \mathscr{P}$ (where $\ell = \ell(\lambda)$), it holds that

$$\Pr[\mathrm{Dec}_{\mathscr{H}}(sk, \mathrm{Eval}_{\mathscr{H}}(evk, f_\lambda, c_1, ..., c_\ell)) \neq f_\lambda(\mu_1, ..., \mu_\ell)] = \mathrm{negl}(\lambda),$$

where $(pk, sk, evk) \leftarrow \mathrm{KeyGen}_{\mathscr{H}}(1^\lambda)$ and $c_i \leftarrow \mathrm{Enc}_{\mathscr{H}}(pk, \mu_i)$ for all i.

In addition, a homomorphic encryption scheme $\mathscr{H}$ is *compact* if there exist a polynomial $s = s(\lambda)$ such that the output length of $\mathrm{Eval}_{\mathscr{H}}$ is at most s bits long, regardless of f or the number of inputs.

An encryption scheme is called *fully homomorphic (FHE)* if it is homomorphic for the class of all boolean functions and it satisfies the compactness condition.

We now give the definition of a bridge:

Definition 3. *Let $\mathscr{S}_j = (\mathscr{P}_j, \mathscr{C}_j, \mathrm{KeyGen}_j, \mathrm{Enc}_j, \mathrm{Dec}_j)$, $j \in \{1, 2\}$ be two encryption schemes. A bridge $\mathbf{B}_{\iota,f}$ from $\mathscr{S}_1$ to $\mathscr{S}_2$ consists of:*

1. *An injective function $\iota : \mathscr{P}_1 \to \mathscr{P}_2$ such that:*
   (a) *ι is computable by a deterministic polynomial time algorithm;*
   (b) *there exists a deterministic polynomial time algorithm which computes $\iota^{-1} : \mathscr{P}_2 \to \mathscr{P}_1$, i.e. outputs the symbol $\perp$ if the input is not in the image of ι and the preimage of the input otherwise,*
2. *A PPT bridge key generation algorithm, which has the following three stages. First, the algorithm gets the security parameter λ and uses it to run the key generation algorithm of $\mathscr{S}_1$ in order to obtain a pair of keys sk_1, pk_1. In the second stage the algorithm uses sk_1 to find a secret key sk_2 of level λ for $\mathscr{S}_2$, and then calls the key generation algorithm of $\mathscr{S}_2$ to produce pk_2. In the final stage, the algorithm takes as input the quadruple (sk_1, pk_1, sk_2, pk_2) and outputs a bridge key bk.*
3. *A PPT algorithm f which takes as input the bridge key bk and a ciphertext $c_1 \in \mathscr{C}_1$ and outputs a ciphertext $c_2 \in \mathscr{C}_2$,*

such that

$$Pr\{\mathrm{Dec}_2(sk_2, f(bk, \mathrm{Enc}_1(pk_1, m))) = \iota(m)\} = 1 - \mathrm{negl}(\lambda),$$

where the probability is taken over the experiment of running the key generation and encryption algorithms and choosing uniformly $m \leftarrow \mathscr{P}_1$.

Notice that the definition above includes the case in which any of the two schemes is symmetric. Also, the plaintext spaces are fixed, i.e. they do not depend on the security parameter λ. One can define a bridge between encryption schemes for which the plaintext spaces do depend on λ, as in the case of RSA or Paillier cryptosystems. However, in this article we are considering only the former situation.

Remark 2. The bridge key generation algorithm does not necessarily output all possible pairs (sk_1, sk_2). Even though any secret key sk_1 of the scheme $\mathscr{S}_1$ may be outputted by the key generation algorithm of the bridge, only few sk_2's may occur. The bridge key generation algorithm produces the following $\{SK_{1,\lambda}\}_\lambda$-ensembles of finite distributions $\{SK_{2,\lambda}\}_\lambda$, $\{PK_{i,\lambda}\}_\lambda, i \in \{1,2\}$, and $\{BK_\lambda\}_\lambda$. The morphisms between these ensembles of finite distributions are illustrated in Figure 1.

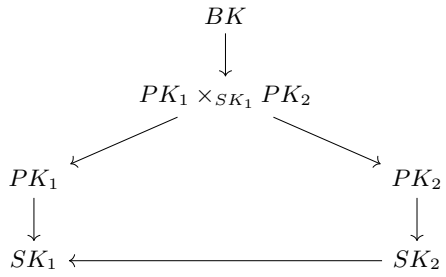


Fig. 1. Probability distributions for bridges

We mentioned earlier the idea of thinking of a bridge as a (category theoretical) morphism between encryption schemes. Although we do not claim to have defined a category, from this point of view, it is natural to address the existence of identity morphisms. We briefly explain below that the identity map between one encryption scheme to itself is a bridge.

Example 1. If $\mathscr{S}$ is an encryption scheme, then the identity map $\mathscr{C} \to \mathscr{C}$ gives rise to a bridge. The bridge key generation algorithm generates a unique secret key sk and two (independently generated) public keys pk_1, pk_2 corresponding to this secret key. The algorithm outputs $(sk, pk_1, sk, pk_2, \mathrm{NIL})$. We emphasize that the bridge key and the choices of pk_1 and pk_2 do not play any role in the evaluation of the bridge map.

3 The security of a bridge

The aim of this section is to define and investigate the IND-CPA security of a bridge. We start by defining and extending the notion of IND-CPA security of a scheme and then we move to the discussion concerning the security of a bridge.

Definition 4 (IND-CPA Security). *Let $\mathscr{S} = (\mathrm{KeyGen}_{\mathscr{S}}, \mathrm{Enc}_{\mathscr{S}}, \mathrm{Dec}_{\mathscr{S}})$ be a public key encryption scheme. We define an experiment $\mathrm{Expr}_b[\mathcal{A}]$ parameterized by a bit $b \in \{0,1\}$ and an efficient (PPT) adversary $\mathcal{A}$:*

$$\begin{aligned}\mathrm{Expr}_b[\mathcal{A}](1^\lambda): &1.\ (pk, sk) \longleftarrow \mathrm{KeyGen}_{\mathscr{S}}(1^\lambda)\\ &2.\ (m_0, m_1) \longleftarrow \mathcal{A}(1^\lambda, pk)\\ &3.\ \mathrm{ct} \longleftarrow \mathrm{Enc}_{\mathscr{S}}(pk, x_b)\\ &4.\ b' \longleftarrow \mathcal{A}(\mathrm{ct})\\ &5.\ \mathrm{return}(b')\end{aligned}$$

The advantage of adversary $\mathcal{A}$ against the IND-CPA security of the scheme is

$$\mathrm{Adv}^{IND\text{-}CPA}[\mathcal{A}](\lambda) := |\Pr\left\{\mathrm{Expr}_0[\mathcal{A}](1^\lambda) = 1\right\} - \Pr\left\{\mathrm{Expr}_1[\mathcal{A}](1^\lambda) = 1\right\}|,$$

where the probability is over the randomness of $\mathcal{A}$ and of the experiment. We say that the scheme is IND-CPA secure if for any efficient adversary $\mathcal{A}$, the advantage $\mathrm{Adv}^{IND\text{-}CPA}[\mathcal{A}]$ is negligible as a function of λ. In the case of a symmetric encryption scheme, the adversary $\mathcal{A}$ is given access to an encryption oracle.

Remark 3. As in the previous definition, when considering the security of a private encryption scheme, it is standard to replace the public key by an encryption oracle. From this point of view, a symmetric encryption scheme is a public encryption scheme whose public key consists of the access to an encryption oracle. Although we will give security definitions and proofs for public key encryption schemes, unless otherwise specified, these can be extended to the symmetric key setting using the above paradigm.

Let $\mathscr{S}$ be an encryption scheme and let K be some data outputted by an oracle whose input is the triple $(1^\lambda, sk_{\mathscr{S}}, pk_{\mathscr{S}})$. We shall denote by $\mathscr{S}[K]$ the encryption scheme whose public key is the pair $(pk_{\mathscr{S}}, K)$, and the encryption and decryption algorithms are exactly as in $\mathscr{S}$. The only difference between the schemes $\mathscr{S}$ and $\mathscr{S}[K]$ is related to their security. More precisely, an adversary attacking the scheme $\mathscr{S}[K]$ has more information than an adversary attacking $\mathscr{S}$. We say that an adversary $\mathcal{A}$ attacking $\mathscr{S}[K]$ is an adversary attacking $\mathscr{S}$ *with knowledge* K. For example, K can be a set consisting of $\mathscr{S}$-encryptions of the bit representation of the secret key, as used in [17] for the bootstrapping procedure. It is commonly assumed that such K's do not affect the security of the encryption scheme, assumption called *circular security*. The following definition aims to generalize the *circular security* assumption for some general data K.

Definition 5. *We say that some knowledge K is negligible for an encryption scheme $\mathscr{S}$ if for any adversary $\mathcal{A}$ attacking $\mathscr{S}[K]$ there exists an adversary $\mathcal{A}'$ attacking $\mathscr{S}$ such that*

$$|\mathrm{Adv}^{IND\text{-}CPA}[\mathcal{A}](\lambda) - \mathrm{Adv}^{IND\text{-}CPA}[\mathcal{A}'](\lambda)|$$

is negligible as a function of λ.

Notice that any adversary attacking $\mathscr{S}$ gives rise, in the obvious way, to an adversary attacking $\mathscr{S}[K]$, so that if K is negligible for $\mathscr{S}$ then the IND-CPA security of $\mathscr{S}$ is equivalent to the IND-CPA security of $\mathscr{S}[K]$.

In order to define the IND-CPA security of a bridge, we shall associate to it, in a canonical way, an encryption scheme; the security of the bridge will be, by definition, the security of the associated encryption scheme. Let $\mathbf{B}_{\iota,f}$ be a bridge, then the associated encryption scheme

$$\mathscr{G}_f = (\mathscr{P}_{\mathscr{G}_f}, \mathscr{C}_{\mathscr{G}_f}, \mathrm{KeyGen}_{\mathscr{G}_f}, \mathrm{Enc}_{\mathscr{G}_f}, \mathrm{Dec}_{\mathscr{G}_f})$$

is defined as follows. The plaintext space is $\mathscr{P}_{\mathscr{G}_f} = \mathscr{P}_1$, and the ciphertext space is $\mathscr{C}_{\mathscr{G}_f} = \mathscr{C}_1 \times \mathscr{C}_2$. The algorithm $\mathrm{KeyGen}_{\mathscr{G}_f}$ uses the key generation algorithm of the bridge to get $sk_1, pk_1, sk_2, pk_2, bk$. The secret key $sk_{\mathscr{G}_f}$ is the pair (sk_1, sk_2), and the public key $pk_{\mathscr{G}_f}$ is (pk_1, pk_2, bk).

For any $m \in \mathscr{P}_{\mathscr{G}_f}$, its encryption is defined by:

$$\mathrm{Enc}_{\mathscr{G}_f}(pk_{\mathscr{G}_f}, m) := (a, f(bk, b))\,,$$

where $a, b \leftarrow \mathrm{Enc}_1(pk_1, m)$. Finally, the decryption of a ciphertext $c_{\mathscr{G}_f} = (c_1, c_2) \in \mathscr{C}_1 \times \mathscr{C}_2$ is obtained using the formula:

$$\mathrm{Dec}_{\mathscr{G}_f}(sk_{\mathscr{G}_f}, c_{\mathscr{G}_f}) := \mathrm{Dec}_1(sk_1, c_1).$$

We notice that the decryption of $\mathscr{G}_f$ satisfies

$$\mathrm{Dec}_{\mathscr{G}_f}\Big(sk_{\mathscr{G}_f}, (a, f(bk, b))\Big) = \iota^{-1}\Big(\mathrm{Dec}_2(sk_2, f(bk, b))\Big),$$

for any $(a, f(bk, b)) \leftarrow \mathrm{Enc}_{\mathscr{G}_f}(pk_{\mathscr{G}_f}, m)$ with overwhelming probability, due to the third condition in the definition of a bridge. One can immediately verify that the correctness of the encryption scheme $\mathscr{G}_f$ follows from the correctness of $\mathscr{S}_1$.

Remark 4. The notation and construction are inspired by the construction of the graph of a function.

Now we define the IND-CPA security of a bridge.

Definition 6. *The IND-CPA security of the bridge $\mathbf{B}_{\iota,f}$ is the IND-CPA security of its associated encryption scheme $\mathscr{G}_f$.*

We have the following immediate result.

Proposition 1. *If a bridge $\mathbf{B}_{\iota,f}$ is IND-CPA secure, then the encryption scheme $\mathscr{S}_1$ is also IND-CPA secure.*

Proof. Indeed, we can associate to any adversary $\mathcal{A}_1$ which is trying to break the IND-CPA security of $\mathscr{S}_1$, an adversary $\mathcal{A}_f$ for the encryption scheme $\mathscr{G}_f$, as follows. For any pair $(a, f(bk, b))$ proposed by the challenger to $\mathcal{A}_f$, where $a, b \leftarrow \mathrm{Enc}_1(m)$, the attacker $\mathcal{A}_f$ sends the triple (λ, pk_1, a) to $\mathcal{A}_1$ and returns the output of $\mathcal{A}_1(\lambda, pk_1, a)$.

It is clear that

$$\mathrm{Adv}^{\text{IND-CPA}}[\mathcal{A}_f](\lambda) = \mathrm{Adv}^{\text{IND-CPA}}[\mathcal{A}_1](\lambda),$$

and the result follows.

In the next theorem, the encryption scheme $\mathscr{S}_1[PK_{\mathscr{G}_f}]$ is the scheme $\mathscr{S}_1$ with knowledge $PK_{\mathscr{G}_f}$. Namely, after running the key generation algorithm of $\mathscr{S}_1$ and receiving the pair (sk_1, pk_1), the challenger has access to an oracle that runs the second part of the key generation algorithm of the bridge to get sk_2, pk_2, bk. Thus, an IND-CPA attacker on this scheme will receive pk_1, pk_2, bk.

Theorem 1. *The encryption scheme $\mathscr{S}_1[PK_{\mathscr{G}_f}]$ is IND-CPA secure if and only if $\mathscr{G}_f$ is IND-CPA secure.*

Proof. We first show that if $\mathscr{G}_f$ is IND-CPA secure, then $\mathscr{S}_1[PK_{\mathscr{G}_f}]$ is IND-CPA secure. Suppose $\mathcal{A}$ is an IND-CPA attacker on $\mathscr{S}_1[PK_{\mathscr{G}_f}]$ scheme. We construct the following adversary $\mathcal{B}$ attacking the IND-CPA security of $\mathscr{G}_f$ as follows. At start, $\mathcal{B}$ takes as input $(1^\lambda, pk_{\mathscr{G}_f})$ and executes the program $\mathcal{A}(1^\lambda, pk_{\mathscr{G}_f})$. The attacker $\mathcal{B}$ receives $(m_0, m_1) \leftarrow \mathcal{A}(1^\lambda, pk_{\mathscr{G}_f})$ and sends this pair to its challenger. The latter samples $b \leftarrow \{0, 1\}$ and returns to $\mathcal{B}$ the challenge $c = (c_1, f(bk, c_1'))$, where $c_1, c_1' \leftarrow \mathrm{Enc}_1(pk_1, m_b)$. Finally, $\mathcal{B}$ terminates by outputting the bit $b' \leftarrow \mathcal{A}(c_1)$. One obtains that

$$\mathrm{Adv}^{\text{IND-CPA}}_{\mathscr{G}_f}[\mathcal{B}](\lambda) = \mathrm{Adv}^{\text{IND-CPA}}_{\mathscr{S}_1[PK_{\mathscr{G}_f}]}[\mathcal{A}](\lambda),$$

which proves this implication.

To prove the other implication, we first point out that using a standard hybrid argument one can show that the IND-CPA security of an encryption scheme is equivalent its 2-IND-CPA security (see [28] for a detailed discussion). As opposed to the IND-CPA game, in the 2-IND-CPA game the attacker receives from the challenger two encryptions of m_b, instead of one.

Suppose that $\mathcal{B}$ is an IND-CPA attacker on $\mathscr{G}_f$. We construct a 2-IND-CPA attacker $\mathcal{A}$ for the scheme $\mathscr{S}_1[PK_{\mathscr{G}_f}]$ as follows. The attacker $\mathcal{A}$ receives as input $(1^\lambda, pk_{\mathscr{G}_f})$ and sends this to $\mathcal{B}$. On this input, the attacker $\mathcal{B}$ produces two messages $m_0, m_1 \in \mathscr{P}_1$ which are sent to $\mathcal{A}$ and the latter passes them to its challenger. After receiving m_0, m_1, the challenger of $\mathcal{A}$ chooses $b \leftarrow \{0, 1\}$ and returns $c_1, c_1' \leftarrow \mathrm{Enc}_1(m_b)$ to the attacker $\mathcal{A}$. The attacker $\mathcal{A}$, knowing bk, is

able to compute $f(bk, c_1') \in \mathcal{C}_2$ and finishes by outputting $b' \leftarrow \mathcal{B}(c_1, f(bk, c_1'))$. Now, one can verify that

$$\mathrm{Adv}^{2-\mathrm{IND}-\mathrm{CPA}}_{\mathscr{S}_1[PK_{\mathscr{G}_f}]}[\mathcal{A}](\lambda) = \mathrm{Adv}^{\mathrm{IND}-\mathrm{CPA}}_{\mathscr{G}_f}[\mathcal{B}](\lambda).$$

By the discussion in the previous paragraph, the scheme $\mathscr{S}_1[PK_{\mathscr{G}_f}]$ is 2-IND-CPA secure, so that $\mathcal{A}$ has negligible advantage. The last equality shows that $\mathcal{B}$ has also negligible advantage, which ends the argument.

Recall that the *bridge key generation algorithm* produces the following ensembles of $\{SK_{1,\lambda}\}_\lambda$ distributions: $\{PK_{1,\lambda}\}_\lambda$, $\{PK_{2,\lambda}\}_\lambda$ and $\{BK_\lambda\}_\lambda$. Let $\mathscr{F}$ be the ensemble of finite distributions of triples (pk_1, pk_2, bk). Note that $\pi_1 : \mathscr{F} \to PK_1$ is a morphism of finite distributions, so $\mathscr{F}$ is a PK_1-distribution as discussed in Section 2.1.

Theorem 2. *Assume that $\mathscr{S}_1$ is IND-CPA secure and there exists a polynomial time constructible on fibers ensemble of PK_1-distributions $\widetilde{\mathscr{F}}$ which is computational indistinguishable from $\mathscr{F}$. Then the bridge $\mathbf{B}_{\iota,f}$ is IND-CPA secure.*

Proof. Without losing generality we assume that $\mathscr{P}_1 = \{0, 1\}$. By the above theorem, it is enough to prove that $\mathscr{S}_1[PK_{\mathscr{G}_f}]$ is IND-CPA secure. We do the proof by contradiction, so we suppose that $\mathcal{A}$ is an adversary attacking the scheme $\mathscr{S}_1[PK_{\mathscr{G}_f}]$ with non-negligible advantage. We think of $\mathcal{A}$ as being a distinguisher between the ensembles of distributions $\mathscr{F} \times_{PK_1} \mathrm{Enc}_1(PK_1, 0)$ and $\mathscr{F} \times_{PK_1} \mathrm{Enc}_1(PK_1, 1)$. The first claim is that, if $\mathcal{A}$ can distinguish with non-negligible advantage between these two distributions then $\mathcal{A}$ distinguishes with non-negligible advantage between $\widetilde{\mathscr{F}} \times_{PK_1} \mathrm{Enc}_1(PK_1, 0)$ and $\widetilde{\mathscr{F}} \times_{PK_1} \mathrm{Enc}_1(PK_1, 1)$. To prove the claim we suppose that this is not the case and we construct a distinguisher $\mathcal{D}$ for the distributions $\mathscr{F}$ and $\widetilde{\mathscr{F}}$. As the ensemble of distributions $\widetilde{\mathscr{F}}$ is computationally indistinguishable from $\mathscr{F}$, for every λ, the distribution $\widetilde{\mathscr{F}}_\lambda$ consists of triples of the form (pk_1, α, β).

The distinguisher $\mathcal{D}$ runs as follows. It first receives a triple (pk_1, x, y) from the challenger, chooses at random a bit $b \leftarrow \{0, 1\}$ and encrypts b using pk_1 to obtain a ciphertext c. The distinguisher $\mathcal{D}$ sends the quadruple (pk_1, x, y, c) to $\mathcal{A}$ and outputs

$$\mathcal{D}(pk_1, x, y) := \begin{cases} 1 & \text{if } \mathcal{A}(pk_1, x, y, c) = b \\ 0 & \text{otherwise} \end{cases}.$$

We note that the labels $b = 1$ and $b = 0$, as outputted by $\mathcal{A}$, correspond to the ensembles $\mathscr{F}$ and $\widetilde{\mathscr{F}}$, respectively. Notice that

$$\Pr\{\mathrm{Expr}_1[\mathcal{D}] = 1\} = \frac{1}{2}\Pr\{\mathrm{Expr}_0[\mathcal{A}|_{\mathscr{F}}] = 0\} + \frac{1}{2}\Pr\{\mathrm{Expr}_1[\mathcal{A}|_{\mathscr{F}}] = 1\},$$

where $\mathrm{Expr}_b[\mathcal{A}|_{\mathscr{F}}]$ means that in the experiment Expr_b the challenger chooses the triple $(pk_1, x, y) = (pk_1, pk_2, bk)$ according to $\mathscr{F}$. Using analogous notation

for $\widetilde{\mathscr{F}}$, we have:

$$\Pr\{\mathrm{Expr}_0[\mathcal{D}]=1\}=\frac{1}{2}\Pr\left\{\mathrm{Expr}_0[\mathcal{A}|_{\widetilde{\mathscr{F}}}]=1\right\}+\frac{1}{2}\Pr\left\{\mathrm{Expr}_1[\mathcal{A}|_{\widetilde{\mathscr{F}}}]=0\right\}.$$

Since the advantage of $\mathcal{A}|_{\mathscr{F}}$ is non-negligible, there exist a positive integer k such that

$$\left|\Pr\{\mathrm{Expr}_1[\mathcal{D}]=1\}-\frac{1}{2}\right|>\frac{1}{\lambda^k} \tag{1}$$

for infinitely many λ's. Also, since $\mathrm{Adv}[\mathcal{A}|_{\widetilde{\mathscr{F}}}](\lambda)=\mathrm{negl}(\lambda)$, we have

$$\left|\Pr\{\mathrm{Expr}_0[\mathcal{D}]=1\}-\frac{1}{2}\right|=\mathrm{negl}(\lambda). \tag{2}$$

From (1) and (2) we infer that

$$\mathrm{Adv}[\mathcal{D}](\lambda)=|\Pr\{\mathrm{Expr}_1[\mathcal{D}]=1\}-\Pr\{\mathrm{Expr}_0[\mathcal{D}]=1\}|$$

is non-negligible, which contradicts the assumption about the computational indistinguishability of the two distributions $\mathscr{F}$ and $\widetilde{\mathscr{F}}$.

Now we use $\mathcal{A}|_{\widetilde{\mathscr{F}}}$ to construct an adversary $\mathcal{B}$ on $\mathscr{S}_1$. After receiving the pair (pk_1,c) (as before $c\leftarrow\mathrm{Enc}_1(pk_1,b)$) from the challenger, $\mathcal{B}$ is using the sampling algorithm of $\widetilde{\mathscr{F}}$ to get a triple (pk_1,α,β). The adversary $\mathcal{B}$ sends (pk_1,α,β,c) to $\mathcal{A}|_{\widetilde{\mathscr{F}}}$ and outputs the bit received from it. It is clear that

$$\mathrm{Adv}[\mathcal{B}](\lambda)=\mathrm{Adv}[\mathcal{A}|_{\widetilde{\mathscr{F}}}](\lambda)$$

so that $\mathcal{B}$ breaks the IND-CPA security of $\mathscr{S}_1$ with non-negligible advantage, and this contradicts our assumption.

4 A general recipe for constructing bridges

As we shall explain in what follows, the *Recrypt* algorithm, used in the boostrapping procedure that transforms a somewhat homomorphic encryption scheme into a fully homomorphic encryption scheme (see [17]), can be adapted to our situation in order to give a general recipe for the construction of a bridge. We will call this method *Gentry's recipe* and say that the bridges obtained using it are of *Gentry type*.

Let us consider an encryption scheme

$$\mathscr{S}=(\mathscr{P}_{\mathscr{S}},\mathscr{C}_{\mathscr{S}},\mathrm{KeyGen}_{\mathscr{S}},\mathrm{Enc}_{\mathscr{S}},\mathrm{Dec}_{\mathscr{S}})$$

and a homomorphic encryption scheme

$$\mathscr{H}=(\mathscr{P}_{\mathscr{H}},\mathscr{C}_{\mathscr{H}},\mathrm{KeyGen}_{\mathscr{H}},\mathrm{Enc}_{\mathscr{H}},\mathrm{Dec}_{\mathscr{H}},\mathrm{Eval}_{\mathscr{H}}),$$

such that $\mathscr{P}_{\mathscr{H}}$ has a ring structure and there exists an injective map $\iota:\mathscr{P}_{\mathscr{S}}\hookrightarrow\mathscr{P}_{\mathscr{H}}$ satisfying the properties 1.(a)-(b) in Definition 3.

In this construction, the key generation algorithm is as follows. First, it runs $\text{KeyGen}_{\mathscr{S}}(1^\lambda)$ to sample from the distribution $SK_{\mathscr{S}}$ and then, independently, it runs $\text{KeyGen}_{\mathscr{H}}(1^\lambda)$ to sample from $SK_{\mathscr{H}}$. We point out that the distribution SK_2 in the definition of the bridge is in fact the product $SK_{\mathscr{S}} \times SK_{\mathscr{H}}$ and the map $SK_2 \to SK_1$ (see Figure 1) is the projection on the first component $SK_{\mathscr{S}} \times SK_{\mathscr{H}} \to SK_{\mathscr{S}}$. Samples for the public keys $pk_{\mathscr{S}}$ and $pk_{\mathscr{H}}$ are generated, independently, using the key generation algorithms of the two schemes. Given a quadruple $(sk_{\mathscr{S}}, pk_{\mathscr{S}}, sk_{\mathscr{H}}, pk_{\mathscr{H}})$ constructed as above, the algorithm creates bk as the vector of encryptions of all the bits of $sk_{\mathscr{S}}$ under $pk_{\mathscr{H}}$ (see below). This is how the distribution of bridge keys BK is obtained.

The PPT algorithm f mentioned in the third part of Definition 3 is in this case the homomorphic evaluation (in $\mathscr{H}$) of the algorithm $\text{Dec}_{\mathscr{S}}$. We need to realise $\text{Dec}_{\mathscr{S}}$ as a map $\mathscr{P}_{\mathscr{H}}^{\ell} \to \mathscr{P}_{\mathscr{H}}$, and for this we use the ring structure on $\mathscr{P}_{\mathscr{H}}$. Suppose that the ciphertext space $\mathscr{C}_{\mathscr{S}}$ is a subset of $\{0,1\}^n$ and that the set of secret keys is a subset of $\{0,1\}^e$, so that $\text{Dec}_{\mathscr{S}} : \{0,1\}^e \times \{0,1\}^n \to \mathscr{P}_{\mathscr{S}}$. We construct the map $\widetilde{\text{Dec}}_{\mathscr{S}} : \mathscr{P}_{\mathscr{H}}^e \times \mathscr{P}_{\mathscr{H}}^n \to \mathscr{P}_{\mathscr{H}}$ as follows. Letting $\mathscr{P}_{\mathscr{H}}$ be a subset of $\{0,1\}^m$, we have that $\iota \circ \text{Dec}_{\mathscr{S}} : \{0,1\}^e \times \{0,1\}^n \to \mathscr{P}_{\mathscr{H}}$ is a vector $(g_1, ..., g_m)$ of boolean circuits expressed using XOR and AND gates. Let $\tilde{g}_i : \mathscr{P}_{\mathscr{H}}^e \times \mathscr{P}_{\mathscr{H}}^n \to \mathscr{P}_{\mathscr{H}}$ be the circuit obtained by replacing each $\text{XOR}(x,y)$- gate by $x \oplus y := 2(x+y) - (x+y)^2$ and each $\text{AND}(x,y)$ gate by $x \otimes y := x \cdot y$, where $+$ and $\cdot$ are the addition and multiplication in $\mathscr{P}_{\mathscr{H}}$. Notice that the subset of $\mathscr{P}_{\mathscr{H}}$ consisting of its zero element $0_{\mathscr{H}}$ and its unit $1_{\mathscr{H}}$ together with $\oplus$ and $\otimes$ is a realisation of the field with two elements inside $\mathscr{P}_{\mathscr{H}}$. In other words, if $c = (c[1], ..., c[n]) \in \mathscr{C}_{\mathscr{S}}$ and $sk_{\mathscr{S}} = (sk[1], ..., sk[e])$ is the secret key, then $\tilde{g}_i(sk[1]_{\mathscr{H}}, ..., sk[e]_{\mathscr{H}}, c[1]_{\mathscr{H}}, ..., c[n]_{\mathscr{H}}) = m_{\mathscr{H}}$ if $g_i(sk[1], ..., sk[e], c[1], ..., c[n]) = m$ for all i, where $m \in \{0,1\}$. For an element $x \in \mathscr{P}_{\mathscr{H}}$, we let $[x = 1_{\mathscr{H}}]$ be the equality test, which returns 1 if $x = 1_{\mathscr{H}}$ and 0 otherwise. Finally, $\widetilde{\text{Dec}}_{\mathscr{S}} : \mathscr{P}_{\mathscr{H}}^e \times \mathscr{P}_{\mathscr{H}}^n \to \{0,1\}^m$ is defined by:

$$([\tilde{g}_i(y_1, ..., y_e, x_1, ..., x_n) = 1_{\mathscr{H}}])_{i=\overline{1,m}}\,.$$

One can verify that

$$\widetilde{\text{Dec}}_{\mathscr{S}}(sk[1]_{\mathscr{H}}, ..., sk[e]_{\mathscr{H}}, c[1]_{\mathscr{H}}, ..., c[n]_{\mathscr{H}}) = \iota \circ \text{Dec}_{\mathscr{S}}(sk, c).$$

Now we are ready to define the bridge map. Given a ciphertext $c \in \mathscr{C}_{\mathscr{S}}$, the algorithm f first encrypts the n bits of c (viewed as elements of $\mathscr{P}_{\mathscr{H}}$) under $pk_{\mathscr{H}}$ and retains these encryptions in a vector $\tilde{c}$. The bridge key bk is obtained by encrypting the bits of $sk_{\mathscr{S}}$ under $pk_{\mathscr{H}}$. Then, the algorithm outputs:

$$f(bk, c) = \text{Eval}_{\mathscr{H}}(evk_{\mathscr{H}}, \widetilde{\text{Dec}}_{\mathscr{S}}, bk, \tilde{c})$$

Assuming that $\mathscr{H}$ can evaluate $\widetilde{\text{Dec}}_{\mathscr{S}}$ we have:

$$\begin{aligned}\text{Dec}_{\mathscr{H}}(f(bk, c)) &= \text{Dec}_{\mathscr{H}}\left(\text{Eval}_{\mathscr{H}}(evk_{\mathscr{H}}, \widetilde{\text{Dec}}_{\mathscr{S}}, bk, \tilde{c})\right) \\ &= \iota\left(\text{Dec}_{\mathscr{S}}(\text{Dec}_{\mathscr{H}}(bk), \text{Dec}_{\mathscr{H}}(\tilde{c}))\right) \\ &= \iota\left(\text{Dec}_{\mathscr{S}}(sk_{\mathscr{S}}, c)\right)\end{aligned}$$

which shows that third condition in the definition of a bridge is satisfied.

Remark 5. The above construction relies on the fact that the plaintext space of $\mathscr{H}$, being a ring, can be used to simulate an $\mathbb{F}_2$-structure inside it.

An example of the above construction can be found in [18], where the authors managed to homomorphically evaluate the AES-128 circuit (encryption and decryption) using an optimized implementation of the BGV scheme [7]. Once the plaintext spaces and the embedding ι are fixed, the evaluation of this decryption circuit can be seen as a Gentry type bridge. The bridge key consists of the BGV encryptions of the eleven AES round keys (see Section 4 of [18]). We note that here the round keys are embedded in the plaintext, so it was not necessary to encrypt the bits of the round keys, as discussed at the beginning of the section. This results in a simpler homomorphic evaluation of AES decryption. Nonetheless, this bridge is essentially obtained using Gentry's recipe.

4.1 On the security of Gentry type bridges

The aim of this subsection is to show that if $\mathscr{S}$ and $\mathscr{H}$ are IND-CPA secure, then any Gentry type bridge $B_{\iota,f}$ from $\mathscr{S}$ to $\mathscr{H}$ is IND-CPA secure. The plan is to apply Theorem 2 to the above construction.

Recall that $\mathscr{F}$ is the ensemble of finite distributions of triples $(pk_{\mathscr{S}}, pk_{\mathscr{H}}, bk)$, where bk is a vector of encryptions of the form $(bk[1], ..., bk[e])$ with $bk[i] \leftarrow \mathrm{Enc}_{\mathscr{H}}(pk_{\mathscr{H}}, sk[i]_{\mathscr{S}})$ for all i. Next, let $\widetilde{\mathscr{F}}$ be the ensemble of finite distributions of triples $(pk_{\mathscr{S}}, pk_{\mathscr{H}}, \widetilde{bk})$, where $pk_{\mathscr{S}}$, $pk_{\mathscr{H}}$ are independently outputted by $\mathrm{KeyGen}_{\mathscr{S}}$ and $\mathrm{KeyGen}_{\mathscr{H}}$, respectively and $\widetilde{bk} := (\widetilde{bk}[1], ..., \widetilde{bk}[e])$ with $\widetilde{bk}[i] \leftarrow \mathrm{Enc}(pk_{\mathscr{H}}, 0_{\mathscr{H}})$ for all $i \in \overline{1,e}$. Notice that $\widetilde{\mathscr{F}}$ is polynomial-time constructible on fibers as a $PK_{\mathscr{S}}$-ensemble of finite distributions (see Definition 1). Let us remark that one can choose $\widetilde{\mathscr{F}}$ in a different way, setting $\widetilde{bk}$ to be a vector of encryptions of any fixed e-long bit vector. If the scheme $\mathscr{H}$ is IND-CPA secure, then one can prove by a standard hybrid argument (see the next proposition) that the two versions are in fact computational indistinguishable. Therefore, the choice of the particular fixed bit vector that is encrypted to get $\widetilde{bk}$ does not matter.

Proposition 2. *If $\mathscr{H}$ is IND-CPA secure, then the ensembles $\mathscr{F}$ and $\widetilde{\mathscr{F}}$ are computationally indistinguishable.*

Proof. Let $\mathcal{D}$ be a distinguisher between the two ensembles $\mathscr{F}$ and $\widetilde{\mathscr{F}}$. We denote by $\mathscr{G}_i$ the distribution of triples $(pk_{\mathscr{S}}, pk_{\mathscr{H}}, x)$ where the pair $(pk_{\mathscr{S}}, pk_{\mathscr{H}})$ is chosen exactly as in the case of $\mathscr{F}$, or $\widetilde{\mathscr{F}}$, and $x := (x[1], ..., x[e])$ where $x[j] \leftarrow \mathrm{Enc}(pk_{\mathscr{H}}, sk_{\mathscr{S}}[j])$ for all $j \in \overline{1,i}$ and $x[j] \leftarrow \mathrm{Enc}(pk_{\mathscr{H}}, 0)$ for all $j \in \overline{i+1,e}$. Notice that $\{\mathscr{G}_{e(\lambda)}\}_\lambda$ is the same as $\mathscr{F}$, and $\{\mathscr{G}_0\}_\lambda$ is $\widetilde{\mathscr{F}}$. For each $i \in \overline{1,e}$ we construct an attacker $\mathcal{B}_i$ on the scheme $\mathscr{H}$ as follows. The attacker receives from the challenger the triple $(1^\lambda, pk_{\mathscr{H}}, c)$, where c is either an encryption of 0 or an encryption of 1. The attacker uses $\mathrm{KeyGen}_{\mathscr{S}}$ to generate a pair $(sk_{\mathscr{S}}, pk_{\mathscr{S}})$

and then constructs an e-long vector y as follows: $y[j] \leftarrow \mathrm{Enc}(pk_{\mathscr{H}}, sk_{\mathscr{S}}[j])$ for $j < i$, $y[i] = c$, and $y[j] \leftarrow \mathrm{Enc}(pk_{\mathscr{H}}, 0)$ for $j > i$. Then the attacker $\mathcal{B}_i$ runs $\mathcal{D}(1^\lambda, pk_{\mathscr{S}}, pk_{\mathscr{H}}, y)$ and outputs $sk[i]$ if the answer received from $\mathcal{D}$ is $\mathscr{F}$ and 0 otherwise. Basically, $\mathcal{D}$ can be used as a distinguisher between the ensembles $\{\mathscr{G}_{i-1}\}_\lambda$ and $\{\mathscr{G}_i\}_\lambda$, which gives rise to $\mathcal{B}_i$. Notice that

$$\mathrm{Adv}^{\mathrm{IND-CPA}}[\mathcal{D}](\lambda) \leq \sum_{i=1}^{e(\lambda)} \mathrm{Adv}^{\mathrm{IND-CPA}}[\mathcal{B}_i](\lambda),$$

where we used the fact that the advantage of $\mathcal{B}_i$ is equal to the advantage of $\mathcal{D}$ as a distinguisher between $\mathscr{G}_i$ and $\mathscr{G}_{i-1}$. Since $\mathscr{H}$ is IND-CPA secure and $e(\lambda)$ is polynomial in λ, we get that $\mathcal{D}$ has negligible advantage.

The result of Proposition 2 combined with Theorem 2 yields the following result:

Theorem 3. *Assume that $\mathscr{S}$ and $\mathscr{H}$ are both IND-CPA secure, then any Gentry type bridge $\mathbf{B}_{\iota,f}$ from $\mathscr{S}$ to $\mathscr{H}$ is IND-CPA secure.*

4.2 A variant of Gentry's recipe

The aim of this subsection is to give a new variant of Gentry's recipe for the construction of bridges. For this, we need first to introduce the product of two encryption schemes. Suppose that $\mathscr{S}_i = (\mathscr{P}_i, \mathscr{C}_i, \mathrm{KeyGen}_i, \mathrm{Enc}_i, \mathrm{Dec}_i)$, $i \in \{1, 2\}$ are two encryption schemes, then the product $\mathscr{S}_1 \times \mathscr{S}_2$ is defined as follows. The plaintext space is defined as $\mathscr{P}_1 \times \mathscr{P}_2$ and the ciphertext space as $\mathscr{C}_1 \times \mathscr{C}_2$. The Key Generation algorithm of the product scheme uses independently the key generation algorithms of the two schemes to produce two pairs (sk_1, pk_1) and (sk_2, pk_2) of keys and sets the secret key as (sk_1, sk_2), and sets the public key as (pk_1, pk_2). An encryption of a message $(m_1, m_2) \in \mathscr{P}_1 \times \mathscr{P}_2$ is just a pair (c_1, c_2), where $c_1 \leftarrow \mathrm{Enc}_1(pk_1, m_1)$ and $c_2 \leftarrow \mathrm{Enc}_2(pk_2, m_2)$. Finally, the decryption of (c_1, c_2) is $(\mathrm{Dec}_1(sk_1, c_1), \mathrm{Dec}_2(sk_2, c_2))$. In the same way, one can define the product of $p \geq 2$ encryption schemes. If $\mathscr{H}$ is an encryption scheme, we shall denote by $\mathscr{H}^p$ the product of p copies of $\mathscr{H}$.

Now, we describe this new construction. We use the same notations as in the beginning of this section, and we assume that $\mathscr{P}_{\mathscr{H}} = \{0, 1\}$. Let $\iota : \mathscr{P}_{\mathscr{S}} \hookrightarrow \{0, 1\}^p$ be a representation of the plaintext space of $\mathscr{S}$, which can be viewed as the map $\iota : \mathscr{P}_{\mathscr{S}} \hookrightarrow \mathscr{P}_{\mathscr{H}}^p$, by identifying $\{0, 1\}^p$ with the plaintext space of $\mathscr{H}^p$. We construct a bridge from $\mathscr{S}$ to $\mathscr{H}^p$ as follows. Notice that the decryption algorithm of $\mathscr{S}$ is in fact a p-long vector of boolean algorithms $g_i : \{0, 1\}^e \times \{0, 1\}^n \to \{0, 1\}$, that is $\mathrm{Dec}_{\mathscr{S}}(sk_{\mathscr{S}}, c) = (g_1(sk_{\mathscr{S}}, c), ..., g_p(sk_{\mathscr{S}}, c))$, where $\{0, 1\}^n$ and $\{0, 1\}^e$ correspond to $\mathscr{C}_{\mathscr{S}}$ and the support of secret keys of $\mathscr{S}$, respectively. The bridge key bk is obtained by encrypting the bits of $sk_{\mathscr{S}}$ under each component of the public key of $\mathscr{H}^p$.

The bridge map f is the vector obtained by homomorphically evaluating the circuits g_i in $\mathscr{H}$. More precisely

$$f(bk, c) = \left(\mathrm{Eval}_{\mathscr{H}}(evk_{\mathscr{H}}, g_i, bk, \tilde{c})\right)_{i=\overline{1,p}},$$

where $\tilde{c}$ is defined as above. Notice that, if $c \leftarrow \mathrm{Enc}(pk_{\mathscr{S}}, m)$ then

$$\mathrm{Dec}_{\mathscr{H}^p}(f(bk, c)) = \iota(m).$$

Security. As in the previous subsection, it can be shown that if $\mathscr{S}$ and $\mathscr{H}$ are IND-CPA secure, then the bridge is also IND-CPA secure. The proof is very similar to that of Proposition 2, hence omitted here.

5 Various Gentry type bridges

The aim of this section is to emphasize the fact that, for an encryption scheme $\mathscr{S}$, different representations for the decryption algorithm $\mathrm{Dec}_{\mathscr{S}}$ give rise to different bridges from $\mathscr{S}$ to a FHE scheme $\mathscr{H}$. For practical applications, one can select the appropriate representation that best suits the implementation of the desired application. Having this in mind, we chose to exhibit the encryption scheme CSGN introduced in [4] and implemented in [10], whose decryption algorithm admits at least four fundamentally different representations. We shall restrict ourselves in discussing the security of these bridges, because the security of the CSGN scheme is not entirely understood.

5.1 Description of the CSGN scheme

We give a brief description of the CSGN scheme. For more details regarding the parameter selection, we refer to [4]. The plaintext space is the field $\mathbb{F}_2$ and the ciphertext space of this scheme is $\mathbb{F}_2^n$ with the monoid structure defined by component-wise multiplication. A simplified version of the scheme is defined as follows.

- $\mathrm{KeyGen}_{\mathrm{CSGN}}(1^\lambda)$: Choose dimension parameters n, d and s of size $\mathrm{poly}(\lambda)$, a uniformly random subset S of $\{1, 2, \ldots, n\}$ of size s, and a finite distribution X on $\{1, 2, ..., d\}$ according to [4]. Set the secret key sk to be the characteristic function of S, viewed as a bit vector.
- $\mathrm{Enc}_{\mathrm{CSGN}}$: To encrypt 0, choose first $k \in \{1, 2, ..., d\}$ according to X and then choose uniformly at random d numbers $i_1, \ldots, i_d$ from the set $\{1, 2, \ldots, n\}$, such that exactly k of them are in S. Finally, output the vector in $\mathbb{F}_2^n$ whose components corresponding to the indices $i_1, \ldots, i_d$ are equal to 0 and the others are equal to 1. To encrypt 1, choose uniformly at random d numbers $i_1, \ldots, i_d$ from the set $\{1, 2, \ldots, n\}$, such that none of them is in S, and output the resulting vector in $\mathbb{F}_2^n$ as before.
- $\mathrm{Dec}_{\mathrm{CSGN}}$: To decrypt a ciphertext c using the secret key sk, output 0 if c has at least one component equal to 0 corresponding to an index from S and 1, otherwise.

The output of the decryption algorithm can be written as

$$\mathrm{Dec}_{\mathrm{CSGN}}(sk, c) = \prod_{i \in S} c_i.$$

Notice that, the decryption map is a homomorphism of monoids from $(\mathbb{F}_2^n, \cdot)$ to the monoid $(\mathbb{F}_2, \cdot)$ with the usual multiplication.

In what follows, we present four variants of bridges from the CSGN scheme, denoted by $\mathscr{S}$, to various FHE schemes. The latter are going to be denoted by $\mathscr{H}$. Also, the pairing $\langle \cdot, \cdot \rangle : R^n \times R^n \to R$ will always be the standard inner product over the ring R.

5.2 1st bridge

Let $\mathscr{H}$ be any FHE scheme with plaintext space the field with two elements; hence, the map ι is the identity map. The secret key $sk_{\mathscr{S}}$ can be represented by the n-dimensional standard vectors e_i, where $i \in S$. The bridge key generation algorithm encrypts each entry of the vectors e_i, $i \in S$ using $pk_{\mathscr{H}}$ to obtain the bridge key $bk = \{\widetilde{e_1}, ..., \widetilde{e_s}\}$, a set of vectors consisting of the aforementioned encryptions.

We remark that the decryption algorithm of $\mathscr{S}$ may be written as

$$\mathrm{Dec}_{\mathscr{S}}(sk_{\mathscr{S}}, c) = \prod_{i \in S} \langle c, e_i \rangle,$$

so that the bridge algorithm f is as follows:

$$f(bk, c) = \prod_{i=1}^{s} \langle c, \widetilde{e_i} \rangle = \prod_{i=1}^{s} \left(\sum_{c[j]=1} \tilde{e_i}[j] \right).$$

For simplicity, we chose the trivial encryptions as the encryptions of the bits of c with $\mathscr{H}$.

5.3 2nd bridge

We are in the same setting as before, where both plaintext spaces are $\mathbb{F}_2$. Recall that the secret key $sk_{\mathscr{S}}$ is the characteristic function of the set S, represented as an n-dimensional bit vector. Then, the decryption of $\mathscr{S}$ can be alternatively written as

$$\mathrm{Dec}_{\mathscr{S}}(sk_{\mathscr{S}}, c) = \prod_{i=1}^{n} \Big(1 - (1 - c[i]) sk_{\mathscr{S}}[i]\Big) = \prod_{c[i]=0} (1 - sk_{\mathscr{S}}[i]).$$

The bridge key bk is constructed as $bk := \{\widetilde{sk_{\mathscr{S}}}[1], ..., \widetilde{sk_{\mathscr{S}}}[n]\}$, where for every i, $\widetilde{sk_{\mathscr{S}}}[i]$ is an encryption of $1 - sk_{\mathscr{S}}[i]$ under $pk_{\mathscr{H}}$. Finally, the bridge is given by

$$f(bk, c) = \prod_{c[i]=0} \widetilde{sk_{\mathscr{S}}}[i].$$

Remark 6. The last formula shows that this bridge can be constructed even if the scheme $\mathscr{H}$ is homomorphic only with respect to multiplication. For example, it can be used when $\mathscr{H} = \mathscr{S}$ obtaining something that resembles the key-switching technique in some FHE schemes.

5.4 3rd bridge

Here, the scheme $\mathscr{H}$ can be any FHE scheme with plaintext space the finite field $\mathbb{F}_p$, where p is a prime (for example the BGV and B/FV schemes, see [7], [8] and [14]).

The bridge key generation algorithm instantiates $\mathrm{KeyGen}_{\mathscr{S}}(1^\lambda)$ and then $\mathrm{KeyGen}_{\mathscr{H}}(1^\lambda)$, assuring that the characteristic of $\mathscr{P}_{\mathscr{H}}$ is larger than the Hamming weight of $sk_{\mathscr{S}}$, that is $p > s$. It then chooses positive integers $x_1, ..., x_s$ such that $p = 1 + x_1 + \cdots + x_s$, and fixes a bijection $\varphi : S \to \{1, ..., s\}$. Consider the vector $sk \in \mathbb{F}_p^n$, where $sk[i] = 0$ if $sk_{\mathscr{S}}[i] = 0$ and $sk[i] = x_{\varphi(i)}$, otherwise. For every $i \in \{1, \ldots, n\}$, write $\widetilde{sk}[i]$ for an encryption of $sk[i]$ under $pk_{\mathscr{H}}$. In this case, the bridge key bk is the set of $\mathscr{H}$ encryptions $bk = \{\widetilde{sk}[1], \ldots, \widetilde{sk}[n]\}$.

We remark that if $\iota : \mathbb{F}_2 \hookrightarrow \mathbb{F}_p$ denotes the usual embedding, then the decryption of $\mathscr{S}$ satisfies

$$\mathrm{Dec}_{\mathscr{S}}(sk_{\mathscr{S}}, c) = \iota^{-1}\Big(1 - \big(1 + \langle\, c, sk\rangle_{\mathbb{F}_p}\big)^{p-1}\Big).$$

The bridge map is defined as

$$f(bk, c) = \mathrm{Enc}_{\mathscr{H}}(pk_{\mathscr{H}}, 1) - \left(\mathrm{Enc}_{\mathscr{H}}(pk_{\mathscr{H}}, 1) + \sum_{c[i]=1} \widetilde{sk}[i]\right)^{p-1},$$

where the additions, subtractions and exponentiation on the right hand side are homomorphic operations on the ciphertexts of $\mathscr{H}$.

Remark 7. As mentioned in the discussion following Definition 3, one can develop a theory of bridges for which the plaintext spaces of the two encryption schemes vary with λ along the same lines. The bridge constructed here falls in this category because the plaintext space of $\mathscr{H}$ is chosen after the size of the secret key is selected, as part of the Setup/KeyGen algorithm.

5.5 4th bridge

This bridge is based on an idea used in [2] for the bootstrapping procedure of the GSW scheme. Notice that if c is a ciphertext in $\mathscr{S}$, encrypted using $pk_{\mathscr{S}}$, then c decrypts to 1 if and only if the inner product $\langle c, sk_{\mathscr{S}}\rangle_{\mathbb{Z}} = s$, namely

$$\mathrm{Dec}_{\mathscr{S}}(sk_{\mathscr{S}}, c) = [\langle\, c, sk_{\mathscr{S}}\rangle_{\mathbb{Z}} = s],$$

where $[x = y]$ is, as before, the equality test.

We observe that in the computation of the inner product $\langle\, c, sk_{\mathscr{S}}\rangle_{\mathbb{Z}}$ one uses only the additive structure of $\mathbb{Z}$ (also $\mathbb{Z}_m$ with $m > s$ would be sufficient for our purposes). To find a representation of the cyclic group $(\mathbb{Z}_m, +)$, one needs first to embed it into the symmetric group $\mathfrak{S}_m$. The generator $1 \in \mathbb{Z}_m$ is sent by this injective homomorphism to the cyclic permutation $\pi_1 \in \mathfrak{S}_m$, defined as $\pi_1(i) = i + 1$ for $1 \leq i < m$ and $\pi_1(m) = 1$. On the other hand, the group $\mathfrak{S}_m$ is

isomorphic to the multiplicative group of m-by-m permutation matrices, that is matrices with 0 or 1 entries, having exactly one nonzero element in each row and each column. The isomorphism maps the permutation $\pi \in \mathfrak{S}_m$ to the matrix $M_\pi = [e_{\pi(1)}, ..., e_{\pi(m)}]$, where $e_i \in \{0,1\}^m$ is the i^{th} standard basis vector. The composition of these two homomorphisms gives us an embedding for the cyclic group $(\mathbb{Z}_m, +)$. For implementation purposes, it is good to notice that the permutation matrices in the image of this embedding can be represented more compactly by just their first column, because the remaining columns are just the successive cyclic shifts of this column.

Let us explain how the bridge is constructed. Let $m = s + 1$ and take $sk = (sk[1], ..., sk[n])$ to be the aforementioned representation of the secret key $sk_{\mathscr{S}}$, that is $sk[i] = M_{\pi_1}$ if $sk_{\mathscr{S}} = 1$ and $sk[i]$ is the identity matrix otherwise. Set $\widetilde{sk}[i]$ to be an encryption of $sk[i]$ under $pk_{\mathscr{H}}$ for all $i \in \overline{1,n}$, meaning that we encrypt with $\mathscr{H}$ each entry of the matrix $sk[i]$. The bridge key bk consists of $\{\widetilde{sk}[1], \ldots, \widetilde{sk}[n]\}$.

The algorithm f takes as input bk and c and computes the matrix

$$P^c := \prod_{c[i]=1} \widetilde{sk}[i],$$

where the right hand side is a product of encrypted matrices, performed homomorphically in $\mathscr{C}_{\mathscr{H}}$. We remark that the last entry of the first row of P^c is an encryption of the value returned by the equality test $[\langle c, sk_{\mathscr{S}} \rangle_{\mathbb{Z}} = s]$. Consequently, we let the output of the bridge map be

$$f(bk, c) := P^c_{1,s+1}.$$

6 A bridge not of Gentry type

In this section we give an example of a bridge that does not follow Gentry's recipe. We start by recalling the Goldwasser-Micali and Sander-Young-Yung encryption schemes. A bridge from the former to the latter is then presented. The section ends with an interesting application of this bridge.

6.1 Goldwasser-Micali Cryptosystem

The Goldwasser-Micali encryption scheme is an asymmetric key encryption algorithm developed by Shafi Goldwasser and Silvio Micali in [21]. If p, q are two primes and $N = p \cdot q$, then let $J_1(N) := \{x \in (\mathbb{Z}/N\mathbb{Z})^\times | \left(\frac{x}{N}\right) = 1\}$ be the multiplicative group of invertible integers modulo N with Jacobi symbol equal to 1. The GM-encryption scheme $(\mathbb{Z}/2\mathbb{Z}, J_1(N), \text{KeyGen}_{GM}, \text{Enc}_{GM}, \text{Dec}_{GM})$ is given as follows:

- **KeyGen**(1^λ): Choose two primes $p = p(\lambda), q = q(\lambda)$ of size λ and let $N = pq$. Choose $\eta \in (\mathbb{Z}/N\mathbb{Z})^\times$ such that $\left(\frac{\eta}{p}\right) = \left(\frac{\eta}{q}\right) = -1$, which yields that $\eta \in J_1(N)$. The public key is the pair $(N, \gamma := \eta \cdot u^2)$, where u is a random element of $(\mathbb{Z}/N\mathbb{Z})^\times$. The secret key is the pair (p, q).

- **Enc**: To encrypt $m \in \mathbb{Z}/2\mathbb{Z}$, choose a random $\xi \in \mathbb{Z}/N\mathbb{Z}$ and let $\mathrm{Enc}_{GM}(m) = \gamma^m \xi^2$.
- **Dec**: To decrypt $c \in J_1(N)$, compute the Jacobi symbol $\left(\frac{c}{p}\right)$. Set $\mathrm{Dec}_{GM}(c) = 0$ if the answer is 1 and $\mathrm{Dec}_{GM}(c) = 1$ if the answer is -1.

The GM-encryption scheme is homomorphic with respect to addition in $\mathbb{Z}/2\mathbb{Z}$ and multiplication in $J_1(N)$, i.e.

$$\mathrm{Dec}_{GM}(c_1 \cdot c_2) = \mathrm{Dec}_{GM}(c_1) + \mathrm{Dec}_{GM}(c_2)$$

for all $c_1, c_2 \in J_1(N)$.

6.2 The Sander-Young-Yung Cryptosystem

In this section we present a homomorphic encryption scheme over the multiplicative monoid $(\mathbb{Z}/2\mathbb{Z}, \cdot)$ introduced in [27]. To describe the scheme we shall use the encryption scheme of Goldwasser-Micali, which was recalled in the previous section.

- **Keygen**(1^λ): Choose two primes $p = p(\lambda)$, $q = q(\lambda)$ as in the Goldwasser-Micali scheme. Choose $\ell = \ell(\lambda)$ of size $\Theta(\lambda)$. Compute $N = pq$. The public key and secret keys are the same as in the Goldwasser-Micali scheme.
- **Enc**: If $m = 1$ set $v = (0, ..., 0) \in \{0,1\}^\ell$. If $m = 0$ set $v = (v_1, ..., v_n) \in \{0,1\}^\ell$, where the components v_i are randomly chosen in $\{0,1\}$, not all equal to 0. Encrypt each component of v with the Goldwasser-Micali scheme to get a vector in $\mathscr{C}_{SYY} := J_1(N)^\ell$.
- **Dec**: To recover the plaintext from the cyphertext $c \in \mathscr{C}$, first decrypt each component of c using the decryption algorithm of the Goldwasser-Micali scheme, and then if the obtained vector is the 0-vector the message decrypts to 1, else to 0.

Let us describe an operation $\odot$ on the ciphertext space $\mathscr{C}_{SYY}$. If x and y are two ciphertexts then $z := x \odot y$ is defined as follows:

1. Choose uniformly at random two $\ell \times \ell$ matrices over $\mathbb{Z}/2\mathbb{Z}$ until two nonsingular matrices $A = (a_{ij})$ and $B = (b_{ij})$ are found.

2. If $x = (x_1, ..., x_\ell)$, $y = (y_1, ..., y_\ell)$, then compute

$$z_i = \prod_{j, a_{ij}=1} x_j \cdot \prod_{j, b_{ij}=1} y_j$$

for all i.

3. Pick uniformly at random $r_1, ..., r_\ell \in (\mathbb{Z}/N\mathbb{Z})^\times$ and set $z = (z_1 r_1^2, ..., z_\ell r_\ell^2)$.

Let us denote by v_c the bit vector obtained by applying the decryption algorithm of the Goldwasser-Micali scheme componentwise to the ciphertext $c \in \mathscr{C}$. If $z := x \odot y$ then Step 2 above is equivalent to:

$$v_z = Av_x + Bv_y,$$

where the operations are the usual addition and multiplication in $\mathbb{Z}/2\mathbb{Z}$. Notice that $\mathrm{Dec}_{SYY}(z) \neq \mathrm{Dec}_{SYY}(x) \cdot \mathrm{Dec}_{SYY}(y)$ if and only if $Av_x + Bv_y = \vec{0}$ (here $\vec{0}$ is the zero vector in $(\mathbb{Z}/2\mathbb{Z})^\ell$), and $v_x \neq \vec{0}$, $v_y \neq \vec{0}$. Since $v_x \neq \vec{0}$ and A is nonsingular, the product Av_x can be any nonzero vector in $(\mathbb{Z}/2\mathbb{Z})^\ell$, and in fact any such vector occurs with the same probability. Of course, the same is true for Bv_y such that the situation described above occurs with probability $\leq \frac{1}{2^\ell}$. In other words, except with exponentially small probability, we have that

$$\mathrm{Dec}_{SYY}(x \odot y) = \mathrm{Dec}_{SYY}(x) \cdot \mathrm{Dec}_{SYY}(y).$$

6.3 A bridge from GM to SYY

In this section we construct a bridge from the Goldwasser-Micali encryption scheme to the Sander-Young-Yung encryption scheme. After generating a secret key (p, q) of GM, the key generation algorithm of the bridge sets the same pair (p, q) as the secret key for the SYY encryption scheme. Then, the public keys for the two encryption schemes are generated independently using their respective key generation algorithms. After that, the bridge key generation algorithm does not output anything, i.e. the support of the distribution BK is the empty set.

Now, for $c \in J_1(N)$, choose uniformly at random a non-singular matrix $A \in \mathrm{GL}_\ell(\mathbb{Z}/2\mathbb{Z})$ and compute

$$t_i = \prod_{j, a_{ij}=1} c\gamma' = (c\gamma')^{|\{j|a_{ij}=1\}|}$$

for all $i \in \overline{1,\ell}$, where γ' is the second component of the public key of the SYY scheme. Pick uniformly at random $r_1, \ldots, r_\ell \in (\mathbb{Z}/N\mathbb{Z})^\times$ and set

$$f(c) = (t_1 r_1^2, ..., t_\ell r_\ell^2).$$

If $\mathrm{Dec}_{GM}(c) = 1$, then $\mathrm{Dec}_{GM}(c\gamma') = 0$ so that $\mathrm{Dec}_{GM}(t_i) = 0$, $\forall i$. Therefore, $v_{f(c)} = \vec{0}$ and hence $\mathrm{Dec}_{SYY}(f(c)) = 1$. On the other hand, if $\mathrm{Dec}_{GM}(c) = 0$, then $\mathrm{Dec}_{GM}(c\gamma') = 1$, and since A is nonsingular there exist $i \in \overline{1,\ell}$ such that $\mathrm{Dec}_{GM}(t_i) = 1$. We get that $v_{f(c)} \neq \vec{0}$, equivalently $\mathrm{Dec}_{SYY}(f(c)) = 0$.

Remark 8. The security of this bridge reduces to the security of the GM scheme (see [21]) using Theorem 2. Indeed, the bridge key distribution is empty, thus trivially polynomial-time constructible on fibers. On the other hand, the security of SYY encryption scheme can be easily reduced to the security of GM (see [27]). Alternatively, one can use Theorem 1 instead of 2. To see this, note that in the notation of Section 3, the public key of the scheme attached to this bridge $PK_{\mathscr{G}_f}$ consists of just GM's public key and the security of $GM[PK_{\mathscr{G}_f}]$ is equivalent to the security of GM.

6.4 An application

As an application of the above bridge we show that the comparison circuit can be evaluated homomorphically. For this, let $\vec{x} = (x_1, x_2, ..., x_n)$ and $\vec{y} = (y_1, y_2, ..., y_n)$ be two bit vectors. The two vectors coincide if and only if

$$(x_1 + y_1 + 1) \cdot ... \cdot (x_n + y_n + 1) = 1,$$

so that the comparison circuit $[\vec{x} = \vec{y}]$ is defined by

$$[\vec{x} = \vec{y}] := (x_1 + y_1 + 1) \cdot ... \cdot (x_n + y_n + 1).$$

Suppose now that $\vec{c} = (c_1, ..., c_n)$ and $\vec{d} = (d_1, ..., d_n)$ are encryptions of the vectors $\vec{x}, \vec{y}$ with the Goldwasser-Micali cryptosystem. To homomorphically evaluate the comparison circuit, we compute:

$$\text{Eval}([\vec{x} = \vec{y}], \vec{c}, \vec{d}\,) := \Big(\Big(\big(f(c_1 \cdot d_1 \cdot \gamma) \odot f(c_2 \cdot d_2 \cdot \gamma) \big) \odot ... \Big) \odot f(c_n \cdot d_n \cdot \gamma) \Big).$$

Notice that $\text{Dec}_{SYY}\Big(\text{Eval}([\vec{x} = \vec{y}], \vec{c}, \vec{d}\,)\Big) = [\vec{x} = \vec{y}]$, except with negligible probability in the security parameter.

We end this section with the following reflection. When two encryption schemes admit the construction of a bridge which has an empty bridge key, this may be interpreted as some sort of entanglement between the schemes. Along the same line of thought, if one can prove that such a bridge cannot be constructed, the encryption schemes may be regarded as being independent.

Appendix

We conducted experiments for the bridges described in sections 5 and 6. For each of the four different bridges in section 5, we compare the results of the homomorphic evaluation of a circuit consisting of only one monomial in the following two ways. First, we encrypt each factor of the monomial and perform the homomorphic multiplications of these factors using the CSGN scheme. Then, bridges described in section 5 are applied, in turn, to obtain a ciphertext in a fully (leveled) homomorphic encryption scheme based on (R)LWE. We compare this to the alternative option of evaluating the monomial directly on encryptions in the FHE scheme. If the degree of the monomial is larger than a certain threshold, the first procedure outperforms the second in terms of speed. We identified this threshold for each of the FHE schemes in which we performed experiments.

These computations were carried on a virtual machine having an Intel CPU (I7-4770, 4 cores, 3.4 GHz, 12 GB RAM), using a single threaded implementation. Table 1 consists of an overview of the processing times for each bridge using the implementations of BGV, BFV and TFHE schemes, namely the HElib [22], SEAL [24] and TFHE [11] software libraries. In the first two columns of the table, one can find the version of the bridge that was implemented, the FHE

target scheme and the security parameters for the two schemes. The timings are measured such that all encryptions maintain approximately the same security level λ and listed in the last two columns. The small variation in λ is due to parameter tuning in the different software libraries.

Table 1. Bridge evaluation

Bridge (CSGN-λ)	**LWE (λ)**	**ENC (Bridge key)**	**Bridge time**
$1^{\text{st}}(125)$	BGV(121)	69 *sec*	2.6 *sec*
$1^{\text{st}}(125)$	TFHE(128)	186 *ms*	38.33 *sec*
$1^{\text{st}}(125)$	BFV(128)	38.97 *sec*	209.95 *ms*
$2^{\text{nd}}(125)$	BGV(114)	14.6 *sec*	68.28 *sec*
$2^{\text{nd}}(125)$	TFHE(128)	2.94 *ms*	1049 *ms*
$2^{\text{nd}}(125)$	BFV(128)	698 *ms*	2.24 *sec*
$3^{\text{rd}}(120)$	BGV(145)	7.65 *sec*	248 *ms*
$3^{\text{rd}}(120)$	BFV(128)	8.2 *sec*	156.46 *ms*
$4^{\text{th}}(115)$	TFHE(128)	162.6 *ms*	989.4 *sec*

The reason we are missing an implementation for our third bridge using the TFHE library comes from the lack of flexibility in choosing as plaintext space a ring of characteristic $p > 2$ in this library. Additionally, we felt that adapting the TFHE library was beyond the scope of our work. Also, the timing for running the fourth bridge in BGV and BFV could not be measured because of large memory usage, which exceeded the virtual machine RAM. Moreover, regarding the fourth bridge, the implementation is optimized to store only the first column of each associated bit in the secret key, while the matrix multiplications involve only homomorphic algebraic operations on encryptions from the first column of the matrices.

There is no doubt that homomorphically evaluating a circuit whose polynomial representation has a large number of monomials of low degree using the bridge is inefficient and there is little hope for optimizations in terms of speed. However, if some monomials have large degree, one might choose to do so, because first performing multiplications in the CSGN scheme, followed by additions in the (R)LWE setting might result in lower noise growth. Moreover, by increasing the multiplicative depth of the circuit, we observe that its evaluation is faster using the bridge than evaluating the circuit entirely in the (R)LWE schemes. This can be observed in the figures below.

Since the multiplication in the CSGN scheme is inexpensive, the evaluation time in the bridge using BGV, BFV and TFHE is almost constant as it essentially consists only of the evaluation time of the bridge algorithm for one CSGN ciphertext. Small variations in execution time for the bridge are due to the CPU scheduling process. The drops in evaluation times occur when the instruction-specific and data-specific cache at different levels in the CPU is filled with numerous repetitive instructions. The timings for evaluating the circuit entirely in the BGV or BFV scheme grow linearly with the degree of the monomial. We notice

that in the TFHE case, the running time of the evaluation starts growing exponentially in the number of multiplications, at some point. This is explained by the fact that the TFHE software library goes automatically into bootstrapping, whereas in the HElib and SEAL software libraries we can choose parameters in which one can evaluate the circuit without the costly bootstrapping procedure.

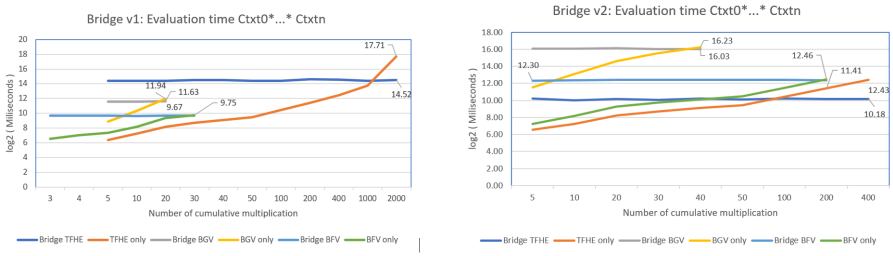


Fig. 2. The first and second bridges

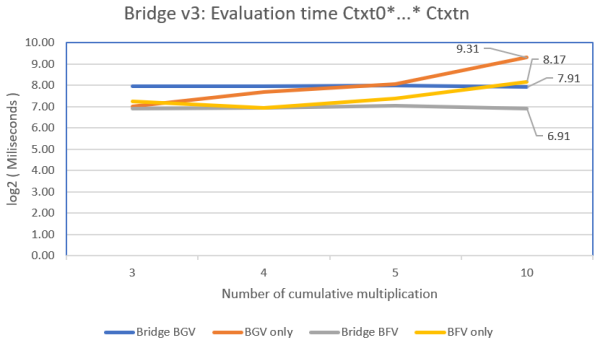


Fig. 3. The third bridge - BGV & BFV

We now report on the implementation of the bridge from the Goldwasser-Micali encryption scheme to the Sander-Young-Yung encryption scheme constructed in the last section. In the table below, one can find the timings required for running the bridge, as well as the ones needed for the homomorphic evaluation of the comparison circuit. The measurements were performed on an Intel I7-1068NG7 CPU laptop with 32GB of RAM. Since the parameter ℓ of the SYY scheme does not have an impact on the security, but rather on the probability to correctly decrypt the ciphertext $\left(\geq 1 - \frac{1}{2^\ell}\right)$, we fix ℓ to be 50.

Table 2. Homomorphic evaluation of comparison circuit using GM-SYY bridge

n	$\log_2(N)$	**GM** $\cdot$	**SYY** $\odot$	**GM** $\rightarrow$ **SYY**	$[\vec{x} = \vec{y}]$
4	1024	0.002 ms	10.02 ms	4.54 ms	58.35 ms
4	2048	0.003 ms	29.96 ms	11.64 ms	164.44 ms
4	4096	0.008 ms	84.01 ms	32.82 ms	467.43 ms
8	1024	0.003 ms	10.70 ms	4.77 ms	123.98 ms
8	2048	0.004 ms	30.12 ms	11.89 ms	336.27 ms
8	4096	0.008 ms	84.65 ms	33.46 ms	945.26 ms
16	1024	0.002 ms	10.8 ms	4.87 ms	251.44 ms
16	2048	0.004 ms	29.69 ms	11.55 ms	660.17 ms
16	4096	0.008 ms	85.49 ms	33.71 ms	1907.78 ms
32	1024	0.003 ms	10.4 ms	4.69 ms	484.10 ms
32	2048	0.004 ms	30.29 ms	11.82 ms	1348.44 ms
32	4096	0.009 ms	82.51 ms	32.34 ms	3576.41 ms

The parameters n and N in Table 2 stand for the bit-lengths of $\vec{x}, \vec{y}$ and, respectively, the Goldwaser-Micalli modulus. The timings required for the one homomorphic operation in each scheme can be found in the third and the fourth columns. We notice that the timings presented above grow linearly with the number of bits required to represent the input data. This can be observed in the following figure.

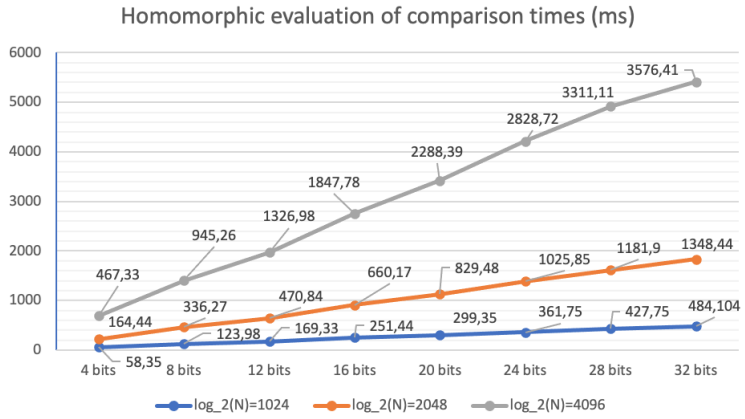


Fig. 4. Evaluation times for the comparison circuit using GM-SYY bridge

References

1. Acar, A., Aksu, H., Uluagac, A.S., Conti, M.: A Survey on Homomorphic Encryption Schemes: Theory and Implementation, ACM Computing Surveys, **51**(4), Article No. 79 (2018).
2. Alperin-Sheriff J., Peikert C.: Faster Bootstrapping with Polynomial Error. In: Garay J.A., Gennaro R. (eds) Advances in Cryptology, CRYPTO 2014, LNCS, vol 8616, pp. 297–314. Springer, Berlin, Heidelberg (2014).
3. Awodey, S.: Category theory, 2nd edn, Oxford University Press, Oxford (2010).
4. Barcau, M., Paşol, V., Pleşca, C.: Monoidal Encryption over $\mathbb{F}_2$, In: Lanet JL., Toma C. (eds) Innovative Security Solutions for Information Technology and Communications, SECITC 2018, LNCS, vol 11359, pp. 504–517, Springer, Cham (2019).
5. Boneh D., Goh EJ., Nissim K.: Evaluating 2-DNF Formulas on Ciphertexts. In: Kilian J. (eds) Theory of Cryptography, TCC 2005, LNCS, vol. 3378, pp. 325–341. Springer, Berlin, Heidelberg (2005).
6. Boura, C., Gama, N., Georgieva, M., Jetchev, D.: CHIMERA: Combining Ring-LWE-based Fully Homomorphic Encryption Schemes, Journal of Mathematical Cryptology **14**(1), pp. 316 – 338 (2020).
7. Brakerski, Z., Gentry, C., Vaikuntanathan, V.: (Leveled) fully homomorphic encryption without bootstrapping, ACM Transactions on Computation Theory **6**(3), No. 13, pp. 1–36 (2014).
8. Brakerski, Z.: Fully homomorphic encryption without modulus switching from classical GapSVP, In: Safavi-Naini R., Canetti R. (eds) Advances in Cryptology, CRYPTO 2012, LNCS, vol. 7417, pp. 868 – 886. Springer, Berlin, Heidelberg (2012).
9. Castagnos, G., Imbert, L., Laguillaumie, F.: Encryption Switching Protocols Revisited: Switching Modulo p, In: Katz J., Shacham H. (eds) Advances in Cryptology, CRYPTO 2017, LNCS, vol. 10401, pp. 255 – 287, Springer, Cham (2017).
10. certSIGN RD: CSGN GitHub repository, https://github.com/certFHE/CSGN. Last accessed on 20 May 2021.
11. Chillotti, I., Gama, N., Georgieva, M. and Izabachène, M.: TFHE: Fast Fully Homomorphic Encryptionover the Torus, Journal of Cryptology, **33** pp. 34 – 91 (2020).
12. Couteau, G., Peters, T., Pointcheval, D.: Encryption Switching Protocols, In: Robshaw M., Katz J. (eds) Advances in Cryptology, CRYPTO 2016, LNCS, vol. 9814, pp. 308 – 338. Springer, Berlin, Heidelberg (2016).
13. Doroz, Y., Hu, Y., Sunar, B.: Homomorphic AES Evaluation Using NTRU, IACR Cryptology ePrint Archive: Report 2014/039.
14. Fan, J., Vercauteren, F.: Somewhat Practical Fully Homomorphic Encryption, IACR Cryptol. ePrint Arch. 2012: 144 (2012).
15. Gentry, C: A fully homomorphic encryption scheme, PhD thesis, Stanford University, 2009.
16. Gentry, C: Fully homomorphic encryption using ideal lattices, In: STOC '09: Proceedings of the forty-first annual ACM symposium on Theory of computing, pp. 169 – 178, Association for Computing Machinery, New York, NY, United States (2009).
17. Gentry, C.: Computing arbitrary functions of encrypted data, Communications of the ACM, **53**(3), pp. 97 – 105 (2010).
18. Gentry C., Halevi S., Smart N.P.: Homomorphic Evaluation of the AES Circuit. In: Safavi-Naini R., Canetti R. (eds) Advances in Cryptology, CRYPTO 2012, LNCS, vol. 7417, pp. 850–867. Springer, Berlin, Heidelberg (2012).
19. Goldreich, O.: A note on computational indistinguishability, Information Processing Letters, **34**(6), pp. 277 – 281.

20. Goldwasser, S., Micali, S.: Probabilistic encryption and how to play mental poker keeping secret all partial information, In: STOC '82: Proceedings of the fourteenth annual ACM symposium on Theory of computing, pp. 365 – 377. Association for Computing Machinery, New York, NY (1982).
21. Goldwasser, S., Micali, S.: Probabilistic Encryption, Journal of Computer and System Sciences, **28**(2), pp. 270–299 (1984).
22. HElib library homepage: An Implementation of homomorphic encryption by Halevi and Shoup, https://github.com/shaih/HElib/.
23. Ishai, Y., Paskin, A.: Evaluating branching programs on encrypted data. In: Vadhan S.P. (eds) Theory of Cryptography, TCC 2007, LNCS, vol. 4392, pp. 575–594. Springer, Berlin, Heidelberg (2007).
24. Microsoft Research, Redmond, WA., Microsoft SEAL (release 3.6), https://github.com/Microsoft/SEAL, November, 2020.
25. Paillier, P.: Public-key cryptosystems based on composite degree residuosity classes. In: Stern J. (eds) Advances in Cryptology, EUROCRYPT 1999, LNCS, vol. 1592, pp. 223 – 238. Springer, Berlin, Heidelberg (1999).
26. Rivest, R., Adleman, L., Dertouzos, M. L.: On data banks and privacy homomorphisms, Foundation of Secure Computations, **4**(11), pp. 169–180 (1978).
27. Sander, T., Young, A., Yung, M.: Non-Interactive CryptoComputing For NC^1. In: FOCS '99: Proceedings of the 40th Annual Symposium on Foundations of Computer Science, pp. 554 – 566, IEEE Computer Society, NW Washington, DC, United States (1999).
28. Smart, N.: Cryptography Made Simple, Springer, Cham (2016).