Achieving CCA1-security in homomorphic encryption

Eduardo Morais[1], **Diego F. Aranha**[1], **Ricardo Dahab**[1]

[1]Institute of Computing – University of Campinas (Unicamp)

{dfaranha, rdahab}@ic.unicamp.br, emorais@lasca.ic.unicamp.br

Abstract. *This paper proposes the combination of homomorphic encryption and verifiable computation to avoid key recovery attacks and achieve CCA1-secure constructions of Somewhat Homomorphic Encryption (SHE) schemes described in the literature. We also provide concrete parameters, based on the best-attack analysis, concluding that the AGCD [van Dijk et al. 2010] family of SHE schemes may be the best implementation choice under certain circumstances.*

Resumo. *Este artigo propõe a combinação de encriptação homomórfica e computação verificável para evitar ataques de recuperação de chaves e obter segurança CCA1 em construções de esquemas parcialmente homomórficos descritos na literatura. Além disso, são propostos parâmetros concretos, baseados na análise do melhor ataque, concluindo que a família AGCD [van Dijk et al. 2010] de esquemas SHE pode ser consideradas a melhor escolha em determinadas circunstâncias.*

1. Introduction

Homomorphic encryption has been a topic of great interest for the Cryptology community over the last few years, since Craig Gentry's breakthrough in 2009 [Gentry 2009]. Although Fully Homomorphic Encryption (FHE) is yet to become practical, SHE schemes can be used to construct practical applications. An important drawback, however, is the fact that all but one of the SHE schemes described in the literature are susceptible to *key recovery attacks*, a concrete threat in many scenarios. In this work, we investigate how verifiable computation can be combined with homomorphic encryption in order to avoid key recovery attacks. Indeed, we show that it is possible to achieve CCA1-security for homomorphically evaluating quadratic multivariate polynomials.

1.1. Homomorphic encryption

Informally, homomorphic encryption provides the possibility of having a pair of encryption and decryption functions, ENC, DEC, that allows the computation of a function f on an encrypted text c, such that $\text{DEC}(f(c)) = f(m)$, where $c = \text{ENC}(m)$. That is, allowing functions to be computed on encrypted texts without the need for decrypting them first. The following two definitions formalize this notion.

Definition 1.1. *Correctness. A scheme* $\mathcal{E}(\text{KEYGEN}, \text{DEC}, \text{ENC}, \text{EVAL})$ *is* correct *if, for a determined circuit* $\mathbf{C}$ *and every key pair* (sk, pk)*, where* sk *is the private key and* pk *is the public key generated by* KEYGEN*, any message tuple* $\overline{m} = \langle m_1, \ldots, m_t \rangle$ *and corresponding ciphertexts* $\overline{c} = \langle c_1, \ldots, c_t \rangle$*, that is,* $c_i = \text{ENC}_{pk}(m_i)$ *for* $1 \leq i \leq t$*, then we have that*

$$\text{DEC}_{sk}(\text{EVAL}_{pk}(\mathbf{C}, \overline{c})) = \mathbf{C}(\overline{m}).$$

Furthermore, algorithms KEYGEN*,* DEC*,* ENC *and* EVAL *must have polynomial complexity.*

Definition 1.2. *Fully Homomorphic Encryption. A scheme* $\mathcal{E}$ *is* correct for a class $\mathbf{S}_\mathbf{C}$ *of circuits, if it is correct for each* $\mathbf{C} \in \mathbf{S}_\mathbf{C}$*. Moreover,* $\mathcal{E}$ *is denominated* fully homomorphic *if it is correct for every algebraic circuit, or, equivalently, if it is correct for every Boolean circuit.*

SHE schemes corresponds to homomorphic encryption schemes that are correct for circuits whose multiplicative depth is limited by a certain upper bound, denoted by ℓ.

1.2. Security model

We say that a cryptosystem is secure against *chosen ciphertext attacks* (CCA2) if there is no polynomial time adversary that can win the following game with non-negligible probability.

Setup. The challenger obtains $(\mathtt{sk}, \mathtt{pk}) = \text{KEYGEN}(\lambda)$, where λ is a security parameter, and sends $\mathtt{pk}$ to the adversary $\mathcal{A}$.

Queries. $\mathcal{A}$ sends ciphertexts to the challenger, before or after the challenge, who returns the corresponding plaintexts.

Challenge. The adversary randomly generates two plaintexts $m_0, m_1 \in \mathcal{M}$ and sends to the challenger, who then randomly chooses a bit $b \in \{0, 1\}$ and computes the ciphertext $c = \text{ENC}_{\mathtt{pk}}(m_b)$. The challenger sends c to $\mathcal{A}$.

Answer. $\mathcal{A}$ sends a bit b' to the challenger and wins the game if $b' = b$.

If we allow queries only before the challenge, we say that the cryptosystem is secure against CCA1 adversaries (lunchtime attacks). Queries can be interpreted as accesses to a *decryption oracle*. If, instead, we only allow access to an *encryption oracle*, namely the adversary can choose any message to be encrypted under the same key pair, then we say that the cryptosystem is secure against *chosen plaintext attacks* (CPA). Clearly, resistance to CCA2 attacks are the most desirable.

In homomorphic encryption, it is impossible to achieve CCA2 security, because the adversary can simply add to the encrypted message some encryption of zero, which can be obtained by querying the encryption oracle, and send it back to the decryption oracle. Many FHE schemes have as public value an encryption of the private key bits, which can be sent to the decryption oracle before the challenge, making such schemes insecure against CCA1 adversaries. Indeed, a *key recovery* attack is stronger than a CCA1 attack;

also, Loftus et al [Loftus et al. 2011] showed that Gentry's construction over ideal lattices is vulnerable to key recovery attacks and presented the only somewhat homomorphic encryption scheme that is known to be CCA1-secure.

In 2015, Dahab, Galbraith and Morais showed that the NTRU-based family of SHE schemes is vulnerable to key recovery attacks [Dahab et al. 2015]. Hence, except for Loftus et al's [Loftus et al. 2011] scheme, no other known SHE proposal achieves CCA1 security.

1.3. Homomorphic verifiable computation

Although homomorphic encryption is a very flexible cryptographic primitive, when applied to the cloud computing scenario, it lacks an important property: the ability to verify if a given homomorphic computation corresponds to what the client desired. A verifiable computation scheme could solve this problem, provided two requirements are met. First, the cloud must not spend much more time to perform the verifiable computation when compared to the non-verifiable solution. Second, the client must be able to verify the result faster than the time it takes to perform the entire computation by himself. There are proposals [Gennaro et al. 2010, Chung et al. 2010] that use homomorphic encryption to construct a verifiable scheme, because it is possible to offer input and output privacy, since both are encrypted. However, the underlying security model does not allow verification queries. Recently, Fiore, Gennaro and Pastro [Fiore et al. 2014] proposed a new construction that does allow verification queries, improving on the security model. They showed how to solve practical problems, such as computing quadratic multivariate polynomials over encrypted data, which can be used to homomorphically compute statistical functions. We remark that this application requires only one level of multiplications, which is an important characteristic to be considered in order to calculate the parameters of the underlying SHE scheme.

Definition 1.3. *A verifiable computation scheme $\mathcal{VC}$ is defined by the algorithms* $(\text{KEYGEN}, \text{PROBGEN}, \text{COMPUTE}, \text{VERIFY})$*, as follows:*

Key generation. *Algorithm* $\text{KEYGEN}(1^\lambda, f)$ *generates secret key* $\mathtt{sk}$ *and evaluation key* $\mathtt{esk}$*.*

Problem generation. *Using secret key* $\mathtt{sk}$*, algorithm* PROBGEN *receives as input ciphertext* c_i *and computes the corresponding authentication tag* σ_i*, such that* $\sigma_i = \text{AUTH}_{vk}(c_i, (\cdot, i))$*.*

Verification. *Given the secret* $\mathtt{sk}$*, tag* σ *and ciphertext* c*, we have that* $\text{VERIFY}_{sk}(\sigma, c)$ *returns* 1 *if* $c = f([c_i])$ *and* $\sigma = \text{AUTH}_{vk}(c, (\cdot, i))$*. Otherwise, it returns* 0*.*

Evaluation. *Given* $\sigma_1, \ldots, \sigma_t$ *and the description of function* f*, algorithm* $\text{COMPUTE}_{esk}([\sigma_i], \Delta, f)$ *returns the authentication tag* σ *that corresponds to the ciphertext* $c = \text{EVAL}_{edk}([c_i], f)$ *obtained by running the* EVAL *algorithm from the underlying homomorphic encryption scheme. We say that the* $\mathcal{VC}$ *scheme is* correct *if* $\text{VERIFY}_{sk}(\sigma, c)$ *outputs* 1*.*

2. Main contributions

Here we list the main contributions of this work:

- the combination of verifiable computation and homomorphic encryption in order to avoid key recovery attacks and achieve CCA1-security. This is relevant because almost every SHE scheme in the literature is vulnerable to CCA1 attacks, which are a feasible scenario in cloud computing;
- the comparison of AGCD-based [van Dijk et al. 2010] and BGV-based [Brakerski et al. 2011] SHE schemes for a specific scenario, where we can homomorphically evaluate quadratic multivariate polynomials. In general, BGV is considered the best choice for the implementation of somewhat homomorphic encryption; in this particular case, however, the AGCD scheme, which is simpler, also offers better performance in practice;
- the proposal of concrete parameters for a specific multiplicative depth to be secure against the attacks described in the literature, based on the utilization of some variation of the LLL algorithm.

References

Brakerski, Z., Gentry, C., and Vaikuntanathan, V. (2011). Fully homomorphic encryption without bootstrapping. *Electronic Colloquium on Computational Complexity (ECCC)*, 18:111.

Chung, K., Kalai, Y., and Vadhan, S. (2010). Improved delegation of computation using fully homomorphic encryption. In Rabin, T., editor, *Advances in Cryptology - CRYPTO 2010*, volume 6223 of *Lecture Notes in Computer Science*, pages 483–501. Springer Berlin Heidelberg.

Dahab, R., Galbraith, S., and Morais, E. (2015). Adaptive key recovery attacks on NTRU-based somewhat homomorphic encryption schemes. In Lehmann, A. and Wolf, S., editors, *Information Theoretic Security*, volume 9063 of *Lecture Notes in Computer Science*, pages 283–296. Springer International Publishing.

Fiore, D., Gennaro, R., and Pastro, V. (2014). Efficiently verifiable computation on encrypted data. Cryptology ePrint Archive, Report 2014/202. `http://eprint.iacr.org/`.

Gennaro, R., Gentry, C., and Parno, B. (2010). Non-interactive verifiable computing: Outsourcing computation to untrusted workers. In Rabin, T., editor, *Advances in Cryptology - CRYPTO 2010*, volume 6223 of *Lecture Notes in Computer Science*, pages 465–482. Springer Berlin Heidelberg.

Gentry, C. (2009). Fully homomorphic encryption using ideal lattices. In *STOC '09: Proceedings of the 41st annual ACM symposium on Theory of computing*, pages 169–178, New York, NY, USA. ACM.

Loftus, J., May, A., Smart, N. P., and Vercauteren, F. (2011). On CCA-secure somewhat homomorphic encryption. In *In Selected Areas in Cryptography*, pages 55–72.

van Dijk, M., Gentry, C., Halevi, S., and Vaikuntanathan, V. (2010). Fully homomorphic encryption over the integers. In *Proceedings of the 29th Annual International Conference on Theory and Applications of Cryptographic Techniques*, EUROCRYPT'10, pages 24–43, Berlin, Heidelberg. Springer-Verlag.