Symmetric Somewhat Homomorphic Encryption over the Integers

Bogdan Kulynych

Abstract

We describe a symmetric variant of homomomorphic encryption scheme by van Dijk et al. [DGHV10], semantically secure under the error-free approximate-GCD problem. We also provide the implementation of the scheme as a C/C++ library. The scheme allows to perform "mixed" homomorphic operations on ciphertexts and plaintexts, eliminating the need to encrypt new ciphertexts using the public key for some applications, specifically, in secure function evaluation setting. Compared to the original scheme and other homomorphic encryption schemes, the properties of the symmetric variant enable for smaller communication cost for applications like privacy-preserving cloud computing, and private information retrieval.

Introduction

Fully homomorphic encryption (FHE) is a special kind of encryption that allows arbitrary computations on encrypted data. The first FHE scheme was shown to be possible in the breakthrough work by Gentry [Gen09] in 2009. A number of other FHE schemes based on different hardness assumptions were proposed since then [DGHV10, BGV12, Bra12, LTV12, FV12].

While the original scheme was based on ideal lattices [Gen09], van Dijk et al. proposed a new FHE scheme [DGHV10] over the integers in 2010. Both these schemes follow Gentry's blueprint to achieve fully homomorphic property. They are known as the *first generation* FHE.

The schemes produce similar "noisy" ciphertexts, where noise grows larger as more homomorphic operations are performed on a given ciphertext. When the noise reaches some maximum amount, the ciphertext becomes undecryptable. Following Gentry's approach, one first constructs a *somewhat homomorphic encryption* scheme, i.e. scheme that is capable of evaluating a limited amount of homomorphic operations before the ciphertext becomes undecryptable. Secondly, one defines *bootstrapping* procedure that eliminates the noise in ciphertexts (*ciphertext refreshing*). The procedure consists of homomorphic evaluation of the scheme's decryption circuit, which for a given ciphertext results in a different encryption of the same plaintext with reduced noise. Using bootstrapping, arbitrary binary circuits can be evaluated by refreshing the ciphertexts before the noise reaches the threshold. The disadvantage of this approach is that the bootstrapping procedure is very costly to perform.

To avoid bootstrapping, a new technique known as *modulus switching* was introduced by Brakerski, Gentry, and Vaikuntanathan in 2012 [BGV12]. They proposed a *leveled* FHE scheme, i.e. the one in which noise grows linearly with multiplicative depth of the circuit. Such scheme, however, has huge public key storage requirements. In 2012 Brakerski introduced the notion of *scale-invariance* for *leveled* FHE schemes [Bra12] that allows to reduce the storage significantly. This technique was applied to BGV scheme [BGV12] by Fan and Vercauteren in 2012 [FV12] resulting in FV scheme, and was used to construct a scheme called YASHE by Bos et al. in 2013 [BLLN13] based on work from [LTV12]. The mentioned schemes [BGV12, Bra12, FV12, BLLN13] as well as improved scheme by Gentry, Sahai, and Waters from 2013 [GSW13] based on [Bra12, BGV12] are known as *second generation* FHE.

Existing FHE mplementations One of the practical publicly available implementations of FHE is the C++ library implementing the BGV scheme [BGV12] with certain improvements [SV11, GHS12a] called `helib` [GHS12b]. There are also experimental implementations of the variant of DGHV scheme in Sage [CNT12], and FV and YASHE in C++ [LN14].

Our contributions In this work we focus on somewhat homomorphic DGHV scheme over the integers [DGHV10]. We notice that DGHV supports "mixed" homomorphic operations on ciphertexts and plaintexts, which in the context of secure function evaluation allows to eliminate the need for either the client or the remote worker to encrypt all of the inputs of the algorithm, implying symmetric setting. We then describe a symmetric variant of the DGHV scheme as seen in [YKPB13], propose secure parameter constraints, and provide its usable implementation as a C/C++ library.

Preliminaries

Generic Homomorphic Encryption Scheme

Whereas a regular public-key encryption scheme $\mathcal{E}$ consists of three algorithms $\mathsf{KeyGen}_{\mathcal{E}}$, $\mathsf{Encrypt}_{\mathcal{E}}$, $\mathsf{Decrypt}_{\mathcal{E}}$, a homomorphic encryption scheme additionally includes homomorphic operations on ciphertexts that we denote as $\mathsf{Add}_{\mathcal{E}}$ and $\mathsf{Mult}_{\mathcal{E}}$.

$\mathsf{KeyGen}_{\mathcal{E}}(1^{\lambda})$. Given a security parameter λ, output a public and private key pair $(\mathbf{pk}, \mathbf{sk})$, where $\mathbf{pk}$ is a public key, and $\mathbf{sk}$ is a private key.

$\mathsf{Encrypt}_{\mathcal{E}}(\mathbf{pk}, m \in \{0,1\})$. Given a public key $\mathbf{pk}$ and a plaintext message $m \in \{0,1\}$, output ciphertext $c \in \mathcal{C}$, where $\mathcal{C}$ is ciphertext space.

$\mathsf{Decrypt}_{\mathcal{E}}(\mathbf{sk}, c \in \mathcal{C})$. Given a private key $\mathbf{sk}$ and a ciphertext $c = \mathsf{Encrypt}_{\mathcal{E}}(\mathbf{pk}, m)$, output original plaintext message $m \in \{0,1\}$.

$\mathsf{Add}_{\mathcal{E}}(\mathbf{pk}, c_1 \in \mathcal{C},\ c_2 \in \mathcal{C})$ (resp. $\mathsf{Mult}_{\mathcal{E}}$). Given a public key $\mathbf{pk}$ and two ciphertexts $c_1 = \mathsf{Encrypt}_{\mathcal{E}}(\mathbf{pk}, m_1 \in \{0,1\})$ and $c_2 = \mathsf{Encrypt}_{\mathcal{E}}(\mathbf{pk}, m_2 \in \{0,1\})$, output ciphertext $c' \in \mathcal{C}$.

To be homomorphic, a scheme $\mathcal{E} = (\mathsf{KeyGen}_{\mathcal{E}}, \mathsf{Encrypt}_{\mathcal{E}}, \mathsf{Decrypt}_{\mathcal{E}}, \mathsf{Add}_{\mathcal{E}}, \mathsf{Mult}_{\mathcal{E}})$ has to be able to correctly evaluate some class of binary circuits.

Definition (Correct Homomorphic Decryption). A scheme $\mathcal{E}$ with security parameter λ is *correct* for a L-input binary circuit C composed of $\oplus$ (Add) and $\wedge$ (Mult) gates, if for any key pair $(\mathbf{pk}, \mathbf{sk}) = \mathsf{KeyGen}_{\mathcal{E}}(1^{\lambda})$, and any L plaintext bits $m_1, ..., m_L \in \{0, 1\}$ and corresponding ciphertexts $c_1, ..., c_L$, where $c_i = \mathsf{Encrypt}_{\mathcal{E}}(\mathbf{pk}, m_i)$, the following holds:

$$C(m_1, ..., m_L) = \mathsf{Decrypt}_{\mathcal{E}}(\mathbf{sk}, \mathsf{Evaluate}_{\mathcal{E}}(\mathbf{pk}, C, c_1, ..., c_L)),$$

where $\mathsf{Evaluate}_{\mathcal{E}}$ simply applies $\mathsf{Add}_{\mathcal{E}}$ and $\mathsf{Mult}_{\mathcal{E}}$ to the ciphertexts at corresponding gates of the circuit C.

Definition (Homomorphic Scheme). We call scheme $\mathcal{E}$ *(somewhat) homomorphic* if it is correct for any circuit $C \in \mathcal{C}_{\mathcal{E}}$ for some class of circuits $\mathcal{C}_{\mathcal{E}}$. We call scheme $\mathcal{E}$ *fully homomorphic*, if it is correct for any circuit C.

The Somewhat Homomorphic DGHV Scheme

Construction In this section we recall the DGHV somewhat homomorphic encryption scheme over the integers proposed in [DGHV10], specifically the variant with an error-free public key element. Four parameters are used to control the number of elements in the public key and the bit-length of various integers. We denote by τ the number of elements in the public key, γ their bit-length, η the bit-length of the private key p, and ρ (resp. ρ') the bit-length of the noise in the public key (resp. fresh ciphertext). All of the parameters are polynomial in security parameter λ.

For integers z, p we denote the reduction of z modulo p by $(z \mod p)$ or $[z]_p$.

$\mathsf{DGHV.KeyGen}(1^{\lambda})$. Randomly choose odd η-bit integer p from $(2\mathbb{Z}+1) \cap (2^{\eta-1}, 2^{\eta})$ as private key. Randomly choose integers $q_0, q_1, ..., q_\tau$ each from $[1, 2^{\gamma}/p)$ subject to the condition that the largest q_i is odd, and relabel $q_0, q_1, ..., q_\tau$ so that q_0 is the largest. Randomly choose $r_1, ..., r_\tau$ each from $(-2^{\rho}, 2^{\rho})$. Set error-free public key element $x_0 = q_0 \cdot p$ and $x_i = q_i p + r_i$ for all $1 \leq i \leq \tau$. Output $(\mathbf{pk}, \mathbf{sk})$, where $\mathbf{pk} = (x_0, x_1, ..., x_\tau)$, and $\mathbf{sk} = p$.

$\mathsf{DGHV.Encrypt}(\mathbf{pk}, m \in \{0, 1\})$. Choose a random subset $S \subset \{1, 2, ..., x_\tau\}$ and a random noise integer r from $(-2^{\rho'}, 2^{\rho'})$. Output the ciphertext:

$$c = \left[m + 2r + 2\sum_{i \in S} x_i \right]_{x_0}$$

$\mathsf{DGHV.Decrypt}(\mathbf{sk}, c \in \mathcal{C})$. Output $[c \mod p]_2$.

$\mathsf{DGHV.Add}(\mathbf{pk}, c_1 \in \mathcal{C},\ c_2 \in \mathcal{C})$. Output $[c_1 + c_2]_{x_0}$

$\mathsf{DGHV.Mult}(\mathbf{pk}, c_1 \in \mathcal{C},\ c_2 \in \mathcal{C})$. Output $[c_1 \cdot c_2]_{x_0}$

It was shown in [DGHV10] that scheme is somewhat homomorphic, i.e. a limited number of homomorphic operations can be performed on ciphertext. Specifically, roughly η/ρ' homomorphic multiplications can be performed in a way that the ciphertext is still correctly decryptable (the ciphertext noise must not exceed p). The scheme can be extended to evaluate arbitrary circuits using the bootstrapping technique [DGHV10].

Semantic security The security of the DGHV scheme described as above is based on the error-free approximate-GCD problem. We use the following distribution over γ-bit integers:

$$\mathcal{D}_\rho(p, q_0) = \{\mathsf{Choose}\ q \leftarrow [0, q_0),\ \mathsf{Choose}\ r \leftarrow (-2^\rho, 2^\rho)\ :\ \mathsf{Output}\ x = q \cdot p + r\}$$

Definition ((ρ, η, γ)-Error-Free Approximate-GCD). For a random η-bit odd integer p, given $x_0 = q_0 \cdot p$, where q_0 is a random odd integer in $(2\mathbb{Z} + 1) \cap [0, 2^\gamma/p)$, and polynomially many samples from $\mathcal{D}_\rho(p, q_0)$, output p.

The scheme is semantically secure if the error-free approximate-GCD problem is hard [DGHV10]. Certain constraints have to be put on scheme parameters in order to achieve λ-bit security.

Improvements The scheme has been improved a number of times since it appeared. Public key compression techniques were introduced in [CMNT11, CNT12], achieving public key size of 1GB and 10.1 MB respectively at the 72-bit security level. A modified scheme featuring batching capabilities allowing for SIMD-style operations was proposed in [CLT13]. The most recent improvement at the moment of writing is the SIDGHV scale-invariant modification [CLT14] based on the techniques from [Bra12].

The Symmetric Variant of DGHV Somewhat Scheme

Construction

We now describe the symmetric variant of DGHV scheme with an error-free public element as given in [YKPB13]. We denote by γ the bit-length of the public key x_0, η the bit-length of the private key p, and ρ the bit-length of the noise in the public key and fresh ciphertexts. All of the parameters are polynomial in security parameter λ.

$\mathsf{SDGHV.KeyGen}(1^\lambda)$. Randomly choose odd η-bit integer p from $(2\mathbb{Z}+1) \cap (2^{\eta-1}, 2^\eta)$ as private key. Randomly choose odd q_0 from $(2\mathbb{Z}+1)\cap[1, 2^\gamma/p)$, and set $x_0 = q_0 \cdot p$. Output $(\mathbf{pk}, \mathbf{sk})$ where $\mathbf{pk} = x_0$, $\mathbf{sk} = p$.

$\mathsf{SDGHV.Encrypt}(\mathbf{pk}, m \in \{0,1\})$. Choose random q from $\mathbb{Z} \cap [1, 2^\gamma/p]$, and a random noise integer r from $\mathbb{Z} \cap (-2^\rho, 2^\rho)$. Output the ciphertext:

$$c = [q \cdot p + 2r + m]_{x_0}$$

$\mathsf{SDGHV.Decrypt}(\mathbf{sk}, c \in \mathcal{C})$. Output $[c \mod p]_2$.

$\mathsf{SDGHV.Add}(\mathbf{pk}, c_1 \in \mathcal{C}, c_2 \in \mathcal{C})$. Output $[c_1 + c_2]_{x_0}$.

$\mathsf{SDGHV.Mult}(\mathbf{pk}, c_1 \in \mathcal{C}, c_2 \in \mathcal{C})$. Output $[c_1 \cdot c_2]_{x_0}$.

The main difference compared to the original scheme is that only the noise-free public element x_0 is used, while all the other public key elements $x_1, x_2, ..., x_\tau$ are not used, i.e. τ is set to 0.

The scheme is clearly somewhat homomorphic, allowing to compute a limited amount of additions and multiplications. Note the $\mathsf{SDGHV.Add}$ and $\mathsf{SDGHV.Mult}$ here can be used as mixed operations that take a ciphertext and a plaintext as input, since both ciphertext space and plaintext space are subsets of $\mathbb{Z}$. Further explanations will follow.

Semantic security The scheme remains semantically secure if the error-free approximate-GCD problem is hard.

Attacks Given error-free public element $x_0 = q_0 p$, factoring algorithms can be used to find p. [KLYC13] gives overview of such attacks: elliptic curve factoring method which runs in $\exp(\mathcal{O}(\eta^{1/2}))$, and the general number field sieve which runs in $\exp(\mathcal{O}(\gamma^{1/3}))$. The following constraints must be used to resist the attacks: $\eta \geq \mathcal{O}(\lambda^2)$, $\gamma \geq \mathcal{O}(\lambda^3)$.

Given $x_0 = q_0 \cdot p$ and set of $x_i = q_i p + r$, $1 \leq i \leq n$, Lagrasias algorithm can be used to find p in $\mathcal{O}(2^{n/(\eta-\rho)})$ time. This can be resisted by choosing $\gamma/\eta^2 = \omega(\log \lambda)$. For the case $i = 1$, the following attacks can also be used: Chen-Nguyen [CN12] attack running in $\tilde{\mathcal{O}}(2^{\rho/2})$, resisted by requiring $\rho > \mathcal{O}(\lambda)$, and Howgrave-Grahan [HG01] attack, resist by requiring $\gamma > \eta^2/\rho$ [KLYC13].

Note that the parameters chosen for benchmark in [YKPB13] are not secure at the declared level against current approximate-GCD attacks (as also briefly noted in [DC14]). Secure parameter constraints and a proposed parameter set are given below.

Parameter selection We propose to use the following parameter constraints:

- $\rho = \tilde{\mathcal{O}}(\lambda)$ to secure against the Chen-Nguyen attack [CN12].
- $\eta = \tilde{\Omega}(\lambda^2 + \rho \cdot L)$ to resist factoring attacks and allow evaluation of L successive multiplications.
- $\gamma = \eta^2 \omega(log \lambda)$ to resist factoring attacks and the Howgrave-Grahan [HG01] attack

A convenient parameter set could be defined as follows: $\rho = 2\lambda, \eta = \mathcal{O}(\lambda^2 \cdot L), \gamma = \mathcal{O}(\lambda^5 \cdot L^2)$.

Motivation

The variant was proposed in [YKPB13] for the purpose of constructing a practical single-server computational private information retrieval protocol. The authors noticed that the generic PIR protocol they outlined didn't require encrypting new integers on the server side, implying the public key elements $x_1, x_2, ..., x_\tau$ only used in encryption procedure could be omitted. Obtained symmetric scheme has improved the protocol's communication overhead due to the absence of all of the public key elements except the error-free element, while enabling to efficiently evaluate the server-side PIR retrieval algorithm.

The key feature of the SDGHV scheme that allows to avoid encryptions on the server-side is the ability to perform "natural" mixed homomorphic operations on plaintext and ciphertext:

$\mathsf{SDGHV.Add}(\mathbf{pk}, c \in \mathcal{C}, m \in \{0,1\})$. Output $[c+m]_{x_0}$.

$\mathsf{SDGHV.Mult}(\mathbf{pk}, c \in \mathcal{C}, m \in \{0,1\})$. Output $[c \cdot m]_{x_0}$.

Indeed, we can see that the mixed operations are correct (for a certain number of operations). Given the private key p, public key x_0, some messages $m, m' \in \{0,1\}$, and a ciphertext $c = \mathsf{SDGHV.Encrypt}(x_0, m) = [q \cdot p + 2r + m]_{x_0}$:

$$\begin{aligned}
\mathsf{SDGHV.Decrypt}(p, \mathsf{SDGHV.Add}(x_0, c, m')) &= \mathsf{SDGHV.Decrypt}(p, [c+m']_{x_0}) \\
&= \mathsf{SDGHV.Decrypt}(p, [q \cdot p + 2r + m + m']_{x_0}) \\
&= [2r + m + m']_2 \\
&= m \oplus m',
\end{aligned}$$

if $m + m' + 2r < p$. Analogically,

$$\begin{aligned}
\mathsf{SDGHV.Decrypt}(p, \mathsf{SDGHV.Mult}(x_0, c, m')) &= \mathsf{SDGHV.Decrypt}(p, [c \cdot m']_{x_0}) \\
&= \mathsf{SDGHV.Decrypt}(p, [m' \cdot (q \cdot p + 2r + m)]_{x_0}) \\
&= [2rm' + m \cdot m']_2 \\
&= m \wedge m'
\end{aligned}$$

We can see the number of homomorphic additions in the form $c + m'$, where $m' \in \{0,1\}$, is limited to $p - 2r - m$ for a specific ciphertext $c = [q \cdot p + 2r + m]_{x_0}$.

Notes on applying existing DGHV improvements Batching techniques as described in [CLT13, KLYC13, CCK+13] could be applied to SDGHV scheme, but the mixed homomorphic operations correctness would be lost if CRT batching were used. Ciphertext compression techniques [CNT12], was applied in our implementation in order to decrease the communication cost in SFE setting.

Implementation

The scheme was implemented as a C++ library using GNU Multiprecision (*GMP*) for big integer arithmetics computations and *Boost* for testing and serialization. The implementation is publicly available on the Github [Kul15].

References

BGV12. Zvika Brakerski, Craig Gentry, and Vinod Vaikuntanathan. (Leveled) fully homomorphic encryption without bootstrapping. In *Proceedings of the 3rd Innovations in Theoretical Computer Science Conference*, pages 309–325. ACM, 2012.

BLLN13. Joppe W. Bos, Kristin Lauter, Jake Loftus, and Michael Naehrig. Improved Security for a Ring-Based Fully Homomorphic Encryption Scheme. In Martijn Stam, editor, *IMA Int. Conf.*, volume 8308 of *Lecture Notes in Computer Science*, pages 45–64. Springer, 2013.

Bra12. Zvika Brakerski. Fully Homomorphic Encryption without Modulus Switching from Classical GapSVP. In Reihaneh Safavi-Naini and Ran Canetti, editors, *CRYPTO*, volume 7417 of *Lecture Notes in Computer Science*, pages 868–886. Springer, 2012.

CCK+13. JungHee Cheon, Jean-Sébastien Coron, Jinsu Kim, MoonSung Lee, Tancrède Lepoint, Mehdi Tibouchi, and Aaram Yun. Batch Fully Homomorphic Encryption over the Integers. In Thomas Johansson and PhongQ. Nguyen, editors, *Advances in Cryptology – EUROCRYPT 2013*, volume 7881 of *Lecture Notes in Computer Science*, pages 315–335. Springer Berlin Heidelberg, 2013.

CLT13. Jean-Sébastien Coron, Tancrède Lepoint, and Mehdi Tibouchi. Batch Fully Homomorphic Encryption over the Integers. Cryptology ePrint Archive, Report 2013/036, 2013. http://eprint.iacr.org/.

CLT14. Jean-Sébastien Coron, Tancrède Lepoint, and Mehdi Tibouchi. Scale-invariant fully homomorphic encryption over the integers.

In *Public-Key Cryptography–PKC 2014*, pages 311–328. Springer, 2014.

CMNT11. Jean-Sebastien Coron, Avradip Mandal, David Naccache, and Mehdi Tibouchi. Fully Homomorphic Encryption over the Integers with Shorter Public Keys. Cryptology ePrint Archive, Report 2011/441, 2011. http://eprint.iacr.org/.

CN12. Yuanmi Chen and Phong Q. Nguyen. Faster Algorithms for Approximate Common Divisors: Breaking Fully-Homomorphic-Encryption Challenges over the Integers. In David Pointcheval and Thomas Johansson, editors, *EUROCRYPT*, volume 7237 of *Lecture Notes in Computer Science*, pages 502–519. Springer, 2012.

CNT12. Jean-Sébastien Coron, David Naccache, and Mehdi Tibouchi. Public Key Compression and Modulus Switching for Fully Homomorphic Encryption over the Integers. In David Pointcheval and Thomas Johansson, editors, *EUROCRYPT*, volume 7237 of *Lecture Notes in Computer Science*, pages 446–464. Springer, 2012.

DC14. Changyu Dong and Liqun Chen. A Fast Single Server Private Information Retrieval Protocol with Low Communication Cost. In *Computer Security-ESORICS 2014*, pages 380–399. Springer, 2014.

DGHV10. Marten van Dijk, Craig Gentry, Shai Halevi, and Vinod Vaikuntanathan. Fully Homomorphic Encryption over the Integers. In Henri Gilbert, editor, *EUROCRYPT*, volume 6110 of *Lecture Notes in Computer Science*, pages 24–43. Springer, 2010.

FV12. Junfeng Fan and Frederik Vercauteren. Somewhat Practical Fully Homomorphic Encryption. Cryptology ePrint Archive, Report 2012/144, 2012. http://eprint.iacr.org/.

Gen09. Craig Gentry. *A fully homomorphic encryption scheme*. PhD thesis, Stanford University, 2009.

GHS12a. Craig Gentry, Shai Halevi, and Nigel P. Smart. Fully Homomorphic Encryption with Polylog Overhead. In David Pointcheval and Thomas Johansson, editors, *EUROCRYPT*, volume 7237 of *Lecture Notes in Computer Science*, pages 465–482. Springer, 2012.

GHS12b. Craig Gentry, Shai Halevi, and Nigel P. Smart. Homomorphic Evaluation of the AES Circuit. In Reihaneh Safavi-Naini and Ran Canetti, editors, *CRYPTO*, volume 7417 of *Lecture Notes in Computer Science*, pages 850–867. Springer, 2012.

GSW13. Craig Gentry, Amit Sahai, and Brent Waters. Homomorphic encryption from learning with errors: Conceptually-simpler, asymptotically-faster, attribute-based. In Ran Canetti and JuanA. Garay, editors, *Advances in Cryptology – CRYPTO 2013*, volume 8042 of *Lecture Notes in Computer Science*, pages 75–92. Springer Berlin Heidelberg, 2013.

HG01. Nick Howgrave-Graham. Approximate integer common divisors. In *Cryptography and Lattices*, pages 51–66. Springer, 2001.

KLYC13. Jinsu Kim, Moon Sung Lee, Aaram Yun, and Jung Hee Cheon. CRT-based Fully Homomorphic Encryption over the Integers. Cryptology ePrint Archive, Report 2013/057, 2013. http://eprint.iacr.org/.

Kul15. Bogdan Kulynych. Implementation of a Symmetric Variant of DGHV Somewhat Homomorphic Encryption Scheme. https://github.com/bogdan-kulynych/libshe, 2015.

LN14. Tancrède Lepoint and Michael Naehrig. A Comparison of the Homomorphic Encryption Schemes FV and YASHE. Cryptology ePrint Archive, Report 2014/062, 2014. http://eprint.iacr.org/.

LTV12. Adriana López-Alt, Eran Tromer, and Vinod Vaikuntanathan. On-the-fly multiparty computation on the cloud via multikey fully homomorphic encryption. In *Proceedings of the forty-fourth annual ACM symposium on Theory of computing*, pages 1219–1234. ACM, 2012.

SV11. N.P. Smart and F. Vercauteren. Fully Homomorphic SIMD Operations. Cryptology ePrint Archive, Report 2011/133, 2011. http://eprint.iacr.org/.

YKPB13. Xun Yi, Md. Golam Kaosar, Russell Paulet, and Elisa Bertino. Single-Database Private Information Retrieval from Fully Homomorphic Encryption. *IEEE Trans. Knowl. Data Eng.*, 25(5):1125–1134, 2013.