Verifiable computation on encrypted data

TRABAJO DE FIN DE MÁSTER
Curso 2022-2023

UNIVERSIDAD
COMPLUTENSE
MADRID

En colaboración con el Instituto IMDEA Software

MIGUEL MORONA MÍNGUEZ

Facultad de Ciencias Matemáticas
Máster en Ingeniería Matemática

July 6, 2023

Directores:

Dario Fiore
Ignacio Cascudo
Daniele Cozzo
Maria Emilia Alonso

Resumen

El auge de nuevas herramientas como el Cloud Computing puede convertirse en un arma de doble filo si se tiene en cuenta los numerosos peligros que pueden surgir. La Computación Verificable (VC) es una técnica criptográfica en el *estado del arte* que pretende solventar los riesgos de la computación remota y garantizar propiedades como la corrección o la privacidad.

En esta tesis de máster se propone un esquema de Computación Verificable que combina técnicas de cifrado homomórfico y autenticación homomórfica. Tras un análisis de seguridad, se propone una variación de estas técnicas que permite una mayor eficiencia, en términos de coste de comunicación entre las partes. Este trabajo incluye también una precisa estimación de este coste, junto con otra estimación de tiempo computacional en ejecutar los algoritmos necesarios.

Palabras clave: Encriptado homomórfico, Computación Verificable, MAC, Seguridad semántica, Infalsificable.

Abstract

The rise of new tools such as Cloud Computing can become a double-edged sword if we take into account the many dangers that can arise. Verifiable Computation (VC) is a state-of-the-art cryptographic technique that aims to overcome the risks of remote computing and to guarantee properties such as correctness or privacy.

In this master thesis, a Verifiable Computation scheme combining homomorphic encryption and homomorphic authentication techniques is proposed. After a security analysis, a variation of these techniques is proposed that allows for greater efficiency, in terms of communication cost between the parties. This work also includes a precise estimation of this cost, along with an estimation of the computational time in executing the required algorithms.

Keywords: Homomorphic encryption, Verifiable Computation, MAC, Semantic security, Unforgeability.

Agradecimientos

Realizar esta tesis de máster me ha ayudado a darme cuenta de todas las personas a las que tengo que estar agradecido. Por supuesto este trabajo no habría sido posible sin la inestimable ayuda de todos mis directores, a los que les agradezco la motivación y las ganas de enseñarme este tema y todo lo relacionado con ello. Estar estos 5 meses aprendiendo de vosotros ha sido un placer.

Agradezco, como no, a todo el personal de IMDEA Software y en especial a mis compañeros de oficina Claudia y David por enseñarme el maravilloso mundo de la investigación. ¡Vosotros sois investigación!. También me gustaría añadir que el trabajo de investigador no es posible sin la inmensa ayuda de muchas personas detrás de él. Trabajando en IMDEA te das cuenta del cariño que le puedes coger a personas como Pili e Irene, sin ellas el trabajo no sería el mismo.

A mis padres, Virgilio y Alicia y a mis hermanas por estar detrás de mí, ayudándome y respetándome. No os digo lo suficiente lo mucho que os quiero y que agradezco que me apoyaseis en todos mis estudios. A todos mis amigos de Rivas, gracias por creer en mi y sacarme de la rutina cuando más lo necesitaba. Gracias a los chicos del máster por este año. Hemos creado un grupo increíble que me ha hecho sentir cosas maravillosas, sin vosotros nada hubiese igual.

Agradezco a todas las personas que me rodean y que hacen mi persona. Al final soy un pedazo de cada uno de vosotros.

Contents

Chapter 1

Introduction

Can an outside party compute for us, without learning our private data? This is a question that concerns many of recent publications in the cryptography area. To address this problem, one can think in applying cryptographic techniques such as a Verifiable Computation Scheme. In a Verifiable Computation scheme, one can get the data computed, received along with a cryptographic proof that ensures correctness. But this simple consideration may not contemplate the privacy about the inputs.

In the last decade, communication has been considerably changing, and we have been used to sharing data with others. There exist some new ways to share data, which can be sensible. One example of this is "the cloud".

1.1 Sharing data on the Cloud

In today's interconnected world, cloud computing has revolutionised the way we store, access, and share data. The cloud offers immense convenience, making it a preferred choice for individuals and organisations alike. However, it is essential to acknowledge the potential challenges associated with data sharing in the cloud. One of the most important notions we want when we share data is to preserve privacy. The most common "procedure" to preserve data privacy is encrypting it. On a more mathematical way, privacy means that the public encrypted outputs over different inputs are indistinguishable. Nevertheless, this is not the unique necessary property when data is sent.

- **Correctness**: Correct values given by the sender should be seen correct for the receiver.
- **Authenticity**: Users should be able to ensure that the received data comes from the claimed sender.
- **Integrity**: The received data should not have been modified while sending it. The received data is exactly the one the outside party send to us.

These security notions should be at the forefront of our minds when entrusting data to cloud service providers. Implementing robust encryption protocols can help mitigate the risks of unauthorised access and breaches.

1.2 Motivation

In recent years, there has been a significant increase in the adoption of cloud computing and outsourcing of computational tasks to remote servers. This trend has provided numerous benefits, such as cost-effectiveness and scalability. However, it has also raised concerns regarding the security and privacy of sensitive data. Users often hesitate to outsource their computations due to the fear of exposing their confidential information to untrusted third parties.

Verifiable computation on encrypted data has emerged as a promising solution to address these security concerns. The idea behind this approach is to enable clients to delegate computationally intensive tasks to remote servers while ensuring the correctness and integrity of the results. By leveraging encryption techniques, the client can encrypt their data before outsourcing it, thus preserving the confidentiality of the information.

The need for verifiable computation arises from the inherent lack of trust in the remote servers. Clients must be able to verify that the server performs the computations correctly and that the results obtained are valid. This verification becomes even more critical when dealing with sensitive data, such as personal or financial information, where the consequences of malicious or erroneous computations can be severe.

1.3 Scope

To achieve a privacy property for our data, the main solution seems to be to encrypt it. However, in a Verifiable Computation scheme this is usually not enough. Since we want our data to be computed, we have to claim the encryption system to be fully homomorphic or at least support a limited number of operations. This topic is well known since many years ago. Nevertheless, the first proposal for a fully homomorphic encryption scheme was introduced by Gentry in his doctoral thesis fifteen years ago [Gen09].

Our goal will be to provide a Verifiable Computation ($\mathcal{VC}$) system considering running time cost and communication cost between parties. The properties we will have in mind are the ones described above, regarding of correctness and authenticity. To do so, we will apply an HomMAC procedure to our encrypted data and verifying using the homomorphic property. One important attribute we will like to obtain is the efficiency, which is the main scope for many cryptographic schemes.

1.4 Our contribution

Verifiable Computation it's a well known topic [GGP10] [Gol+13] that enables a computationally weak client to "outsource" the computation of a function F on various inputs $x_1, \ldots, x_k$ to one or more workers. To our knowledge, these were the first relevant works that consider privacy in the context of $\mathcal{VC}$. Both of these solutions are, however not very satisfactory in terms of efficiency. Another approach was the one done in [FGP14] where they proposed to use homomorphic MACs in order to prove that the evaluation of FHE ciphertexts has been done correctly. Their solution is inherently bound to computations of quadratic functions. One more recent work in this area is [FNP20] where the authors proposed a new protocol for verifiable computation on encrypted data that supports homomorphic computations of multiplicative depth larger than 1.

This work is along the same lines as the ones mentioned above. We propose a generic $\mathcal{VC}$ scheme where we suppose an homomorphic encryption to preserve data privacy and an homomorphic MAC to authenticate the ciphertexts. As an homomorphic encryption consider the BV encryption and the CF13 as the homomorphic MAC. This authenticator procedure is very efficient but the only problem it has for our scheme is the input space that restricts us to authenticate messages $m \in \mathbb{Z}_p$. Since the BV encryption outputs ciphertexts that belong to rings, we will modify (not in essence, but structurally) this homomorphic MAC procedure in order to be able to authenticate ring values (we will call it $\widetilde{\mathsf{CF13}}$).

It is well known that for many homomorphic encryption schemes, the size of the ciphertext is directly related to the degree of the homomorphic function f. So, due to efficiency and despite [FNP20], this work

is meant to only contemplate low-degree functions (as one can see in the Appendix).

1.5 Outline

In this thesis we first introduce in chapter 2 some definitions and cryptography backgrounds. There, in 2.1, we properly define what a Verifiable Computation scheme is. Later in chapter 3 we talk about some cryptographic procedures such as the Message Authentication Codes and the relevance of the homomorphisms in encryption schemes. In chapter 4 we present our Verifiable Computation scheme which combines an homomorphic encryption and an homomorphic MAC. In this same chapter we give precise estimation of the computational cost in considering this $\mathcal{VC}$ scheme. Also, we formally prove the security notions of a $\mathcal{VC}$ scheme in our generic protocol. Later, in chapter 5 we modify an existent homomorphic MAC in order to reduce the size of the tag and ciphertext and still being accepted by the HomMAC.Ver algorithm. Afterwards, we compare the results with the ones obtained before.

We reserve the appendix section to give precisely examples of some functions and scenarios where this scheme can be useful.

Chapter 2

Background and definitions

The idea of this chapter is to introduce some concepts and definitions to the reader that may help the understanding of the thesis. First of all, we introduce some useful notation.

Notation: If S is a set, $x \leftarrow\!\$\ S$ denotes the process of selecting x uniformly at random in S. We will interpret $\mathcal{R}$ as a finite commutative ring $\mathcal{R} := \mathbb{Z}[X]/(f)$. We refer to λ as a security parameter and the expression 1^λ in $\mathsf{KeyGen}(1^\lambda)$ as the bit length security of the keys generated by the KeyGen algorithm.

Definition 2.1 (Negligible)**.** *We say that a function ϵ: $\mathbb{N} \to \mathbb{R}^+$ is negligible if for every positive polynomial $p(\lambda)$ there exists $\lambda_0 \in \mathbb{N}$ such that for all $\lambda > \lambda_0$: $\epsilon(\lambda) < \frac{1}{p(\lambda)}$*

Definition 2.2 (Pseudorandom Function (PRF))**.** *A PRF is a function $F : \mathcal{K} \times \mathcal{X} \to \mathcal{Y}$ if for all given key $k \in \mathcal{K}$, the function $F(k, \cdot)$ can be efficiently evaluated at all points $x \in \mathcal{X}$ [GGM86].*

We say F is a *pseudorandom function* (PRF) if for any adversary $\mathcal{A}$ running in polynomial time, it is not possible for it to distinguish between an output given by the PRF and a real random generated value.

Definition 2.3 (Arithmetic circuits)**.** *Given a set of variables $X_1, \ldots, X_n$, an arithmetic circuit $\mathcal{C}$ over an algebraic structure $\mathbb{A}$ is a directed graph without cycles with the following properties. Each node is called a gate; gates with in-degree 0 are either input gates labelled with some variable X_i, or constant gates labelled with a field value $c \in \mathbb{A}$; gates with out-degree 0 are called output gates; gates with in-degree and out-degree greater than 0 are called internal gates. These internal gates may indicate a relationship between certain variables [Fio22].*

Definition 2.4 (Exceptional set)**.** *Let $\mathcal{R}$ be a finite commutative ring with identity. Let $\mathbb{A} = \{a_1, \ldots, a_n\} \subset \mathcal{R}$. We say that $\mathbb{A}$ is an exceptional set if $\forall i \neq j$, $a_i - a_j \in \mathcal{R}^*$.*

Theorem 2.1 (Chinese Remainder Theorem)**.** *Let $\mathcal{R}$ be a finite commutative ring with identity. Let I_1, $\ldots$, I_m be m pairwise co-prime ideals of $\mathcal{R}$, i.e. $\forall i \neq j$, $I_i + I_j = \mathcal{R}$. Denote $I = I_1 \cdots I_m$. Then the following map is a ring isomorphism:*

$$\mathcal{R}/I \to \mathcal{R}/I_1 \times \cdots \times \mathcal{R}/I_m$$

$$r \bmod I \mapsto (r \bmod I_1, \ldots, r \bmod I_m)$$

Definition 2.5 (Labelled program)**.** *A labelled program $\mathcal{P}$ consists of a tuple $(f, \tau_1, \ldots, \tau_n)$ where f is a circuit and the binary strings $\tau_1, \ldots, \tau_n \in \{0,1\}^*$ are the labels of the input nodes of f. [GW13]*

2.1 Verifiable Computation

Consider the scenario in which a server is supposed to compute a function f on input m and sends a result y to client. How can the client be ensured that the received value, y, is indeed $f(m)$? Clearly, one solution to this problem is for the client to execute the function f locally, so computing $f(m)$ and comparing this result to y. However, it is evident that such a solution completely vanishes the benefits of remote computation. Furthermore, such a solution could be simply infeasible if the function f is too expensive to be computed by the client or the input m is too large to be stored by the client.

Verifiable computation ($\mathcal{VC}$) is a cryptographic protocol that provides a solution to the problem above [GGP10]. In a nutshell, the idea is that the server can generate a mathematical proof π about the statement that $y = f(m)$.

2.1.1 Formal definition

A formal definition for a Verifiable Computation Scheme $\mathcal{VC} = (\mathsf{KeyGen}, \mathsf{ProbGen}, \mathsf{Compute}, \mathsf{Verify})$ consists of the following algorithms [FGP14]:

- $\mathsf{KeyGen}(f, 1^\lambda) \rightarrow (\mathsf{pk}, \mathsf{sk})$: Based on the security parameter λ, the randomized key generation algorithm generates a public key which is used by the server to compute f. It also computes a matching secret key, kept in private by the client.
- $\mathsf{ProbGen}_{\mathsf{sk}}(x) \rightarrow (\sigma_x, \tau_x)$: The problem generation algorithm uses the secret key sk to encode the input x as a public value σ_x that is given to the server to compute with, and a secret value τ_x which is kept private by the client.
- $\mathsf{Compute}_{\mathsf{pk}}(\sigma_x) \rightarrow (\sigma_y)$: Using the client's public key and the encoded input, the server computes an encoded version of the function's output $y = f(x)$.
- $\mathsf{Verify}_{\mathsf{sk}}(\tau_x, \sigma_y) \rightarrow (acc, y)$: Using the secret key sk and the secret τ_x, the verification algorithm converts the server's output into a bit acc and a string y. If $acc = 1$ we say the client accepts $y = f(x)$, if $acc = 0$ we say the client rejects.

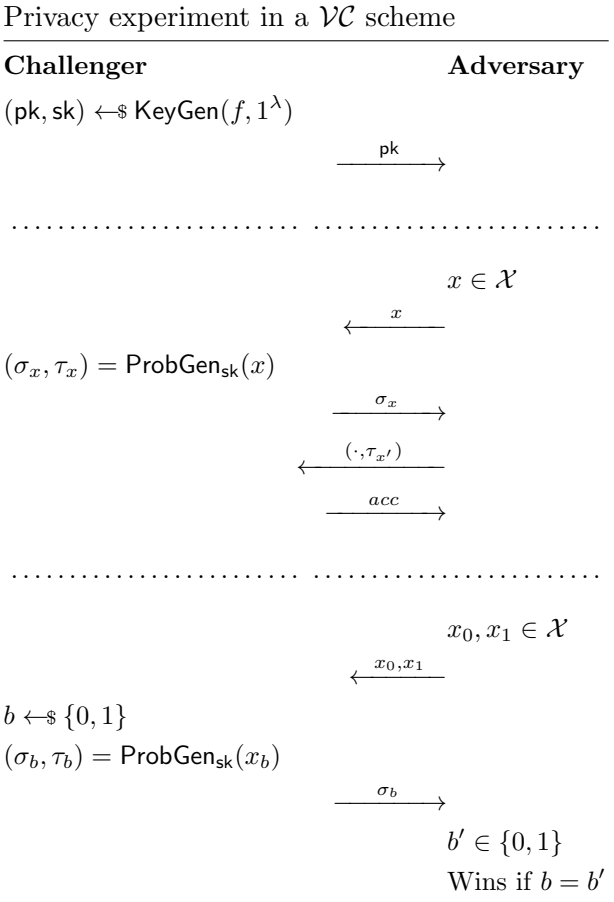
2.1.2 Security properties

Given this algorithms that define the formal definition of a $\mathcal{VC}$, we can describe the following security notions:

Definition 2.6 (Correctness)**.** *A verifiable computation scheme $\mathcal{VC}$ is correct if $\forall f, x$: if $(\mathsf{pk}, \mathsf{sk}) \leftarrow \mathsf{KeyGen}(f, \lambda)$, $(\sigma_x, \tau_x) \leftarrow \mathsf{ProbGen}_{\mathsf{sk}}(x)$ and $\sigma_y \leftarrow \mathsf{Compute}_{\mathsf{pk}}(\sigma_x)$ then $(1, y = f(x)) \leftarrow \mathsf{Verify}_{\mathsf{sk}}(\tau_x, \sigma_y)$*

In other words, every correct computation should always output $acc = 1$ in the Verify algorithm.

Informally, we can define *privacy* when the public outputs of the problem generation algorithm $\mathsf{ProbGen}$ over two different inputs are indistinguishable. More formally, we consider the following security experiment. In this experiment we allow the adversary to call oracles of the $\mathsf{ProbGen}$ and Verify algorithms.



In this experiment, the Adversary can query for the encoded public value σ_x as much as it wants. This is generated by the Challenger, who sends σ_x and stores the private value τ_x. Notice that the Adversary can call for the acceptance bit to the Challenger as an oracle for some pair $(\cdot, \tau_{x'})$ many times but without getting the decryption value y.

Definition 2.7 (Privacy)**.** *For a verifiable computation scheme $\mathcal{VC}$, we define the advantage of an adversary $\mathcal{A}$ in the experiment above as:*

$$Adv_{\mathcal{A}}^{Priv}(\mathcal{VC}, f, \lambda) = \Pr[\boldsymbol{Exp}_{\mathcal{A}}^{Priv}[\mathcal{VC}, f, \lambda] = 1]$$

A verifiable computation scheme $\mathcal{VC}$ is private for a function f, if for any adversary $\mathcal{A}$ running in probabilistic polynomial time:

$$Adv_{\mathcal{A}}^{Priv}(\mathcal{VC}, f, \lambda) \leq \frac{1}{2} + \mathbf{negl}(\lambda)$$

One last property that we want in our $\mathcal{VC}$ scheme is to be sound. Intuitively, a verifiable computation scheme is sound if a malicious worker cannot persuade the verification algorithm to accept an incorrect output. In other words, for a given function f and input x, a malicious worker should not be able to convince the verification algorithm to output $\hat{y}$ such that $f(x) \neq \hat{y}$. Below, we formalise this intuition with an experiment, where $poly(\cdot)$ is a polynomial. [GGP10]

$$\textbf{Experiment } \mathbf{Exp}_{\mathcal{A}}^{Sound}[\mathcal{VC}, f, \lambda]$$

$(\mathsf{pk}, \mathsf{sk}) \leftarrow\!\!\$\ \mathsf{KeyGen}(f, 1^\lambda)$;
For $i = 1, \ldots, l = poly(\lambda)$;
    $x_i \leftarrow \mathcal{A}(\mathsf{pk}, x_1, \sigma_1, \ldots, x_i, \sigma_i)$;
    $(\sigma_i, \tau_i) \leftarrow \mathsf{ProbGen}_{\mathsf{sk}}(x_i)$;
For $j = 1, \ldots, l = poly(\lambda)$;
    $(acc, \cdot) \leftarrow \mathcal{A}^{\mathsf{PubVer}}(j, \hat{\sigma_j})$
$(i, \sigma_y^*) \leftarrow \mathcal{A}(\mathsf{pk}, x_1, \sigma_1, \ldots, x_l, \sigma_l)$;
$(acc, \hat{y}) \leftarrow \mathsf{Verify}_{\mathsf{sk}}(\tau_i, \hat{\sigma}_y)$
If $acc = 1$ and $\hat{y} \neq f(x_i)$, output '1', else '0';

Essentially, the adversary is given oracle access to generate the encoding of multiple problem instances. Also, it is given access to a verification algorithm oracle $\mathcal{A}^{\mathsf{PubVer}}$ that can returns only the acceptance bit for a given input $(j, \hat{\sigma})$. The adversary succeeds if it produces an output that convinces the verification algorithm to accept on the wrong output value for a given input value. We can now define the soundness of the system based on the adversary's success in the above experiment.

Definition 2.8 (Soundness)**.** *For a verifiable computation scheme $\mathcal{VC}$, we define the advantage of an adversary $\mathcal{A}$ in the experiment above as:*

$$Adv_{\mathcal{A}}^{Sound}(\mathcal{VC}, f, \lambda) = \Pr[\boldsymbol{Exp}_{\mathcal{A}}^{Sound}[\mathcal{VC}, f, \lambda] = 1]$$

A verifiable computation scheme $\mathcal{VC}$ is sound for a function f, if for any adversary $\mathcal{A}$ running in probabilistic polynomial time:

$$Adv_{\mathcal{A}}^{Sound}(\mathcal{VC}, f, \lambda) \leq \mathbf{negl}(\lambda)$$

Chapter 3

Homomorphisms in cryptography

Homomorphic properties are used in many cryptographic schemes. The homomorphisms are valuable in scenarios where data privacy is crucial, such as cloud computing and outsourced data processing. By performing operations on encrypted data, sensitive information remains hidden from third parties, including service providers or intermediaries. Those can be found in *state of the art* schemes like (fully) homomorphic encryption [Gen09] [BV11] or even other non-encryption schemes where we care about the integrity of the data [AB09].

In this chapter, first we will introduce how the data integrity can be ensured with the Message Authentication Codes (MACs) and its homomorphic version (HomMACs). Afterwards, we will introduce some homomorphic encryption schemes and its variants with a detailed *somewhat homomorphic encryption* [BV11].

3.1 Authentication algorithms

As we introduced before, authenticity is very important when data is sent. There exist several ways to authenticate data using signature schemes, like Schnorr [Sch90], DSA or ECDSA [JMV01] (based on elliptic curves). The authenticity of these cryptographic signatures schemes can be checked publicly by just applying a public key. Related with these algorithms, there exists a private way to authenticate data where a user (who knows the secret key) can authenticate it if the data comes from a user (can be itself) that knows the secret key. These algorithms are called Message Authentication Codes (MACs) and their homomorphic version (HomMACs) will be very helpful on our scheme.

3.1.1 Message Authentication Codes (MACs)

Suppose two parties (A, B), who share a secret key, wish to ensure that data transmitted between them has not been tampered with. They can then use the shared secret key and a keyed algorithm to produce a tag, or MAC, which is sent with the data. Essentially, a Message Authentication Code is a set of 3-tuple algorithms $(\mathsf{KeyGen}, \mathsf{Auth}, \mathsf{Ver})$ defined as follows:

- $\mathsf{KeyGen}(1^\lambda)$: on input the security parameter λ, the key generation algorithm outputs a secret key sk.
- $\mathsf{Auth}(\mathsf{sk}, m)$: given the secret key sk, and an input message $m \in \mathcal{M}$, it outputs a tag σ.
- $\mathsf{Ver}(\mathsf{sk}, m, \sigma)$: given the secret key sk, a message $m \in \mathcal{M}$ and a tag σ, the verification algorithm outputs 0 (reject) or 1 (accept).

Note we do not assume that the message is secret: we are trying to protect data integrity and not confidentiality.

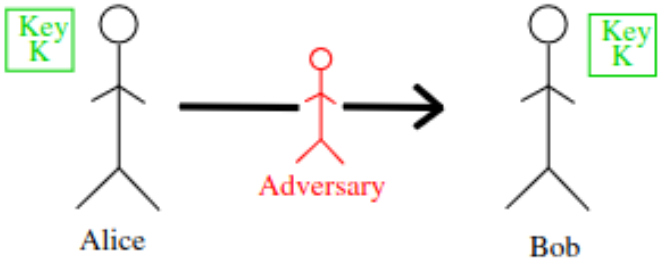
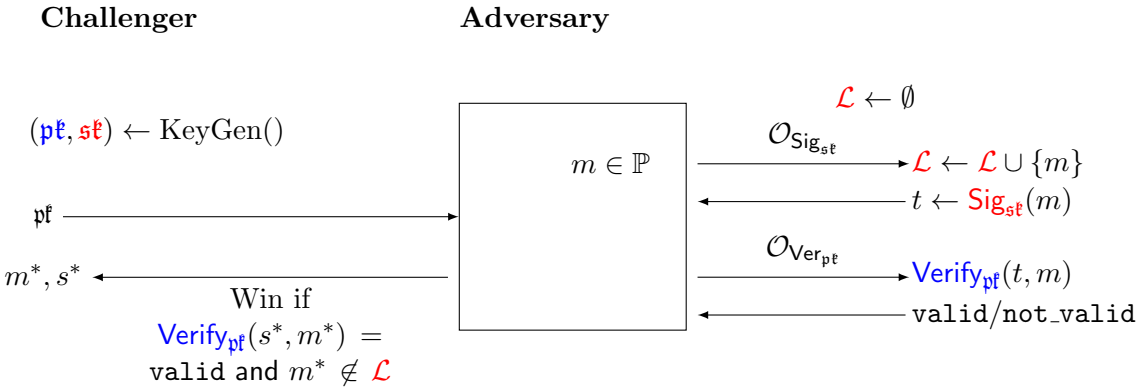


Figure 3.1: *The Authentication Problem. User A wishes to send a message along some communications link to user B, but an active adversary may insert information into the channel. Our goal is to provide B with a method to determine which received messages really came from A.[Bla00]*

3.1.2 Security on MACs

The security notion we want in a MAC is the unforgeability, which is captured on an **EUF-CMA** (Existential Unforgeability under Chosen-Message-Attack) game. The intuitive idea is that no efficient adversary should be able to generate a valid tag on any "new" message that was previously sent (and authenticated).

The EUF-CMA security game (Generalised)



In our case we consider the Signature algorithm as the Auth algorithm and a symmetric key scheme. The adversary wins if he/she can produce a valid signature for a message $m^* \notin \mathcal{L}$. In a more mathematically way, we will consider the following experiment for a message authentication code $\Pi = (\mathsf{KeyGen}, \mathsf{Auth}, \mathsf{Ver})$ an adversary $\mathcal{A}$ and security parameter λ [KL20].

The message authentication forge experiment $\mathsf{MAC\text{-}forge}_{\mathcal{A},\Pi}(\lambda)$*:*

1. *A key is generated by running* $\mathsf{KeyGen}(1^\lambda)$
2. *The adversary* $\mathcal{A}$ *is given input* 1^λ *and oracle access to* $\mathsf{Auth}(\mathsf{sk}, \cdot)$*. The adversary eventually outputs* (m, t)*. Let* $\mathcal{L}$ *denote the set of all queries that* $\mathcal{A}$ *submitted to its oracle.*
3. $\mathcal{A}$ *succeeds if (1)* $\mathsf{Ver}(\mathsf{sk}, m, t) = 1$ *and (2)* $m \notin \mathcal{L}$*. In that case the output of the experiment is defined to be 1.*

Definition 3.1. *A message authentication code* $\Pi = (\mathsf{KeyGen}, \mathsf{Auth}, \mathsf{Ver})$ *is existentially unforgeable under chosen-message-attack if for all probabilistic polynomial-time adversaries* $\mathcal{A}$*, there is a negligible function negl such that:*

$$\Pr[\mathit{MAC\text{-}forge}_{\mathcal{A},\Pi}(\lambda) = 1] \leq \mathit{negl}(\lambda)$$

3.1.3 Homomorphic Message Authentication Codes (HomMACs)

Here we define the formal definition of a HomMAC as it is described in [CF13], which is not too different from the MAC formal definition.

An Homomorphic Message Authentication Code is a set of 4-tuple algorithms $(\mathsf{KeyGen}, \mathsf{Auth}, \mathsf{Eval}, \mathsf{Ver})$ where the only difference with a normal MAC is the Eval algorithm:

- $\mathsf{KeyGen}(1^\lambda)$: on input the security parameter λ, the key generation algorithm outputs a secret key sk and a public evaluation key ek.
- $\mathsf{Auth}(\mathsf{sk}, \tau, m)$: given the secret key sk, and an input message $m \in \mathcal{M}$ (associated with a label τ); it outputs a tag σ.
- $\mathsf{Eval}(\mathsf{ek}, f, \hat{\sigma})$: on input the evaluation key ek, a circuit $f : \mathcal{M}^n \to \mathcal{M}$ and a vector of tags $\hat{\sigma} = (\sigma_1, \ldots, \sigma_n)$, the evaluation algorithm outputs a new tag σ. Informally, if the tags $\sigma_1, \ldots, \sigma_n$ verifies for $m_1, \ldots, m_n$, then σ will verify for $f(m_1, \ldots, m_n)$.
- $\mathsf{Ver}(\mathsf{sk}, m, \sigma, f, \tau_1, \ldots, \tau_n)$: given the secret key sk, a message $m \in \mathcal{M}$ the tag σ, the circuit f and the labels $\tau_1, \ldots, \tau_n$, the verification algorithm outputs 0 (reject) or 1 (accept).

3.1.4 A brief example, the CF13 HomMAC

As an example of a HomMAC, we present the one detailed in the third section of [CF13] whose security relies on a pseudo-random function and a secret point x.

- $\mathsf{KeyGen}(1^\lambda)$. Let p be a prime of roughly λ bits. Choose a seed $K \leftarrow \mathcal{K}$ of a pseudo-random function $F_K : \{0,1\}^* \to \mathbb{Z}_p$ and a random value $x \leftarrow \mathbb{Z}_p$. Output $\mathsf{sk} = (K, x), \mathsf{ek} = p$ and let the message space $\mathcal{M}$ be $\mathbb{Z}_p$.
- $\mathsf{Auth}(\mathsf{sk}, \tau, m)$. To authenticate a message $m \in \mathbb{Z}_p$ with label $\tau \in \{0,1\}^\lambda$, compute $r_\tau = F_K(\tau)$, set $y_0 = m$, $y_1 = \frac{(r_\tau - m)}{x} \bmod p$ and output $\sigma = (y_0, y_1)$. Basically, y_0, y_1 are the coefficients of a degree-1 polynomial $y(z)$ with the special property that it evaluates to m on the point 0 ($y(0) = m$), and it evaluates to r_τ on a hidden random point x ($y(x) = r_\tau$).

  In their construction, they interpret tags σ as polynomials $y \in \mathbb{Z}_p[z]$ of degree $d \geq 1$ in some variable z, i.e., $y(z) = \sum_i y_i z^i$.
- $\mathsf{Eval}(\mathsf{ek}, f, \sigma)$. The homomorphic evaluation algorithm takes as input the evaluation key $\mathsf{ek} = p$, an arithmetic circuit $f : \mathbb{Z}_p^n \to \mathbb{Z}_p$, and a vector σ of tags $(\sigma_1, \ldots, \sigma_n)$. Intuitively, Eval consists in evaluating the circuit f on the tags $\sigma_1, \ldots, \sigma_n$ instead of evaluating it on messages. However, since the values σ_i 's are not messages in $\mathbb{Z}_p$, but rather are polynomials $y \in \mathbb{Z}_p[z]$, they specify how this evaluation is carried through. Eval proceeds gate-by-gate as follows. At each gate g, given two tags σ_1, σ_2 (or a tag σ_1 and a constant $c \in \mathbb{Z}_p$), it runs the algorithm $\sigma \leftarrow \mathsf{GateEval}(\mathsf{ek}, g, \sigma_1, \sigma_2)$ described below that returns a new tag σ, which is in turn passed on as input to the next gate in the circuit.

– $\mathsf{GateEval}(\mathsf{ek}, g, \sigma_1, \sigma_2)$. Let $\sigma_i = y^{(i)} = (y_0^{(i)}, \ldots, y_{d_i}^{(i)})$ for $i = 1, 2$ and $d_i \geq 1$.

If $g = +$, then:

* let $d = \max(d_1, d_2)$. Here we assume without loss of generality that $d_1 \geq d_2$ (i.e., $d = d_1$).

* Compute the coefficients $(y_0, \ldots, y_d)$ of the polynomial $y(z) = y^{(1)}(z) + y^{(2)}(z)$, if needed, padded with zeroes in positions $d_1 + 1, \ldots, d_2$).

If $g = \mathsf{x}$, then:

* let $d = d_1 + d_2$.

* Compute the coefficients $(y_0, \ldots, y_d$) of the polynomial $y(z) = y^{(1)}(z) * y^{(2)}(z)$ using the convolution operator *, i.e., $\forall k = 0, \ldots, d$, define $y_k = \sum_{i=0}^{k} y_i^{(1)} * y_{k-i}^{(2)}$.

If $g = \mathsf{x}$ and one of the two inputs, say σ_2 , is a constant $c \in \mathbb{Z}_p$. This case can be seen as a special case of multiplication if one think of σ_2 as a degree-0 polynomial $y^{(2)}(z)$ such that $y^{(2)}(0) = y^{(2)}(x) = \mathrm{c}$.

Return $\sigma = (y_0, \ldots, y_d)$.

As one can notice, the size of a tag grows only after the evaluation of a multiplication gate (where both inputs are not constants). It is not hard to see that after the homomorphic evaluation of a circuit f , it holds $|\sigma| = d + 1$, where d is the degree of f.

- $\mathsf{Ver}(\mathsf{sk}, m, \mathcal{P}, \sigma)$. Let $\mathcal{P} = (f, \tau_1, \ldots, \tau_n)$ be a labelled program, $m \in \mathbb{Z}_p$ and $\sigma = (y_0, \ldots, y_d)$ be a tag for some $d \geq 1$. Verification proceeds as follows:
  - If $y_0 \neq m$, then output 0 (reject). Otherwise continue as follows.
  - For every input wire of f with label τ compute $r_\tau = F_K(\tau)$.
  - Next, evaluate the circuit on $r_{\tau_1} \ldots r_{\tau_n}$, i.e., compute $\rho \leftarrow f(r_{\tau_1}, \ldots, r_{\tau_n})$ and use x to check whether the following equation holds:

$$\rho = \sum_{k=0}^{d} y_k x^k$$

  If this is true, then output 1. Otherwise output 0.

  Observe that the above applies also to identity programs $\mathcal{I}_t$, in which case the algorithm just checks that $r_\tau = y_0 + y_1 \cdot x$ and $y_0 = m$.

Here, Catalano and Fiore proved that for every gate $\{+, \mathsf{x}\}$ the HomMAC preserved its nice structure so the Ver algorithm would satisfy correctness. In addition, they showed how their HomMAC had "**HomUF-CMA** security", based on the homomorphic property of the HomMAC combined with the **EUF-CMA** security. Every HomMAC (and in particular the one mentioned above) has authentication and evaluation correctness [Cat+14].

Authentication Correctness. Intuitively, an homomorphic MAC satisfies this property if any tag σ generated by the algorithm $\mathsf{Auth}(\mathsf{sk}, \tau, m)$ authenticates with respect to the identity program $\mathcal{I}_\tau$. Formally, we require that for any message $m \in \mathcal{M}$, all keys $(\mathsf{sk}, \mathsf{ek}) \leftarrow\!\!\$\ \mathsf{KeyGen}(1^\lambda)$, any label $\tau \in \{0,1\}^*$ and any tag

$\sigma \leftarrow \mathsf{Auth}(\mathsf{sk}, \tau, m)$, it holds: $\Pr[\mathsf{Ver}(\mathsf{sk}, m, \mathcal{I}_\tau, \sigma) = 1] = 1$.

Evaluation Correctness. Informally, this property states that if the evaluation algorithm is given a vector of tags $\sigma = (\sigma_1, \ldots, \sigma_n)$ such that each σ_i authenticates some message m_i as the output of a labelled program $\mathcal{P}_i$, then the tag σ produced by Eval must authenticate $f(m_1, \ldots, m_n)$ as the output of the composed program $f(\mathcal{P}_1, \ldots, \mathcal{P}_n)$.
More formally, let us fix a pair of keys $(\mathsf{sk}, \mathsf{ek}) \leftarrow\!\$\ \mathsf{KeyGen}(1^\lambda)$, a function $g : \mathcal{M}^t \rightarrow \mathcal{M}$ and any set of message/program/tag triples $(m_i, \mathcal{P}_i, \sigma_i)_{i=1}^t$ such that $\mathsf{Ver}(\mathsf{sk}, m_i, \mathcal{P}_i, \sigma_i) = 1$. If $m^* = g(m_1, \ldots, m_t)$, $\mathcal{P}^* = g(\mathcal{P}_1, \ldots, \mathcal{P}_t)$, and $\sigma^* = \mathsf{Eval}(\mathsf{ek}, g, (\sigma_1, \ldots, \sigma_t))$, then it must hold: $\mathsf{Ver}(\mathsf{sk}, m^*, \mathcal{P}^*, \sigma^*) = 1$.

3.1.5 Security on the CF13 HomMAC

The principal problem for the adversary is still producing a valid signature pair (σ^*, m^*) but since we assume f to be an arithmetic circuit, we have to clearly define what is an honest output, as Catalano and Fiore defined in [CF13], which is also explained in [Cat+14].

Let HomMAC be a homomorphic MAC scheme as defined in 3.1.3. Consider the following experiment $\mathsf{HomUF-CMA}_{\mathcal{A},\mathsf{HomMAC}(\lambda)}$ between a challenger and an adversary $\mathcal{A}$ against HomMAC:

The homomorphic message authentication forge experiment $\mathit{HomUF\text{-}CMA}_{\mathcal{A},\mathsf{HomMAC}}(\lambda)$

Setup. The challenger generates $(\mathsf{sk}, \mathsf{ek}) \leftarrow\!\$\ \mathsf{KeyGen}(1^\lambda)$ and gives ek to $\mathcal{A}$. Also a list $T = \emptyset$ is initialised.

Authentication queries. The adversary can adaptively ask for tags on label-message pairs of its choice. Given a query $(\tau,\ m)$, if $(\tau,\ m) \in T$ (i.e., the query was previously made), then the challenger replies with the same tag generated before. If T already contains a pair $(\tau,\ m') \in T$ with $m' \neq m$ (i.e., the label τ was already queried with a different message), then the challenger ignores the query. Otherwise, if $(\tau, m) \notin T$, the challenger computes $\sigma \leftarrow\!\$\ \mathsf{Auth}(\mathsf{sk}, \tau, m)$, returns σ to $\mathcal{A}$ and updates the list $T = T \cup (\tau, m)$.

Verification queries. The adversary is also given access to a verification oracle. $\mathcal{A}$ can submit a query $(m, \mathcal{P}, \sigma)$ and the challenger replies with the output of $\mathsf{Ver}(\mathsf{sk}, m, \mathcal{P}, \sigma)$.

Forgery. When the adversary stops running, the experiment outputs 1 if one of the verification queries made by A, say $(m^*, \mathcal{P}^*, \sigma^*)$, is a forgery.

The description of the experiment is thus concluded by defining what is a forgery. To this end, we first recall the notion of a *well-defined* program with respect to a list T. Informally, there are two ways for a program $\mathcal{P}^* = (f^*, \tau_1^*, \ldots, \tau_n^*)$ to be well-defined. Either all the τ_i^*'s are in T or, if there are some labels τ_i^* that are not in T, then the inputs associated with such labels are "ignored" by f^* when computing the output. In other words, inputs corresponding to labels not in T do not affect the behaviour of f^* in any way.

More formally, a labelled program $\mathcal{P}^* = (f^*, \tau_1^*, \ldots, \tau_n^*)$ is *well defined on* T if either one of the following two cases occurs:

1. there exists $i \in \{1, \ldots, n\}$ such that $(\tau_i^*, \cdot) \notin T$ (i.e., $\mathcal{A}$ never asked an authentication query with label τ_i^*, so m is not authenticated), and $f^*(\{m_j\}_{(\tau_j, m_j) \in T} \cup \{\tilde{m}_j\}_{(\tau_j, \cdot) \notin T})$ outputs the same value for all possible choices of $\tilde{m}_j \in M$. Here, the notation $f(\{m_j\}_{(\tau_j, m_j)}$ given by Catalano and Fiore is a shorthand for referring to an execution of f where the input wire labelled by τ_j is assigned value m_j.

2. T contains tuples $(\tau_1^*, m_1), \ldots, (\tau_n^*, m_n)$, for some messages $m_1, \ldots, m_n$.

Then we say that a verification query $(m^*, \mathcal{P}^* = (f^*, \tau_1^*, \ldots, \tau_n^*), \sigma^*)$ is a forgery if and only if $\mathsf{Ver}(\mathsf{sk}, m^*, \mathcal{P}^*, \sigma^*) = 1$ and one of the following conditions holds:

1. *Type 1 Forgery*: $\mathcal{P}^*$ is not well-defined on T.
2. *Type 2 Forgery*: $\mathcal{P}^*$ is well-defined on T and $m^* \neq f^*(\{m_j\}_{(\tau_j, m_j) \in T})$, i.e., m^* is not the correct output of the labelled program $\mathcal{P}^*$ when executed on previously authenticated messages.

Definition 3.2. *An homomorphic message authentication code* $\Pi = (\mathsf{KeyGen}, \mathsf{Auth}, \mathsf{Eval}, \mathsf{Ver})$ *is homomorphic existentially unforgeable under chosen-message-attack if for all probabilistic polynomial-time adversaries* $\mathcal{A}$*, there is a negligible function* negl *such that:*

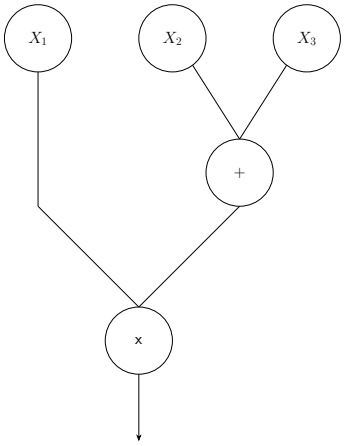
$$\Pr[\mathsf{HomUF} - \mathsf{CMA}_{\mathcal{A},\Pi}(\lambda) = 1] \leq \mathsf{negl}(\lambda)$$

Once we captured its security, we will now show how this HomMAC works using a very simple example:

Example

Consider the function $f(X_1, X_2, X_3) = X_1 \cdot (X_2 + X_3)$
$X_i \in \mathbb{Z}_p$

Whose arithmetic circuit is represented as:



First we set our secret and evaluation keys:

$\mathsf{KeyGen}(1^\lambda) \longrightarrow (\{K, x\}, p)$

We want to authenticate $\hat{x} = f(x_1, x_2, x_3)$, so we first authenticate each element x_i separately (we see those authentications as polynomials) and then we run the Eval algorithm over the tags:

- $\mathsf{Auth}(\mathsf{sk}, \tau_1, x_1) \longrightarrow \sigma_1 = (y_0^{(1)}, y_1^{(1)}) = (x_1, \frac{(r_{\tau_1} - x_1)}{x} \bmod p) \longleftrightarrow \sigma_1(Z) = y_0^{(1)} + y_1^{(1)} Z$
- $\mathsf{Auth}(\mathsf{sk}, \tau_2, x_2) \longrightarrow \sigma_2 = (y_0^{(2)}, y_1^{(2)}) = (x_2, \frac{(r_{\tau_2} - x_2)}{x} \bmod p) \longleftrightarrow \sigma_2(Z) = y_0^{(2)} + y_1^{(2)} Z$
- $\mathsf{Auth}(\mathsf{sk}, \tau_3, x_3) \longrightarrow \sigma_3 = (y_0^{(3)}, y_1^{(3)}) = (x_3, \frac{(r_{\tau_3} - x_3)}{x} \bmod p) \longleftrightarrow \sigma_3(Z) = y_0^{(3)} + y_1^{(3)} Z$

Now, we run the Eval algorithm.

For the first gate, $g_1 = +$, then:

$$\sigma_+(Z) = \sigma_2(Z) + \sigma_3(Z) =$$

$$= y_0^{(2)} + y_1^{(2)} Z + y_0^{(3)} + y_1^{(3)} Z =$$
$$= (y_0^{(2)} + y_0^{(3)}) + (y_1^{(2)} Z + y_1^{(3)} Z) \equiv y_0^{(+)} + y_1^{(+)} Z$$

For the second gate, $g_2 = \mathsf{x}$, then:

$$\sigma(Z) = \sigma_1(Z) * \sigma_+(Z) =$$
$$= (y_0^{(1)} + y_1^{(1)} Z) * (y_0^{(+)} + y_1^{(+)} Z) =$$
$$= (y_0^{(1)} \cdot y_0^{(+)}) + (y_0^{(1)} \cdot y_1^{(+)} + y_0^{(+)} \cdot y_1^{(1)}) Z + (y_1^{(1)} \cdot y_1^{(+)}) Z^2 \equiv y_0 + y_1 Z + y_2 Z^2$$

Here we note that the length of the authenticator polynomial is what Catalano and Fiore showed it should has, $d + 1$ elements where d is the degree of f.

Now, for the Ver algorithm:

1. One can easily check that $\hat{x} = \sigma(0) = y_0 = y_0^{(1)} \cdot y_0^{(4)} = y_0^{(1)} \cdot (y_0^{(2)} + y_0^{(3)}) = x_1 \cdot (x_2 + x_3)$
2. It is also proven that $\rho = \sigma(x) = y_0 + y_1 x + y_2 x^2 = (x_1 \cdot (x_2 + x_3)) + x \cdot (x_1 \cdot (\frac{(r_{\tau_3} + r_{\tau_2} - x_3 - x_2)}{x}) + (x_2 + x_3) \cdot \frac{(r_{\tau_1} - x_1)}{x}) + x^2 \cdot (\frac{(r_{\tau_1} - x_1)}{x} \cdot \frac{(r_{\tau_3} + r_{\tau_2} - x_3 - x_2)}{x}) = r_{\tau_1} \cdot (r_{\tau_2} + r_{\tau_3}) \equiv f(r_{\tau_1}, \ldots, r_{\tau_n})$

3.2 Encryption Schemes

Talking about encryption in cryptography, covers a wide range of methods, including symmetric key encryption, asymmetric key encryption, and homomorphic encryption. Each type of scheme offers distinct advantages and is suitable for specific use cases. Symmetric key encryption schemes [Hil29] employs a shared secret key to encrypt and decrypt data, providing fast and efficient cryptographic operations. Asymmetric key encryption, on the other hand, employs a pair of public and private keys, enabling secure communication and key exchange between parties. One famous asymmetric key encryption is RSA [RSA78].

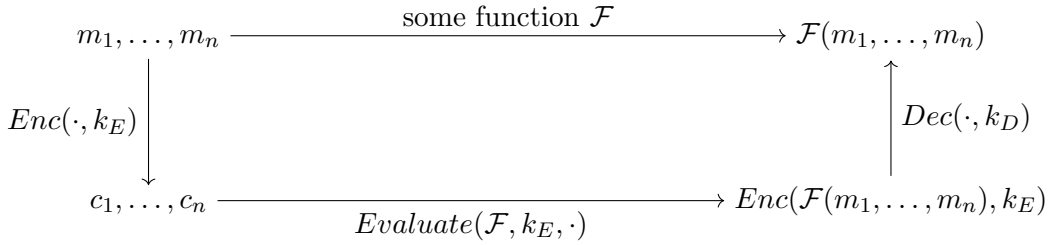
3.2.1 Homomorphic Encryption Schemes

The concept of homomorphic encryption allows for computations to be done directly on the data being encrypted, safeguarding privacy while allowing for data analysis and manipulation [RAD+78]. This attribute can be beneficial since it allows a party to acquire computed data in a two different approaches, as it is shown in the figure bellow. The homomorphic encryption scheme we will be using [BV11] on the following chapters uses an asymmetric public and private key encryption.

Definition 3.3. *Let $\mathcal{F}$ be a set of functions in $\{0,1\}^* \to \{0,1\}$. A public key scheme is $\mathcal{F}-$homomorphic if there exists an evaluation algorithm Eval s.t. $\mathsf{Dec}_{\mathrm{sk}}(\mathsf{Eval}(f, \mathsf{Enc}_{\mathrm{pk}}(x)) = f(x)$ for all $f \in \mathcal{F}$ and $x \in \{0,1\}^*$ of appropriate length [Bra19].*

One can differentiate homomorphic encryption schemes by $\mathcal{F}$. A *fully homomorphic encryption* (FHE) is a homomorphic encryption scheme where $\mathcal{F}$ is the set of all functions. A *somewhat homomorphic encryption* (SHE) is a homomorphic encryption scheme where $\mathcal{F}$ can support a *limited* number of operations such as circuits of a certain multiplicative depth. A *partially homomorphic encryption* (PHE) is a homomorphic encryption scheme where $\mathcal{F}$ only support the evaluation of circuits consisting of only one type of gate, *e.g.* addition or multiplication.

A very nice representation of how an $\mathcal{F}$-homomorphic encryption scheme works with an encryption and decryption keys (k_E, k_D) is the one as follows:



Essentially, one can get to the desired computed value $\mathcal{F}(m_1, \ldots, m_n)$ in two ways: One is just applying the $\mathcal{F}$ function over the plaintexts $m_1, \ldots, m_n$. The second one is cipher the values $m_1, \ldots, m_n$ applying the Eval algorithm to the encrypted data values $c_1, \ldots, c_n$ and decrypt the obtained value to get $\mathcal{F}(m_1, \ldots, m_n)$.

Some famous *partially homomorphic encryption schemes* are RSA [RSA78] and ElGamal [ElG85]. These two encryption schemes can only support the product on their encrypted messages. For example, in the RSA cryptosystem, encrypting a message m involves simply raising it to the power e and taking the result modulo $N = p \cdot q$ where p, q are prime numbers i.e. $c = m^e$. It is not hard to see that the product of two ciphertexts c_1 and c_2 encrypting messages m_1 and m_2 allows to compute the value $c_1 \cdot c_2 \pmod N = (m_1 m_2)^e \pmod N$. However, the addition of two ciphertexts $c_1 + c_2$ it is not supported as the encrypted value of $m_1 + m_2$. To achieve the two operations (addition and multiplication) we have to think deeper and look for more complex encryption schemes.

3.2.2 Security on encryption schemes

In cryptography, the concept of security in an encryption system is defined by the notion of *perfect security* or *semantic security* [Sma16]. Given an encryption system and an output ciphertext from it, we say that the scheme has perfect security if an adversary with infinite computing power can learn nothing about the plaintext given the ciphertext. Semantic security is like perfect security but we only allow an adversary with polynomially bounded computing power.

The trouble with the definition of semantic security is that it is hard to show that a given encryption scheme has this property. Indistinguishability of encryptions, or IND security for short, is a much easier property to confirm for a given system. As the IND security is defined, the adversary has to guess between two given encryptions, so a system that has IND security then it also has semantic security. Hence, to show that a system is semantically secure all we need to do is show that it is IND secure. In essence, there are two types of IND security: Indistinguishability under Chosen Plain Attacks (IND-CPA) and Indistinguishability under Chosen Ciphertexts Attacks (IND-CCA). We will explain them as security games.

IND-CPA Encryption game

IND-CPA Encryption game

Challenger		**Adversary**
$(\mathsf{pk}, \mathsf{sk}) \leftarrow\!\!\$\ \mathsf{KeyGen}(1^\lambda)$		
	$\xrightarrow{\mathsf{pk}}$	
		$m_0, m_1 \in \mathcal{M}$
	$\xleftarrow{m_0, m_1}$	
$b \leftarrow\!\!\$\ \{0,1\}$		
$c_b = \mathsf{Enc}_{\mathsf{pk}}(m_b)$		
	$\xrightarrow{c_b}$	
		$b' \in \{0,1\}$
		Wins if $b = b'$

Basically, an adversary should not distinguish the encryption over two different messages and its winning probability should not be more than like flipping a coin.

Definition 3.4 (IND-CPA Security)**.** *Let* Π *be cryptographic scheme.* Π *is IND-CPA secure if for any adversary* $\mathcal{A}$ *running in probabilistic polynomial time,*

$$\mathrm{Adv}_{\Pi}^{\text{IND-CPA}}(\mathcal{A}) = \Pr[\mathcal{A} \text{ wins}] = \frac{1}{2} + \mathbf{negl}(\lambda)$$

IND-CCA Encryption game

IND-CCA Encryption game

Challenger		**Adversary**
$(\mathsf{pk}, \mathsf{sk}) \leftarrow\!\!\$\ \mathsf{KeyGen}(1^\lambda)$		
	$\xrightarrow{\mathsf{pk}}$	
		$c_0, c_1 \in \mathcal{C}$
		$m_i \rightleftarrows \mathcal{O}_{\mathsf{Dec}}(\mathsf{sk}, c_i)$
	$\xleftarrow{m_0, m_1}$	
$b \leftarrow\!\!\$\ \{0,1\}$		
$c_b = \mathsf{Enc}_{\mathsf{pk}}(m_b)$		
	$\xrightarrow{c_b}$	
		$m \rightleftarrows \mathcal{O}_{\mathsf{Dec}}(\mathsf{sk}, c)$
		$b' \in \{0,1\}$
		Wins if $b = b'$

Now we consider that the adversary has the ability of choosing ciphertexts and calling an "oracle" $\mathcal{O}_{\mathsf{Dec}}$ to get its Decryption whenever it wants. The only request it is needed for the game to be correct is that the adversary cannot ask the Decryption Oracle the challenge c_b.

Definition 3.5 (IND-CCA Security)**.** *Let* Π *be cryptographic scheme.* Π *is IND-CCA secure if for any adversary* $\mathcal{A}$ *running in probabilistic polynomial time,*

$$\mathrm{Adv}_{\Pi}^{\text{IND-CCA}}(\mathcal{A}) = \Pr[\mathcal{A} \text{ wins}] = \frac{1}{2} + \mathbf{negl}(\lambda)$$

In both cases we consider $\Pi = \mathsf{Enc}$. Notice that any IND-CPA/CCA secure cryptosystem must be randomized and yield different encryptions of the same plaintext. Otherwise, the adversary could just query the decryption oracle on encryptions of m_0 and m_1 and then compare the resulting ciphertexts.

Proposition 3.1. *As it is defined, an homomorphic encryption scheme is not IND-CCA secure.*

Proof. We recall that the only requirement we demand in the IND-CCA to be honest is that the adversary cannot send to the Challenger messages that comes from a previous Decryption Oracle call. Also, the Adversary cannot call the Decryption Oracle for c_b.

In general, homomorphic encryption does not meet IND-CCA security and this statement is well studied in literature [CT15]. The proof we give for this is a counter example, considering the homomorphic function $f(x) = 2x$.

IND-CCA game. Homomorphic encryption attack

Challenger		**Adversary**
$(\mathsf{pk}, \mathsf{sk}) \leftarrow\!\!\$\ \mathsf{KeyGen}(1^\lambda)$		
	$\xrightarrow{\mathsf{pk}}$	
		$c \in \mathcal{C}$
		$m \rightleftarrows \mathcal{O}_{\mathsf{Dec}}(\mathsf{sk}, c)$
		$m_0 \leftarrow m$
		$m_1 \leftarrow m_0 + 1$
	$\xleftarrow{m_0, m_1}$	
$b \leftarrow\!\!\$\ \{0,1\}$		
$c_b = \mathsf{Enc}_{\mathsf{pk}}(m_b)$		
	$\xrightarrow{c_b}$	
		$c^* = c_b + \mathsf{Enc}_{\mathsf{pk}}(1)$
		$m^* \rightleftarrows \mathcal{O}_{\mathsf{Dec}}(\mathsf{sk}, c^*)$
		`if` $\mathtt{m}^* = \mathtt{m}_1$
		$b' = 0$
		`else`
		$b' = 1$
		Wins if $b = b'$

Since the adversary does not breaks the game standings, the proof is correct. We have proved that it is possible to distinguish in the IND-CCA game using an homomorphic function (in particular, this simple consideration). □

This proof is valid for any ring since we take $m_1 = m_0 + 1$. Note that, with this consideration, the advantage for the adversary in no longer similar than flipping a coin. In fact:

$$\mathrm{Adv}_{\Pi}^{\text{IND-CCA}}(\mathcal{A}) = \Pr[\mathcal{A} \text{ wins}] = 1$$

We cannot achieve IND-CCA security in a homomorphic encryption system, so we will only consider the IND-CPA security notion in our scheme 4.

3.2.3 LWE

The Learning with Errors Problem (LWE) has been an attractive theoretical computer science topic in these recent years [Reg09][Bal21]. The hardness of the LWE problem is essentially a reduction from worst-case lattice problems. Lattices-based problems are one the main building blocks of post-quantum cryptography because this assumption is believed to be challenging for both classical and quantum computers. Besides, this kind of constructions sometimes allows to a highly versatile cryptographic schemes [BV11] [Gen09] [GLP12] .

Let $\mathbb{Z}_q$ be the ring of integers modulo $q \geq 2$, and let $A \in \mathbb{Z}_q^{m \times n}$, $\boldsymbol{b} \in \mathbb{Z}_q^m$ and a secret $\boldsymbol{s} \in \mathbb{Z}_q^n$. Then, we can formulate a linear system of equations such as $A\boldsymbol{s} = \boldsymbol{b}$. If we are given $A, \boldsymbol{b}$, then it is easy to recover $\boldsymbol{s}$ by standard methods such as Gaussian elimination. Now, suppose that we are given A and a *noisy* vector $\boldsymbol{b}$, this is, $A\boldsymbol{s} + \boldsymbol{e} = \boldsymbol{b}$ where $\boldsymbol{e} \in \mathbb{Z}_q^m$ has small coefficients (and is not known). For example, we can consider the following system, with more equations than variables $n = 4$ (length of $\boldsymbol{s}$) [Reg10]:

$$
\begin{array}{rcrcrcrcrl}
14s_1 & + & 15s_2 & + & 5s_3 & + & 2s_4 & \approx & 8 & \mod 17 \\
13s_1 & + & 14s_2 & + & 14s_3 & + & 6s_4 & \approx & 16 & \mod 17 \\
6s_1 & + & 10s_2 & + & 13s_3 & + & 1s_4 & \approx & 3 & \mod 17 \\
10s_1 & + & 4s_2 & + & 12s_3 & + & 16s_4 & \approx & 12 & \mod 17 \\
9s_1 & + & 5s_2 & + & 9s_3 & + & 6s_4 & \approx & 9 & \mod 17 \\
3s_1 & + & 6s_2 & + & 4s_3 & + & 5s_4 & \approx & 16 & \mod 17 \\
 & & & & \vdots & & & & & \\
6s_1 & + & 7s_2 & + & 16s_3 & + & 2s_4 & \approx & 3 & \mod 17
\end{array}
$$

Where each equation is correct up to some small additive error (say, $\pm$ 1), and our goal is to recover $\boldsymbol{s}$. An answer in this case is $\boldsymbol{s} = (0, 13, 9, 11)$. If not for the error, finding $\boldsymbol{s}$ would be very easy. Introducing the error seems to make the problem significantly more difficult.

The hardness of LWE

Indeed, there are some reasons to believe the LWE problem is hard. One of them is because the best known algorithms for LWE run in exponential time (and even quantum algorithms do not seem to help). Another factor, as we said before, is because lattices-based problems are believed to be challenging. The hardness comes from a reduction of the worst-case hardness of standard lattice problems such as GAPSVP (the decision version of the shortest vector problem) [Pei09]. So, since the LWE problem is directly related to a hard problem, the hardness reduction proof is evident.

3.2.4 Ring-LWE

Although LWE is a very attractive problem, in practical applications it necessitates lengthy computation times and large key sizes. One LWE encryption system typically needs to provide at least n vectors $\{a_1, \ldots, a_n\}$, $a_i \in \mathbb{Z}_q^n$ leading to key sizes of order n^2. Looking at it from a practical point of view, this quadratic order should ideally be reduced to almost linear. Fortunately, there exists an efficient variation of the problem, introduced in [LPR13]: The Ring-Learning with Errors (Ring-LWE) problem. The main idea of Ring-LWE is using noisy ring products in place of linear equations to increase efficiency. The form for rings in Ring-LWE are meant to be $\mathbb{Z}_q[x]/\mathcal{I}$, where the ideal $\mathcal{I}$ is usually designated to be $\mathcal{I} = (x^n \pm 1)$.

Compared to LWE, using Ring-LWE has a significant efficiency advantage. For a single LWE sample $(\boldsymbol{a}, b = \langle \boldsymbol{a}, \boldsymbol{s} \rangle / q + e)$ we required a fresh vector $\boldsymbol{a} \in \mathbb{Z}_q^n$ and we obtained one pseudorandom value $b \in \mathbb{R}/\mathbb{Z}$. On the other hand, a single Ring-LWE sample $(a, b = a \cdot^1 s + e \mod x^n \pm 1)$ requires a ring element a (which

[1] As the ring product

is somewhat equivalent to a vector of $\mathbb{Z}_q^n$), but yields a full vector b of n coordinates. If we assume the hardness of the decision Ring-LWE problem, we will get n pseudorandom values from one sample being benefited from an efficient multiplication operation.

The security of Ring-LWE is not as straightforward as the security of LWE. The main concern of using Ring-LWE is that samples acquire an additional algebraic structure in the form of ideal lattices, induced by the ring, that might be exploited by some clever attacks. For example, there exist instances of primal Ring-LWE that are vulnerable to number-theoretical attacks [Eli+15].

3.2.5 The BV Encryption

We will now introduce the BV Encryption, a Ring-LWE encryption system. This system is meant to be IND-CPA secure since it is assumed Ring-LWE is hard. In the original version of this encryption [BV11], the authors described this as a *somewhat homomorphic encryption*. What we are going to consider is a version of this encryption, which it is explained on the appendix in [Boi+21]. Since its publication, this encryption system has had many variants. One of them, considering a relinearization so the size of the ciphertext could be considerably reduced [BV14]. In this version, described bellow, we consider functions $\hat{f}$ with degree D. Since this version does not consider this linearization, the size of the ciphertext is going to grow according to D.

We consider a function $\hat{f}$ with degree D. Let χ denote a discrete Gaussian distribution over the ring $\mathcal{R} := \mathbb{Z}[x]/(f)$ (where f is mean to be $x^N + 1$) with standard deviation σ. We identify $\mathcal{R}_q := \mathbb{Z}_q[x]/(f)$ as a subset of $\mathcal{R}$ when required (and similarity for $\mathcal{R}_t$). For a fresh message $m \in \mathcal{R}_t$ this version returns the ciphertext $c = (c_0, c_1, 0, \ldots, 0) \in \mathcal{R}_q^D$ The algorithms that run this version of the BV Encryption $=$ $(\mathsf{BV.KeyGen}, \mathsf{BV.Enc}, \mathsf{BV.Dec}, \mathsf{BV.Eval})$ are described as follows:

- $\mathsf{BV.KeyGen}(1^\lambda) \to (\mathsf{pk}, \mathsf{sk})$: sample a secret key $\mathsf{sk} = s \xleftarrow{\$} \chi$ and a public key $\mathsf{pk} = (a_0, b_0 = a_0 s + te_0)$ where $a_0 \xleftarrow{\$} \mathcal{R}_q$ and $e_0 \xleftarrow{\$} \chi$.

- $\mathsf{BV.Enc}(\mathsf{pk}, m) \to (c)$: sample $v, e' \xleftarrow{\$} \chi$ and $e'' \xleftarrow{\$} \chi'$ where χ' is the same as χ but with standard deviation $\sigma' > 2^{\omega(logN)} \cdot \sigma$. Then compute $(a, b) = (a_0 v + te', b_0 v + te'') \in \mathcal{R}_q^2$. Output $c = (c_0, c_1, 0, \ldots, 0)$ $\in \mathcal{R}_q^D$ where:
$$c_0 = m + b \ , \ c_1 = -a$$

- $\mathsf{BV.Dec}(\mathsf{sk}, m) \to (c)$: Using the secret key $\mathsf{sk} = s$, compute $c(s) = \sum_{i=0}^{D-1} c_i s^i$ and output $m = c(s)$ $mod\ t$.

- $\mathsf{BV.Eval}(g, c_1, \ldots, c_r) \to c_g$: addition and multiplication of two ciphertexts $c = \mathsf{Enc}(m)$, $c' = \mathsf{Enc}(m')$ $\in \mathcal{R}_q^D$ are defined as the usual addition and multiplication in $\mathcal{R}_q[Y]$:
  - $c^{(1)} + c^{(2)} = (c_0^{(1)} + c_0^{(2)}, \ldots, c_{D-1}^{(1)} + c_{D-1}^{(2)})$ encrypts $m + m'$.
  - for the multiplication of two ciphertexts $c^* = c^{(1)} \cdot c^{(2)} = (c_0^*, \ldots, c_{D-1}^*)$ we calculate by the convolution operator: $\forall k = 0, \ldots, D-1$, compute $c_k^* = \sum_{i=0}^{k} c_i^{(1)} \cdot c_{k-i}^{(2)}$

Note that in the case of multiplication, the result will not be correct if the degree of c (in Y) exceeds $D-1$.

Chapter 4

The generic VC Scheme

We now introduce our generic $\mathcal{VC}$ scheme to verify computations on encrypted data. This protocol combines an homomorphic encryption scheme (that can be FHE) along with an homomorphic MAC. Since this is a $\mathcal{VC}$ scheme, it should have all the security properties we introduced in 2. We will prove them in this own chapter via reduction games. Then, we will give precise estimations about the communication complexity.

4.1 The one-client protocol

We consider the following scenario. The client generates the public and secret keys $\{\mathsf{pk}, \mathsf{sk}\}$ for the homomorphic encryption scheme HE.Enc as well as a secret verification key vk and a evaluation key ek for the homomorphic MAC. Then it encrypts the messages $m_1, \ldots, m_k$ with the encryption scheme using pk. This results in ciphertexts $c_1, \ldots, c_k$. Then it authenticates these using the homomorphic MAC with secret key vk. This gives tags $\sigma_1, \ldots, \sigma_k$ on $c_1, \ldots, c_k$. It then sends $c_1, \ldots, c_k$ and $\sigma_1, \ldots, \sigma_k$ to the server. Using the homomorphic properties of the encryption scheme, the server first computes the function $f'(\cdot) = \mathsf{HE.Eval}(f, \cdot)$ on the ciphertexts $c_1, \ldots, c_k$, which gives $\hat{c}$. Then it computes the same function over the tags $\sigma_1, \ldots, \sigma_k$, which gives a value σ. Then it sends $\hat{c}$ and $\hat{\sigma}$ back to the client. The client at this point runs the verification algorithm of the MAC over the ciphertexts (it verifies running the function f over the labels $\vec{\tau} = \tau_1, \ldots, \tau_k$) and, if this is successful, it decrypts $\hat{c}$ using sk to get $f(m_1, \ldots, m_k)$.

In order to be more precise, we will now specify the protocols we are using in our $\mathcal{VC}$, which are ProbGen, Compute and Verify.

$\mathsf{ProbGen}_{\mathsf{vk}}(\mathsf{pk}, \vec{x})$:
  $\mathsf{id} \leftarrow\!\$\ \{0,1\}^{\lambda}$
  For $i = 1,\ldots,n$
    $\tau_i = \mathsf{id} \mid i$
    $c_i = \mathsf{Enc}(\mathsf{pk}, x_i)$
    $\hat{\sigma}_i = \mathsf{Auth}(\mathsf{vk}, \tau_i, c_i)$
  $\sigma_x = (c_1, \hat{\sigma_1}, \ldots, c_k, \hat{\sigma_k})$
  $\tau_x = \mathsf{id}$

$\mathsf{Compute}(\sigma_x)$:
  $c_y = \mathsf{HE.Eval}(\mathrm{f}, c_x)$
  $\hat{\sigma}_y = \mathsf{HMAC.Eval}(\mathrm{f'}, \sigma_x)$
  $\sigma_y = (c_y, \hat{\sigma}_y)$

$\mathsf{Verify}_{\mathsf{vk}}(\sigma_y, \tau_x)$:
$\mathcal{P} = \{f', \tau_1, \ldots, \tau_k\}$
$b \leftarrow \mathsf{HMAC.Ver}(\mathsf{vk}, \mathcal{P}, c_y, \hat{\sigma}_y)$
`Return` b

The protocol is shown in Figure 4.1.

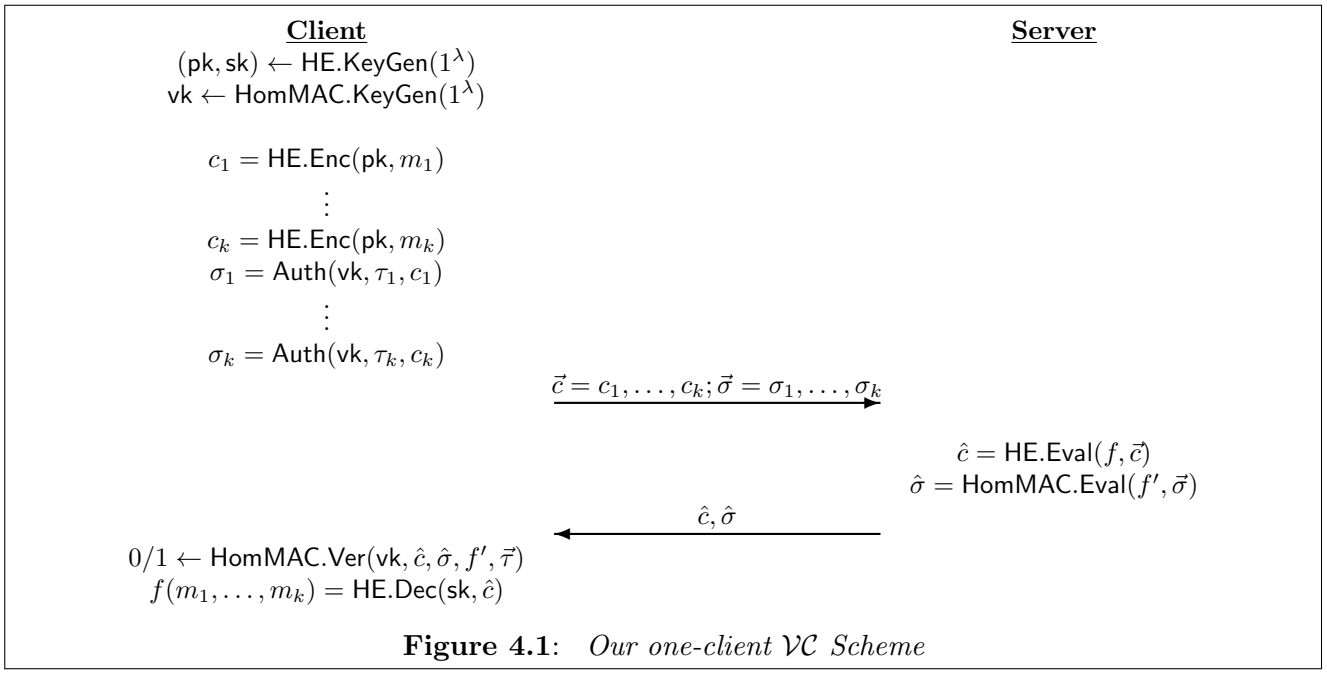


Figure 4.1: *Our one-client $\mathcal{VC}$ Scheme*

One can realise that this scheme vanishes the main idea for remote computation. In fact, the Client must compute the same function (but over the labels) to verify the correctness. This scheme seems to be useful in rare scenarios.

4.2 The two-client protocol

The following protocol now considers now two client machines, $\mathcal{C}_1$ and $\mathcal{C}_2$ that trust each other and share the secret keys vk and sk. We assume the same heavy computation f to be delegated to an external, untrusted server $\mathcal{S}$. For example these might be two machines belonging to the same organisation, where $\mathcal{C}_1$ simply sends data to the server and $\mathcal{C}_2$ receives and verifies the result. So now we have two interactions, one is between $\mathcal{C}_1$ and $\mathcal{S}$, denoted by $\langle \mathcal{C}_1 \leftrightarrows \mathcal{S} \rangle$, and the second is between $\mathcal{S}$ and $\mathcal{C}_2$, denoted by $\langle \mathcal{S} \leftrightarrows \mathcal{C}_2 \rangle$. The goal now is to focus on and reduce the communication of $\langle \mathcal{S} \leftrightarrows \mathcal{C}_2 \rangle$.The protocol is similar to the previous case and is shown in Figure 4.2.

Proposition 4.1 (Correctness)**.** *Our $\mathcal{VC}$ protocols defined in 4.1 and 4.2 are correct.*

Now that we know how is our generic protocol, we can define the correctness property for generic procedures. Informally, we know that this property refers that every correct computation should always output 1 in the Ver algorithm. In order to prove correctness, we recall the homomorphic properties of the scheme.

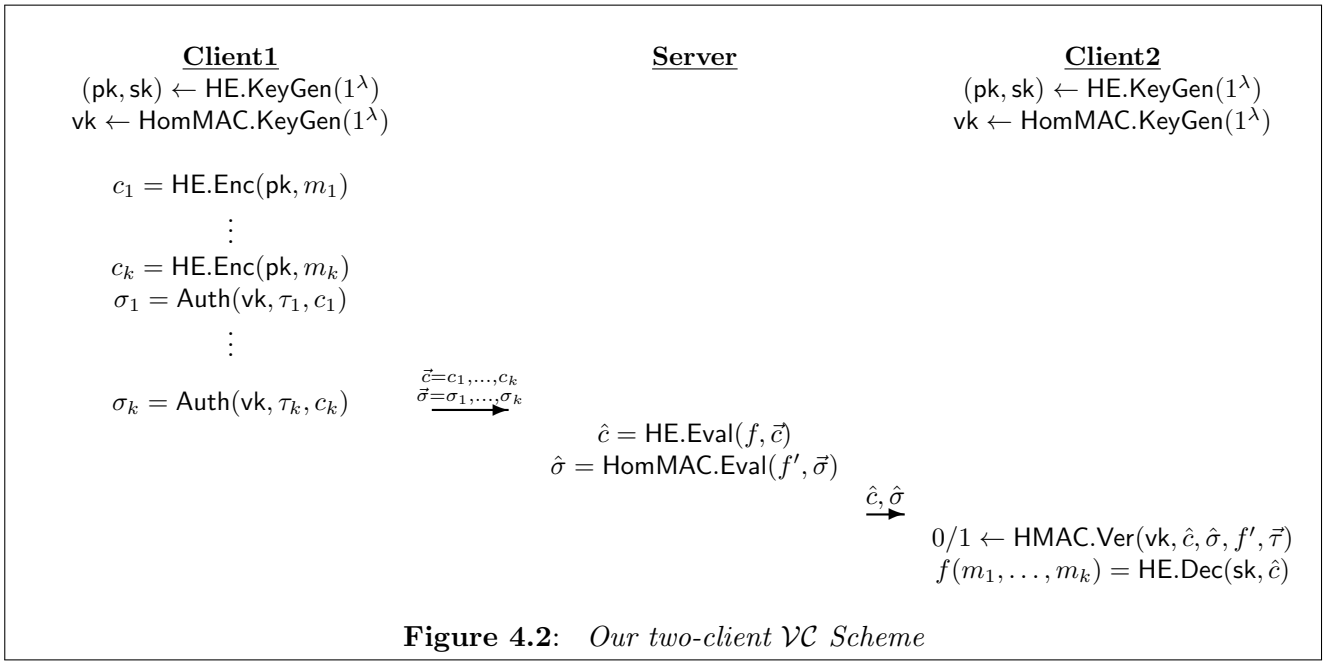


Figure 4.2: *Our two-client $\mathcal{VC}$ Scheme*

Proof. Basically, what ensures the client to decrypt c and be confident about the results is the HomMAC.Ver algorithm. The proof consists in the homomorphic properties of the HomMAC and the HE.Encryption. By definition, an HomMAC has authentication and evaluation correctness. Once the Client runs the HomMAC.Ver algorithm, it can be ensured about the match between the ciphertext $\hat{c}$ and its tag $\hat{\sigma}$. By correctness of the HE.Encryption, the client decrypts and gets the desired value $y = f(m_1, \ldots, m_k)$.

□

4.3 Security properties

The aforementioned schemes we discussed earlier should have all the typical security properties of a conventional $\mathcal{VC}$ scheme as defined in 2. Now that we have shown is correct, we will demonstrate the *Privacy*, and *Soundness* security properties for a $\mathcal{VC}$ scheme. The privacy property directly depend on the application of the homomorphic encryption, while the soundness property relies on the unforgeability of the HomMAC.

4.3.1 Privacy

Basically, the privacy of this $\mathcal{VC}$ Scheme relies on the privacy of the homomorphic encryption. We will prove this factor by reduction and proving that if there exist an efficient algorithm ϕ that breaks our $\mathcal{VC}$ scheme, then it should be an efficient algorithm $\tilde{\phi}$ that breaks the privacy of the homomorphic encryption, which is not possible on the assumption of the IND-CPA game.

Proof. Lets suppose there exists a polynomial run-time algorithm ϕ that breaks our $\mathcal{VC}$ scheme privacy. Then we introduce the following game where the adversary $\mathcal{A}$ has access to the algorithm ϕ as a black box so he/she can distinguish between two given outputs of the homomorphic encryption. In this security game we "simulate" the situation where the adversary becomes the challenger for the Privacy game for a generic $\mathcal{VC}$ and is still the adversary for the IND-CPA game.

The game is explained as follows. The challenger, who is the only one that knows the sk shares the pk with the adversary, as the IND-CPA game proceeds. The adversary also shares the public key pk and the

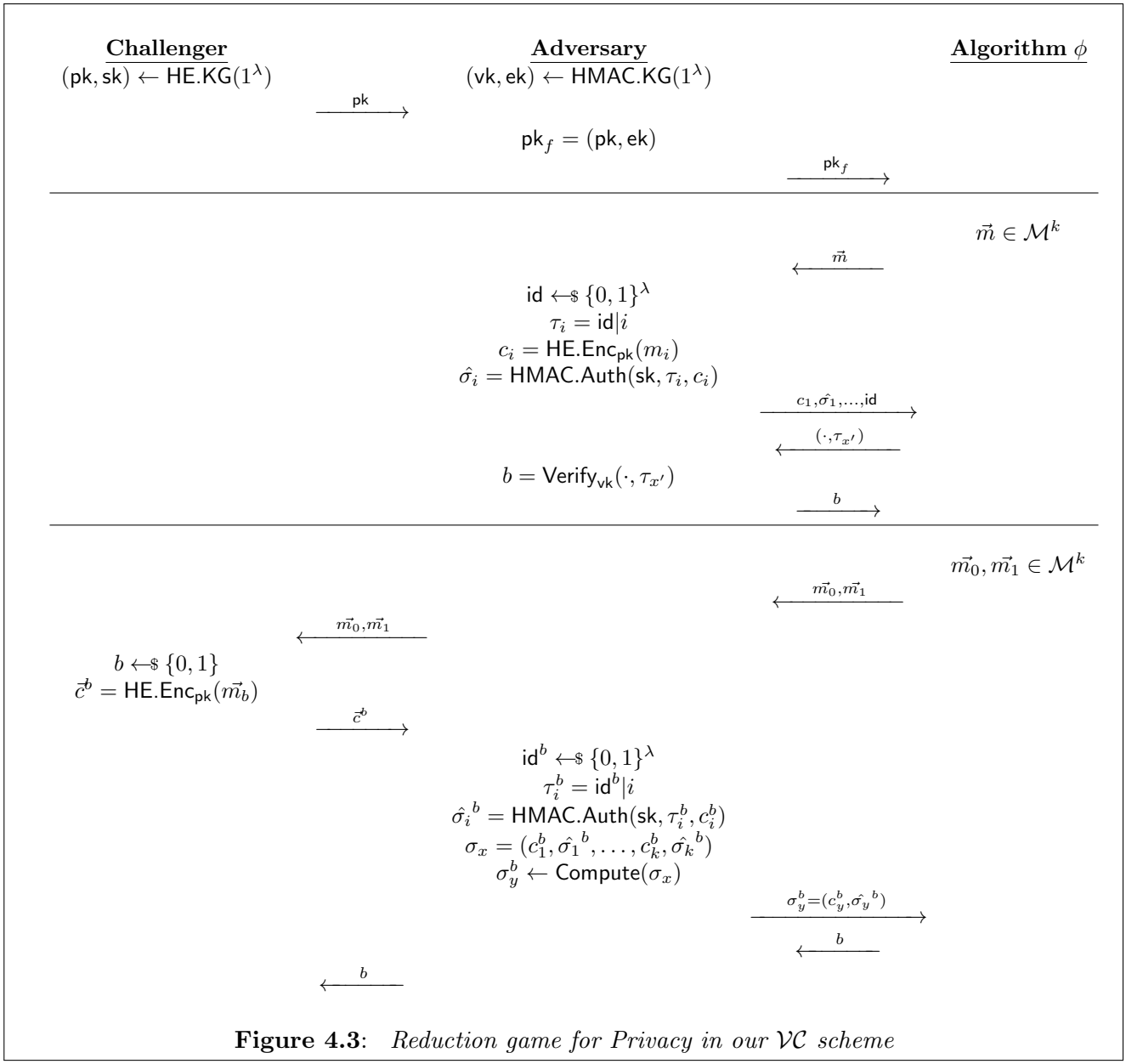


Figure 4.3: *Reduction game for Privacy in our $\mathcal{VC}$ scheme*

evaluation key ek. The algorithm ϕ is allowed to query as much as it wants problem generation outputs for honest inputs $\vec{m}$. This inputs (as it is indicated in our generic $\mathcal{VC}$ scheme is a vector of dimension k. The adversary, who knows vk, replies with ciphertexts labelled with a valid tag (but stores the labels τ_i). As it is specified, the algorithm can asks for a verify oracle for some tags $\tau_{x'}$. At some point, the algorithm ϕ sends the challenge $\vec{m_0}, \vec{m_1}$ to the adversary, who also sends it to the challenger. The challenger replies with the encryption of one of the sets, $\vec{c}^b$. The adversary then authenticates the set and computes (as it knows f) obtaining σ_y^b. The sends it to the algorithm (who can distinguish over them) and it replies with the bit b. This bit essentially is the one that proves that the algorithm can distinguish in our $\mathcal{VC}$ scheme and consequently, what proves that the adversary can break the IND-CPA game.

□

4.3.2 Soundness

Finally, we define the *soundness* property in our $\mathcal{VC}$ scheme. Informally, this property refers to the difficulty for a malicious party to obtain a valid proof using an invalid output. In our scheme means that ciphertexts that are false authenticated cannot pass the verification algorithm. This verification directly relies on the verification algorithm of the homomorphic MAC. Then, the soundness of our $\mathcal{VC}$ should depend on the unforgeability of the authenticator. We will prove this property by a reduction game. In the proof we refer to KG as KenGen for lack of space.

Proof. Lets suppose that exists an efficient algorithm ϕ that breaks the soundness of our $\mathcal{VC}$ scheme and then it will exist an efficient algorithm $\tilde{\phi}$ that breaks the unforgeability of the HomMAC. Then we introduce the following game where the adversary $\mathcal{A}$ has access to the algorithm ϕ as a black box so he/she can forge an input message m. We recall that breaking this unforgeability means winning the **HomUF-CMA** game, where the adversary has to reply with a valid tag for a message m^* as we specified in the second type for a *forgery*. Here, we suppose the labelled program to be *well-defined*.

Basically, in order to "simulate" the situation, the adversary has to be coherent with the challenger in the **HomUF-CMA** game and has to be coherent with the algorithm ϕ in the soundness game for a $\mathcal{VC}$.

We now that in the soundness game for a $\mathcal{VC}$, the algorithm ϕ, that plays the role of the adversary, is allowed to ask for the encoding (tag) of multiple problem instances. Following the rules of our $\mathcal{VC}$ scheme, these are vectors $\vec{c}$ of dimension k. The adversary then, acts as an oracle and asks it to the challenger after labelling the ciphertexts by its own. When the tag is received, the adversary replies to the algorithm ϕ with the same value.

As part of the game, the algorithm ϕ sends the challenge σ_y to verify the correctness of the pair $(c_y, \hat{\sigma_y})$. The adversary plays the same role as before and gets the bit from the Challenger.

At some point the algorithm ϕ breaks the unforgeability of our $\mathcal{VC}$ (because it can). Then, the adversary gets the necessary to break the **HomUF-CMA** game. The key for this situation is that the algorithm outputs σ_y^*, which does not match with the encryption of $f(x_j)$ (lets call it σ_y^j) for the labels associated with $\mathsf{id\_j}$. By correctness of homomorphic encryption this leads us to a *forgery*. □

4.4 Instantiating our VC scheme

Now that we have shown how the generic protocol works, we will apply to our schemes a specific HomMAC and a specific homomorphic encryption. In particular, we consider the $\mathsf{CF13}$ authenticator and BV encryption

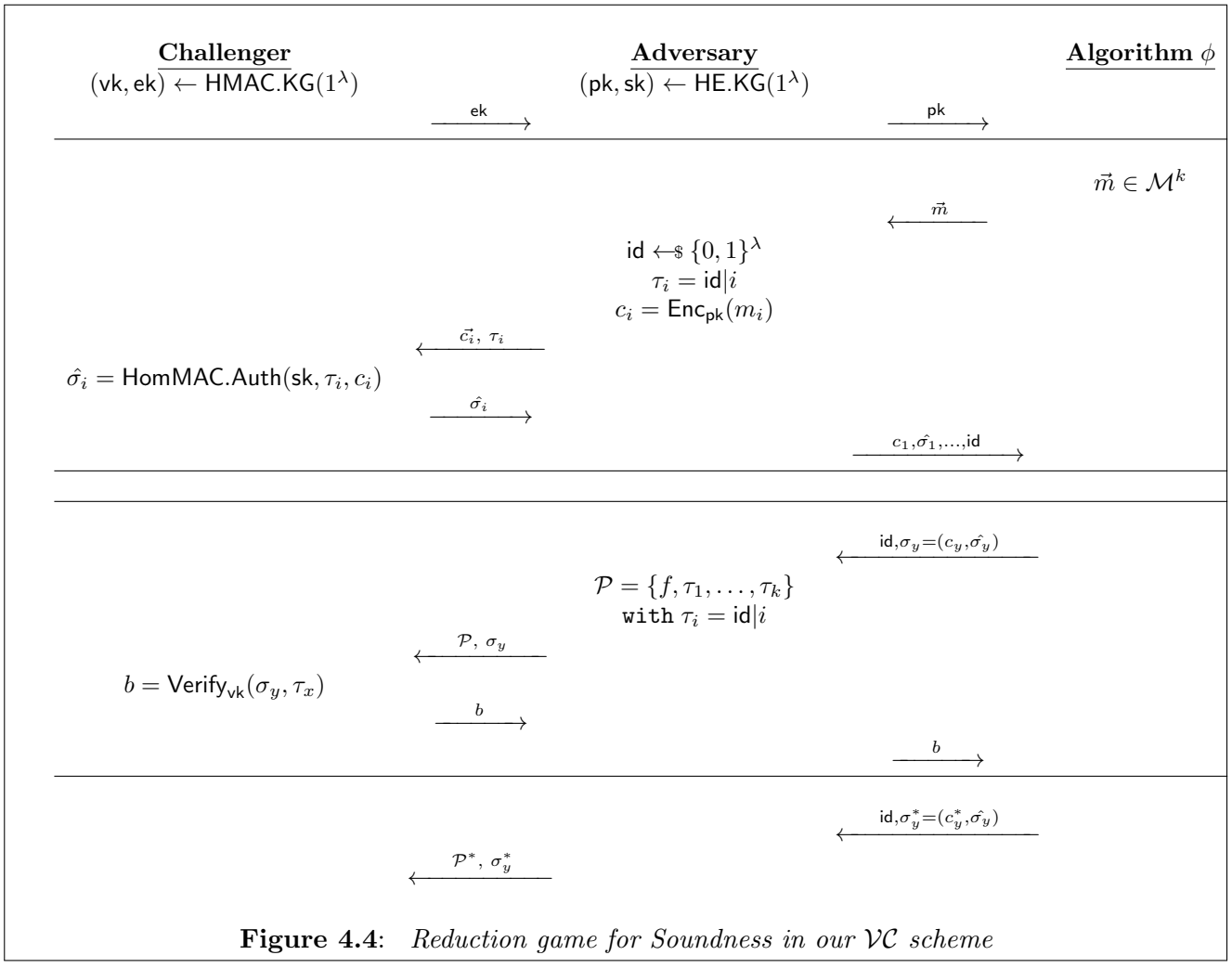


Figure 4.4: *Reduction game for Soundness in our $\mathcal{VC}$ scheme*

we defined in 3.1.4 and 3.2.5. With this two techniques in mind, we will check the correctness property for our protocols and then we will prove the security properties.

4.4.1 How to authenticate data

We remind that the version of the BV Encryption we are using is the one that is explained on the appendix of [Boi+21], as we showed in 3.2.5. Looking back on this section, the BV Encryption is the map:

$$\mathsf{BV.Enc}\ \ \mathcal{R}_t \to \mathcal{R}_q^2$$

To keep it more simple, we first think in $\mathcal{R}_q$ being q a prime number. We now recall that there exists two phases of communication in our protocol. The first one is when the Client 1 send the encrypted data $\{\{c_i\}_{i=1}^k : c_i = (c_0^{(i)}, c_1^{(i)}) \in \mathcal{R}_q^2\}$ and the authentication values $\{(\sigma_i)_{i=1}^k : \sigma_i = (\sigma_{i0}{}^{1}, \sigma_{i1}) \in \mathcal{R}_q^4\}$ where $\mathcal{R}_q := \mathbb{Z}_q[x]/(x^N + 1)$. The second phase of the communication happens when the server sends the computed values $\{c, \sigma\}$ to the Client 2. Our goal will be to estimate this second communication cost, so we should calculate where c and σ live.

As we have shown, for a fresh input value $m \in \mathcal{R}_t$, the $\mathsf{BV.Enc}$ outputs a fresh ciphertext $c \leftarrow \mathsf{BV.Enc}(m) \in \mathcal{R}_q^2$. Nevertheless, when computing a function f over the ciphertexts i.e. $c = f(c_1, \ldots, c_k)$, the length of the computed value c increases, depending directly on the degree d of the function f. One can prove that the computed value c always lives in $\mathbb{F}_q^{d\cdot(d+1)\cdot N}$.

As we explained in 3.1.4, the $\mathsf{CF13}$ authenticator is the map:

$$\mathsf{CF13}: \ \ \mathbb{Z}_q \to \mathbb{Z}_q^2$$

This HomMAC authenticates elements in $\mathbb{Z}_q \equiv \mathbb{F}_q$. Nevertheless, the elements that we want to be authenticated $\{\{c_i\}_{i=1}^k\}$ live in $\mathcal{R}_q^2$. One can interpret this $\mathsf{BV.Enc}$ output as a $2N$-size vector whose elements live in $\mathbb{F}_q$. The first strategy one can think for all the ciphertexts to be authenticated is authenticating each element (coefficient of the polynomial in $\mathcal{R}_q$) separately. Thus, we will no consider yet a reduction in ciphertext and authenticator polynomials $\{c, \sigma\}$ modulo $x^N + 1$ because we may loose the homomorphic correctness and so the verification procedure correctness.

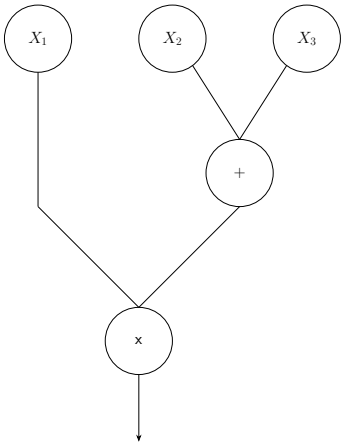
We recall that we authenticate every element separately and it is easy to check that the computed authenticator tag σ always lives in $\mathbb{F}_q^{d\cdot(d+1)^2\cdot N}$. It should be more simple if we show an example. For convenience, we will use the same function we used in 3.1.5.

Example

Consider the function $f(X_1, X_2, X_3) = X_1 \cdot (X_2 + X_3)$
$X_i \in \mathcal{R}_p^2$

Whose arithmetic circuit is represented as:

[1] This is abuse of notation. As the $\mathsf{CF13\ HomMAC}$ works, the size of the output tag is 2 times the size of the input. We interpret σ_{i0} as the authenticator of $c_0^{(i)}$ where $c_0^{(i)} \in \mathcal{R}_q$, so $\sigma_{i0} \in \mathcal{R}_q^2$.



First we set our secret and evaluation keys:

$\mathsf{KeyGen}(1^\lambda) \longrightarrow (\{K, x\}, p)$ being $x \in \mathbb{Z}_p$.

We want to authenticate $\hat{x} = f(x_1, x_2, x_3)$, so we first authenticate each element $X_i \in \mathcal{R}_p^2$ separately and also we authenticate every element $\{(x_i^{0,k}, x_i^{1,k})\}_{k=0}^{N-1}$, with $x_i^{j,k} \in \mathbb{Z}_p$ separately as in 3.1.5:

- $\mathsf{Auth}(\mathsf{sk}, \tau_1^{0,0}, x_1^{0,0}) \longrightarrow \sigma_{0,0,1} = (y_0^{(0,0,1)}, y_1^{(0,0,1)}) = (x_1^{0,0}, \frac{(r_{\tau_1^{0,0}} - x_1^{0,0})}{x} \bmod p) \longleftrightarrow \sigma_{0,0,1}(Z) = y_0^{(0,0,1)} + y_1^{(0,0,1)} Z$
- $\mathsf{Auth}(\mathsf{sk}, \tau_1^{1,0}, x_1^{1,0}) \longrightarrow \sigma_{1,1} = (y_0^{(1,0,1)}, y_1^{(1,0,1)}) = (x_1^{1,0}, \frac{(r_{\tau_1^{1,0}} - x_1^{1,0})}{x} \bmod p) \longleftrightarrow \sigma_{1,0,1}(Z) = y_0^{(1,0,1)} + y_1^{(1,0,1)} Z$

  ...

- $\mathsf{Auth}(\mathsf{sk}, \tau_3^{1,N-1}, x_3^{1,N-1}) \longrightarrow \sigma_{1,N-1,3} = (y_0^{(1,N-1,3)}, y_1^{(1,N-1,3)}) = (x_3^{1,N-1}, \frac{(r_{\tau_3^{1,N-1}} - x_3^{1,N-1})}{x} \bmod p) \longleftrightarrow \sigma_{1,N-1,3}(Z) = y_0^{(1,N-1,3)} + y_1^{(1,N-1,3)} Z$

One can notice that we have 3 input elements $X_i \in \mathcal{R}_p^2$ ($6N$ elements in $\mathbb{Z}_p$) what leads to $12N$ authenticator elements in $\mathbb{Z}_p$. Now, we run the Eval algorithm. Since this operation will be seen as polynomial multiplications, not only the size of the authenticator polynomial will grow as in 3.1.5, but also the ciphertext.

All in all, we will obtain the resulting ciphertext:

$$\hat{x} = x_1 \cdot^2 (x_2 + x_3) \in \mathcal{R}_p^{*3} \equiv \mathbb{F}_p^{3\cdot(2N)} \equiv \mathbb{F}_p^{(d+1)\cdot d \cdot N}$$

And the resulting authenticator polynomial σ such that:

$$\hat{\sigma} = \sigma_1 \cdot^3 (\sigma_2 + \sigma_3) \in \mathcal{R}_p^{*3^2} \equiv \mathbb{F}_p^{9\cdot(2N)} \equiv \mathbb{F}_p^{(d+1)^2 \cdot d \cdot N}$$

Here we notice that we cannot apply f to the ciphertexts and neither to the tags σ's. Instead, we apply a related function to run ciphertexts, $f'(\cdot) = \mathsf{HE.Eval}(f, \cdot)$ whose domain is different to f and allows us to compute ciphertexts. Also, the function that runs the tags $\mathsf{HomMAC.Eval}(f', \cdot)$, runs in another domain.

[2] Ring product without reducing modulo $x^N + 1$.
[3] Same Ring product without reducing modulo $x^N + 1$.

$$f' : \mathbb{F}_p^{2N \cdot k} \to \mathbb{F}_p^{(d+1) \cdot d \cdot N}$$

$$\mathsf{HomMAC.Eval}(f', \cdot) : \mathbb{F}_p^{4N \cdot k} \to \mathbb{F}_p^{(d+1)^2 \cdot d \cdot N}$$

In this case $k = 3$.

One can notice that this length will depend on the degree d of the function f. We recall that we are authenticating coefficient by coefficient and without reducing modulo $x^N + 1$ so not only the dimension of the ciphertext and authenticator will grow, also it will be implied the length.

Note that $\hat{\sigma} \subset \hat{x}$. This is completely natural since every element $a \in \mathbb{F}_p$ has to be authenticated for $(d+1)$ terms in the same space. Essentially one can interpret this $(d+1)$ elements as a polynomial $\hat{\sigma_a} \subset \hat{\sigma}$ of degree d (just like in 3.1.5) such that:

$$\begin{cases} \hat{\sigma_a}(0) = a \\ \hat{\sigma_a}(x) = f(r_{\tau_1^{j,k}}, r_{\tau_2^{j,k}}, r_{\tau_3^{j,k}}) \end{cases}$$

For some specific $j \in \{0, 1\}, k \in \{0, \dots N-1\}$ related to a and satisfying the nice structure of the Ver algorithm in the HomMAC.

4.4.2 Complexity

For the first communication phase we cannot reduce this cost since we should send fresh and original ciphertexts $\{\{c_i\}_{i=1}^k : c_i = (c_0^{(i)}, c_1^{(i)}) \in \mathcal{R}_q^2\}$ and labelled authenticator tags $\{(\sigma_i)_{i=1}^k : \sigma_i = (\sigma_{i0}, \sigma_{i1}) \in \mathcal{R}_q^4\}$. In computational terms:

$$\begin{aligned} \mathsf{k} \cdot (\#\text{bits of } \mathsf{c_i} &= |\mathsf{c_i}| &&= 2 \cdot \mathsf{N} \cdot (\log_2 \mathsf{q})) \\ \mathsf{k} \cdot (\#\text{bits of } \sigma_\mathsf{i} &= |\sigma_\mathsf{i}| &&= 4 \cdot \mathsf{N} \cdot (\log_2 \mathsf{q})) \end{aligned}$$

For the second phase, it is easy to see that the cost of the communication while sending c and σ will highly depend on the defined q prime number and the degree d of the function f. Analogously, in computational terms:

$$\begin{aligned} \#\text{bits of } \mathsf{c} &= |\mathsf{c}| &&= \mathsf{d} \cdot (\mathsf{d}+1) \cdot \mathsf{N} \cdot (\log_2 \mathsf{q}) \\ \#\text{bits of } \sigma &= |\sigma| &&= \mathsf{d} \cdot (\mathsf{d}+1)^2 \cdot \mathsf{N} \cdot (\log_2 \mathsf{q}) \end{aligned}$$

Chapter 5

Our improved VC scheme

In the last chapter we explained how our protocol works and we gave an example of how to authenticate ciphertexts. In this chapter we will modify the CF13 authenticator procedure into a new one that allows us to authenticate ciphertexts belonging to $\mathcal{R}_q$.

Once we know how to authenticate ciphertexts, we will give new precise estimation for the size of the authenticator polynomial σ by using the new authenticator procedure $\widetilde{\mathsf{CF13}}$.

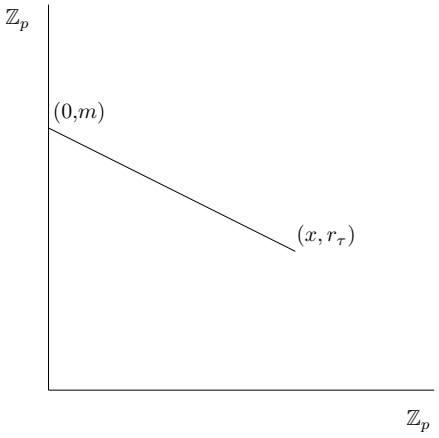
5.1 Our variation of the CF13 HomMAC

As we introduced, there exists a way to reduce the communication cost in our scheme. This idea relies in reducing the size of the authenticator computed value σ. As we have shown in the example 4.4.1, it is natural to think in reducing applying the standard ring product (with a reduction modulo $x^N + 1$) to our both authenticator and ciphertext polynomials. In order to achieve it, we consider a new authenticating procedure (we call it the $\widetilde{\mathsf{CF13}}$) that authenticates inputs $m \in \mathcal{R}_q$ and it would be the map:

$$\widetilde{\mathsf{CF13}}: \ \mathcal{R}_q \to \mathcal{R}_q^2$$

The procedure for this new HomMAC should be not too different from the original, the main variance is that we now have input elements that belong to another space.

In order to interpret correctly the CF13 procedure, one can think in authenticating an element $m \in \mathbb{Z}_p$ as the unique lineal polynomial that interpolates the points $(0, m)$ and (x, r_τ) in the Cartesian's space $\mathbb{Z}_p \times \mathbb{Z}_p$. We recall that x is a secret point, so the slope of this line and r_τ are unknown.



Now, for the $\widetilde{\mathsf{CF13}}$, we have to act in the same way and interpret it as the lineal interpolator polynomial in the Cartesian's space $\mathcal{R}_p \times \mathcal{R}_p$. As one can notice, $x \in \mathcal{R}_p$ must belong to the invertible subset of $\mathcal{R}_p$ due to the Auth procedure *i.e.* $x \in \mathbb{A} \subset \mathcal{R}_p$ and $\mathbb{A} = \{y : \exists y^{-1}\}$. This subset $\mathbb{A}$ trivially exists since $\mathbb{Z}_p \subset \mathcal{R}_p$ as $\mathbb{Z}_p$ can be seen as a zero-degree polynomial in $\mathcal{R}_p$. The authenticating procedure is analogous to 3.1.4.

One can see in the example 4.4.1, that there is a d factor we can omit in both $\hat{x}$ and $\hat{\sigma}$. Also, we authenticate elements in such a different way, so the resulting values for both ciphertext and authenticator are as follows:

(i) $\hat{x} = x_1 \cdot (x_2 + x_3) \in \mathcal{R}_p^3 \equiv \mathbb{F}_p^{3 \cdot N} \equiv \mathbb{F}_p^{(d+1) \cdot N}$

(ii) $\hat{\sigma} = \sigma_1 \cdot (\sigma_2 + \sigma_3) \in \mathcal{R}_p^{3^2} \equiv \mathbb{F}_p^{9 \cdot N} \equiv \mathbb{F}_p^{(d+1)^2 \cdot N}$

5.1.1 Complexity

The intention in considering the $\widetilde{\mathsf{CF13}}$ was reducing the second phase in the communication cost. Even using this new HomMAC, the communication between Client 1 and Server remains intact. Looking a bit above, the reader can guess what is going to be the complexity in sending c and σ based on the previous complexity. We can remove d terms in both:

$$\begin{aligned} \#\text{bits of } \mathsf{c} &= |\mathsf{c}| = (\mathsf{d}+1) \cdot \mathsf{n} \cdot (\log_2 \mathsf{q}) \\ \#\text{bits of } \sigma &= |\sigma| = (\mathsf{d}+1)^2 \cdot \mathsf{n} \cdot (\log_2 \mathsf{q}) \end{aligned}$$

Taking a brief look at this parameters, one can realise that low degree functions (i.e. $d <<$) like, for example statistics can be benefited from this scheme. We provide some examples of these functions in the appendix.

5.2 Running-time cost

Apart from the complexity of sending the ciphertext c and the authenticator σ, there exists computational time in running algorithms like Ver. We first have to distinguish on the algorithms that the Server/Clients run.

Based in Figure 4.2, Server runs only two $\mathsf{HE.Eval}$ algorithms that depends on the degree d of the encrypted version of the function f (which directly relies on f). Now, the running time for the clients ($\mathsf{T_{Cl}}$):

$$\mathsf{T_{Cl}} = \mathsf{T_{Enc}} + \mathsf{T_{Auth}} + \mathsf{T_{Ver}} + \mathsf{T_{Dec}}$$

Lets estimate them separately.

- For the Enc algorithm running-time ($\mathsf{T_{Enc}}$):
  1. KeyGen needs 3 Ring samples: $\{s, a_0, e_0\}$.
  2. Run k times the Enc procedure. For every encryption procedure, there is needed 3 Ring samples, 2 Ring products, 3 Ring additions and 2 standard products.
- For the Auth algorithm running-time ($\mathsf{T_{Auth}}$):
  1. KeyGen needs 2 samples: $\{K, x\}$ where $x \in \mathcal{R}_q$ and $K \leftarrow \mathcal{K}$.
  2. Running k times the Auth procedure needs k PRF evaluations and k Ring multiplications (multiplying by x^{-1}).
- For the Ver algorithm running-time ($\mathsf{T_{Ver}}$):

1. Time in calculating $\rho = f(r_{\tau_1}, \ldots, r_{\tau_k})$, which is a function evaluation.
2. Time in evaluating x over the HomMAC output polynomial σ that the server sends and check whether $\rho = \sigma(x)$. This is a polynomial evaluation over rings.

- For the Dec algorithm running-time ($\mathsf{T}_{\mathsf{Dec}}$):

1. Here we just need a polynomial evaluation $c(s) = \sum_{i=0}^{D-1} c_i s^i$ (this is also a polynomial evaluation over rings) and a reduction modulo t.

We will consider sampling timeless. Then we obtain:

$$\mathsf{T}_{\mathsf{CI}} = (k \cdot (3\mathsf{T}_{\mathsf{Ring\_Add}} + 2\mathsf{T}_{\mathsf{Ring\_Mult}} + 2\mathsf{T}_{\mathsf{Std\_Mult}})) + (k \cdot (\mathsf{T}_{\mathsf{PRF}} + \mathsf{T}_{\mathsf{Ring\_Mult}})) + (\mathsf{T}^k_{\mathsf{f\_eval}} + \mathsf{T}_{\mathsf{Ring\_poly\_eval}}) +$$
$$+ (\mathsf{T}_{\mathsf{Ring\_poly\_eval}} + \mathsf{T}_{\mathsf{Mod\_reduct}})$$

Which is equal to:

$$\mathsf{T}_{\mathsf{CI}} = 3k \cdot \mathsf{T}_{\mathsf{Ring\_Add}} + 3k \cdot \mathsf{T}_{\mathsf{Ring\_Mult}} + 2k \cdot \mathsf{T}_{\mathsf{Std\_Mult}} + k \cdot \mathsf{T}_{\mathsf{PRF}} + \mathsf{T}^k_{\mathsf{f\_eval}} + 2 \cdot \mathsf{T}_{\mathsf{Ring\_poly\_eval}} + \mathsf{T}_{\mathsf{Mod\_reduct}}$$

5.3 Unforgeability of $\widetilde{\mathsf{CF13}}$

In [CF13], the authors prove the unforgeability of the HomMAC having its image in $\mathbb{Z}_p^2$. Since $\widetilde{\mathsf{CF13}}$ image is now in $\mathcal{R}_q$ being q prime, the proof in essence, changes. If one went to this paper, one could notice that there is given a proof for the unforgeability of the original HomMAC that relies on the security of the PRFs functions. The proof also relies on a given proposition based on an application of the Schwartz–Zippel Lemma for fields [Sch80] [Zip79]. Since fiels are not our case in $\widetilde{\mathsf{CF13}}$, we now introduce the generalised version for this Lemma [Bis+18].

Lemma 5.1 (Generalised Schwartz-Zippel Lemma)**.** *Let $\mathcal{R}$ be a finite commutative ring with identity and $f : \mathcal{R}^n \to \mathcal{R}$ be an n-variate non-zero polynomial. Let $\mathbb{A} \subseteq \mathcal{R}$ be a finite exceptional set. Let deg(f) denote the total degree of f. Then*

$$\Pr_{\vec{a} \leftarrow \mathbb{A}^n}[f(\vec{a}) = 0] \leq \frac{deg(f)}{|\mathbb{A}|}$$

The proof is on the appendix. Referring again to the Catalano and Fiore's manuscript, one can adapt the same unforgeability proof they had but in the ring case. The proof is analogous because the only field property they used (apart from the Schwartz-Zippel Lemma application) in the unforgeability proof was the existence of the inverse of the secret key x. The renewed procedure must now consider the secret key $\mathsf{sk} = (K, x)$ with x being an element of a finite exceptional set $\mathbb{A} \subseteq \mathcal{R}$.

5.4 Capturing variants of the BV encryption

Since the publication of the BV Encryption, some versions of this *somewhat homomorphic encryption* have been created. For the privacy property in our $\mathcal{VC}$ scheme, we stress this encryption scheme to have *semantic security*. One parameter that can be related to this security in Ring-LWE are q and n, where this parameters define $\mathcal{R}_q$. If one think in q being a small prime number, then the security for this scheme is less evident than considering q large. Moreover, one even can think in q to be not prime, i.e. $q = \prod_{i=1}^{k} p_i$.

Based on the Chinese Reminder Theorem (CRT), one can interpret the output space of the BV Encryption ($\mathcal{R}_q^2$) as:

$$\mathcal{R}_q^2 = \mathcal{R}_{p_1}^2 \times \cdots \times \mathcal{R}_{p_k}^2$$

This consideration does not leak any information from the secret key sk but it makes decryption easier since the Dec algorithm relies on evaluating the secret key on ciphertexts and reduce modulo t. This operation can be done separately in each $\mathcal{R}_{p_i}^2$, so one can be benefited from parallel computing.

Chapter 6

Conclusions and Future work

In this thesis, we have introduced and explained what is to verify computations on encrypted data, with the aim of maintaining privacy and guaranteeing the integrity of the results. In this work we give a generic $\mathcal{VC}$ scheme that allows to ensure this correctness for external computations and we have shown its weaknesses, which is the main one is the efficiency.

Efficiency (or in more technical terms, *succinctness*) is an important quality in cryptography and computer science. One of the objectives of this thesis was reducing the communication complexity in order to improve efficiency. In this work, we give precise estimations about this complexity, although results can be improved.

There are well known techniques for reducing the communication and give a proof for correctness. One of these techniques consist in apply an homomorphic hash (seen in [Boi+21] for example) and create a commitment to the ciphertexts [FT22]. In brief, an efficient verifiable computation is a very interesting and a *state of the art* topic that can open new lines of research.

Bibliography

[AB09] Shweta Agrawal and Dan Boneh. "Homomorphic MACs: MAC-based integrity for network coding". In: *Applied Cryptography and Network Security: 7th International Conference, ACNS 2009, Paris-Rocquencourt, France, June 2-5, 2009. Proceedings 7*. Springer. 2009, pp. 292–305.

[Bal21] David Balbás. "The Hardness of LWE and Ring-LWE: A Survey". In: *Cryptology ePrint Archive* (2021).

[Bis+18] Anurag Bishnoi et al. "On zeros of a polynomial in a finite grid". In: *Combinatorics, Probability and Computing* 27.3 (2018), pp. 310–333.

[Bla00] John Richard Black Jr. *Message authentication codes*. University of California, Davis, 2000.

[Boi+21] Alexandre Bois et al. "Flexible and efficient verifiable computation on encrypted data". In: *Public-Key Cryptography–PKC 2021: 24th IACR International Conference on Practice and Theory of Public Key Cryptography, Virtual Event, May 10–13, 2021, Proceedings, Part II*. Springer. 2021, pp. 528–558.

[Bra19] Zvika Brakerski. "Fundamentals of fully homomorphic encryption". In: *Providing sound foundations for cryptography: on the work of Shafi Goldwasser and Silvio Micali*. 2019, pp. 543–563.

[BV11] Zvika Brakerski and Vinod Vaikuntanathan. "Fully homomorphic encryption from ring-LWE and security for key dependent messages". In: *Advances in Cryptology–CRYPTO 2011: 31st Annual Cryptology Conference, Santa Barbara, CA, USA, August 14-18, 2011. Proceedings 31*. Springer. 2011, pp. 505–524.

[BV14] Zvika Brakerski and Vinod Vaikuntanathan. "Efficient fully homomorphic encryption from (standard) LWE". In: *SIAM Journal on computing* 43.2 (2014), pp. 831–871.

[Cat+14] Dario Catalano et al. "Generalizing homomorphic MACs for arithmetic circuits". In: *Public-Key Cryptography–PKC 2014: 17th International Conference on Practice and Theory in Public-Key Cryptography, Buenos Aires, Argentina, March 26-28, 2014. Proceedings 17*. Springer. 2014, pp. 538–555.

[CF13] Dario Catalano and Dario Fiore. "Practical homomorphic MACs for arithmetic circuits". In: *Advances in Cryptology–EUROCRYPT 2013: 32nd Annual International Conference on the Theory and Applications of Cryptographic Techniques, Athens, Greece, May 26-30, 2013. Proceedings 32*. Springer. 2013, pp. 336–352.

[CT15] Massimo Chenal and Qiang Tang. "On key recovery attacks against existing somewhat homomorphic encryption schemes". In: *Progress in Cryptology-LATINCRYPT 2014: Third International Conference on Cryptology and Information Security in Latin America Florianópolis, Brazil, September 17–19, 2014 Revised Selected Papers 3*. Springer. 2015, pp. 239–258.

[ElG85] Taher ElGamal. "A public key cryptosystem and a signature scheme based on discrete logarithms". In: *IEEE transactions on information theory* 31.4 (1985), pp. 469–472.

[Eli+15] Yara Elias et al. "Provably weak instances of Ring-LWE". In: *Advances in Cryptology–CRYPTO 2015: 35th Annual Cryptology Conference, Santa Barbara, CA, USA, August 16-20, 2015, Proceedings, Part I 35*. Springer. 2015, pp. 63–92.

[FGP14] Dario Fiore, Rosario Gennaro, and Valerio Pastro. "Efficiently verifiable computation on encrypted data". In: *Proceedings of the 2014 ACM SIGSAC Conference on Computer and Communications Security*. 2014, pp. 844–855.

[Fio22] Dario Fiore. "Verifiable Computation and Succinct Arguments for NP". In: *Asymmetric Cryptography: Primitives and Protocols* (2022), pp. 257–281.

[FNP20] Dario Fiore, Anca Nitulescu, and David Pointcheval. "Boosting verifiable computation on encrypted data". In: *Public-Key Cryptography–PKC 2020: 23rd IACR International Conference on Practice and Theory of Public-Key Cryptography, Edinburgh, UK, May 4–7, 2020, Proceedings, Part II 23*. Springer. 2020, pp. 124–154.

[FT22] Dario Fiore and Ida Tucker. "Efficient Zero-Knowledge Proofs on Signed Data with Applications to Verifiable Computation on Data Streams". In: *Proceedings of the 2022 ACM SIGSAC Conference on Computer and Communications Security*. 2022, pp. 1067–1080.

[Gen09] Craig Gentry. *A fully homomorphic encryption scheme*. Stanford university, 2009.

[GGM86] Oded Goldreich, Shafi Goldwasser, and Silvio Micali. "How to construct random functions". In: *Journal of the ACM (JACM)* 33.4 (1986), pp. 792–807.

[GGP10] Rosario Gennaro, Craig Gentry, and Bryan Parno. "Non-interactive verifiable computing: Outsourcing computation to untrusted workers". In: *Advances in Cryptology–CRYPTO 2010: 30th Annual Cryptology Conference, Santa Barbara, CA, USA, August 15-19, 2010. Proceedings 30*. Springer. 2010, pp. 465–482.

[GLP12] Tim Güneysu, Vadim Lyubashevsky, and Thomas Pöppelmann. "Practical lattice-based cryptography: A signature scheme for embedded systems". In: *Cryptographic Hardware and Embedded Systems–CHES 2012: 14th International Workshop, Leuven, Belgium, September 9-12, 2012. Proceedings 14*. Springer. 2012, pp. 530–547.

[GNS21] Chaya Ganesh, Anca Nitulescu, and Eduardo Soria-Vazquez. "Rinocchio: SNARKs for ring arithmetic". In: *Cryptology ePrint Archive* (2021).

[Gol+13] Shafi Goldwasser et al. "How to run turing machines on encrypted data". In: *Advances in Cryptology–CRYPTO 2013: 33rd Annual Cryptology Conference, Santa Barbara, CA, USA, August 18-22, 2013. Proceedings, Part II*. Springer. 2013, pp. 536–553.

[GW13] Rosario Gennaro and Daniel Wichs. "Fully homomorphic message authenticators". In: *Advances in Cryptology-ASIACRYPT 2013: 19th International Conference on the Theory and Application of Cryptology and Information Security, Bengaluru, India, December 1-5, 2013, Proceedings, Part II 19*. Springer. 2013, pp. 301–320.

[Hil29] Lester S Hill. "Cryptography in an algebraic alphabet". In: *The American Mathematical Monthly* 36.6 (1929), pp. 306–312.

[JMV01] Don Johnson, Alfred Menezes, and Scott Vanstone. "The elliptic curve digital signature algorithm (ECDSA)". In: *International journal of information security* 1 (2001), pp. 36–63.

[KL20] Jonathan Katz and Yehuda Lindell. *Introduction to modern cryptography*. CRC press, 2020.

[LPR13] Vadim Lyubashevsky, Chris Peikert, and Oded Regev. "On ideal lattices and learning with errors over rings". In: *Journal of the ACM (JACM)* 60.6 (2013), pp. 1–35.

[Pei09] Chris Peikert. "Public-key cryptosystems from the worst-case shortest vector problem". In: *Proceedings of the forty-first annual ACM symposium on Theory of computing*. 2009, pp. 333–342.

[RAD+78] Ronald L Rivest, Len Adleman, Michael L Dertouzos, et al. "On data banks and privacy homomorphisms". In: *Foundations of secure computation* 4.11 (1978), pp. 169–180.

[Reg09] Oded Regev. "On lattices, learning with errors, random linear codes, and cryptography". In: *Journal of the ACM (JACM)* 56.6 (2009), pp. 1–40.

[Reg10] Oded Regev. "The learning with errors problem". In: *Invited survey in CCC* 7.30 (2010), p. 11.

[RSA78] Ronald L Rivest, Adi Shamir, and Leonard Adleman. "A method for obtaining digital signatures and public-key cryptosystems". In: *Communications of the ACM* 21.2 (1978), pp. 120–126.

[Sch80] Jacob T Schwartz. "Fast probabilistic algorithms for verification of polynomial identities". In: *Journal of the ACM (JACM)* 27.4 (1980), pp. 701–717.

[Sch90] Claus-Peter Schnorr. "Efficient identification and signatures for smart cards". In: *Advances in Cryptology—CRYPTO'89 Proceedings 9*. Springer. 1990, pp. 239–252.

[Sma16] P Nigel Smart. *Cryptography made simple*. Springer, 2016.

[Zip79] Richard Zippel. "Probabilistic algorithms for sparse polynomials". In: *Symbolic and Algebraic Computation: EUROSM'79, An International Symposium on Symbolic and Algebraic Manipulation, Marseille, France, June 1979 2*. Springer. 1979, pp. 216–226.

Appendix A

Appendix

A.1 Schwartz–Zippel Lemma

Lemma A.1 (Generalised Schwartz-Zippel Lemma)**.** *Let $\mathcal{R}$ be a finite commutative ring with identity and $f : \mathcal{R}^n \to \mathcal{R}$ be an n-variate non-zero polynomial. Let $\mathbb{A} \subseteq \mathcal{R}$ be a finite exceptional set. Let $deg(f)$ denote the total degree of f. Then*

$$\Pr_{\vec{a} \leftarrow \mathbb{A}^n}[f(\vec{a}) = 0] \leq \frac{deg(f)}{|\mathbb{A}|}$$

The proof is identical as the one showed in [GNS21]:

Proof. We prove by induction on the number of variables n. For $n = 1$, let $a_1 \in \mathbb{A}$ be a root of $f(x)$. As $(x - a_1)$ is a monic polynomial, we have that $f(x) = (x - a_1)f_1(x)$, where the $deg(f_1) < deg(f)$. Any other root $a_2 \in \mathbb{A}$ has to be a root of $f_1(x)$, as $(a_2 - a_1) \in \mathcal{R}^*$ and $f(a_2) = 0$. Hence, we have that $f(x) = (x - a_1)(x - a_2)f_2(x)$, where the $deg(f_2) < deg(f_1)$. By iterating this argument, we conclude that $f(x)$ cannot have more roots in $\mathbb{A}$ than $deg(f)$ and hence $\Pr_{a \leftarrow \mathbb{A}}[f(a) = 0] \leq \frac{deg(f)}{|\mathbb{A}|}$.

Assume now the result holds for $(n - 1)$-variate polynomials. Given any n-variate polynomial $f(\vec{x}) \in \mathcal{R}[x_1, \ldots, x_n]$, denote by $k = deg_{x_n}(f)$ the largest power of x_n appearing in any monomial of f. Then we have that:

$$f(\vec{x}) = \sum_{l=1}^{k} x_n^l \cdot g_l(x_1, \ldots, x_{n-1})$$

Denote by $\mathcal{E}_1$ the event $g_k(\vec{a}) = 0$. By definition of k, we know that $g_k(x_1, \ldots, x_{n-1})$ is a non-zero polynomial, so by induction hypothesis $\Pr_{\vec{a} \leftarrow \mathbb{A}^{n-1}}[\mathcal{E}_1] \leq \frac{(deg(f)-k)}{|\mathbb{A}|}$. Assuming $\neg\mathcal{E}_1$ and by applying the same reasoning as for $n = 1$, we have that $f(\vec{a}) \in \mathcal{R}[x_n]$ has at most k roots in $\mathbb{A}$, so $\Pr_{\vec{a} \leftarrow \mathbb{A}}[f(\vec{a}) = 0 | \neg\mathcal{E}_1] \leq \frac{k}{|\mathbb{A}|}$. We finalise by noting that (where the probability is taking over the choice of $\vec{a} \leftarrow \mathbb{A}^n$):

$$\Pr[f(\vec{a}) = 0] = \Pr[f(\vec{a}) = 0 | \neg\mathcal{E}_1] \cdot \Pr[\neg\mathcal{E}] + \Pr[f(\vec{a}) = 0 | \mathcal{E}_1] \cdot \Pr[\mathcal{E}]$$

$$\leq \Pr[f(\vec{a}) = 0 | \neg\mathcal{E}_1] \cdot \Pr[\neg\mathcal{E}] \leq \frac{deg(f) - k}{|\mathbb{A}|} + \frac{k}{|\mathbb{A}|}$$

□

A.2 Function applications

The possible applications for this project go through considering low-degree functions. For vectors $\vec{x} = (x_1, \ldots, x_t)$ and $\vec{y} = (y_1, \ldots, y_t)$ we define the following basic functions:

A.2.1 Statistics

Average

$$\text{avg}(\vec{x}) = \sum_{i=1}^{t} x_i \ /t$$

Variance

$$\text{var}(\vec{x}) = \sum_{j=1}^{t} (x_j - \text{avg}(\vec{x}))^2 \ /t$$

Covariance

$$\text{cov}(\vec{x}, \vec{y}) = \sum_{j=1}^{t} (x_j - \text{avg}(\vec{x}))(y_j - \text{avg}(\vec{y})) \ /t$$

Linear regression

Given two sets of observations as two vectors ($\vec{x}$, $\vec{y}$), the linear regression of $\vec{y}$ as a function of $\vec{x}$ is defined by two coefficients $\hat{\alpha}, \hat{\beta}$ such that:

$$\hat{\beta} = \frac{\text{cov}(\vec{x}, \vec{y})}{\text{var}(\vec{x})}$$

$$\hat{\alpha} = \text{avg}(\vec{y}) - \hat{\beta} \cdot \text{avg}(\vec{x})$$

Notice that if one is working over finite algebraic structures, computing functions like the average can be not possible since, sometimes, decimals cannot be expressed. Then, for functions like average, where a division is needed, one ban do either a preprocessing or just the sum function over $\vec{x}$ since both average and sum functions are equivalent.

A.2.2 Geometry

Euclidean distance

Since we are considering arithmetic circuits, we will not compute non linear functions like the square root. Furthermore, one can see this function equivalent to the Squared Euclidean distance, which is:

$$||\vec{x} - \vec{y}||_2^2 = \sum_{i=1}^{t} (x_i - y_i)^2$$

A.3 Scenarios

We are about to imagine some scenarios where a user wants to delegate the computation of a function to an external and untrusted server. This scenarios may require the data to be encrypted, for many reasons such as identity security or even in a legal manner (medical data). Every scenario will need the data not only encrypted but authenticated.

A.3.1 Scenario 1

Here we consider the Client 1 to be some medical institution, like an hospital and the Client 2 to be some government institution, which requires some statistics about population health. In this case, sharing medical data in not allowed.

The protocol works as described in 4.2:

The medical institution encrypts the data with $\mathsf{BV.Enc}$ and authenticates it using $\widetilde{\mathsf{CF13}}$. Then it sends $c_1, \ldots, c_k$ and $\tau_1, \ldots, \tau_k$ to the server. The server computes and sends the results to the government institution, that knows vk, f and also K, which defines the PRF function.

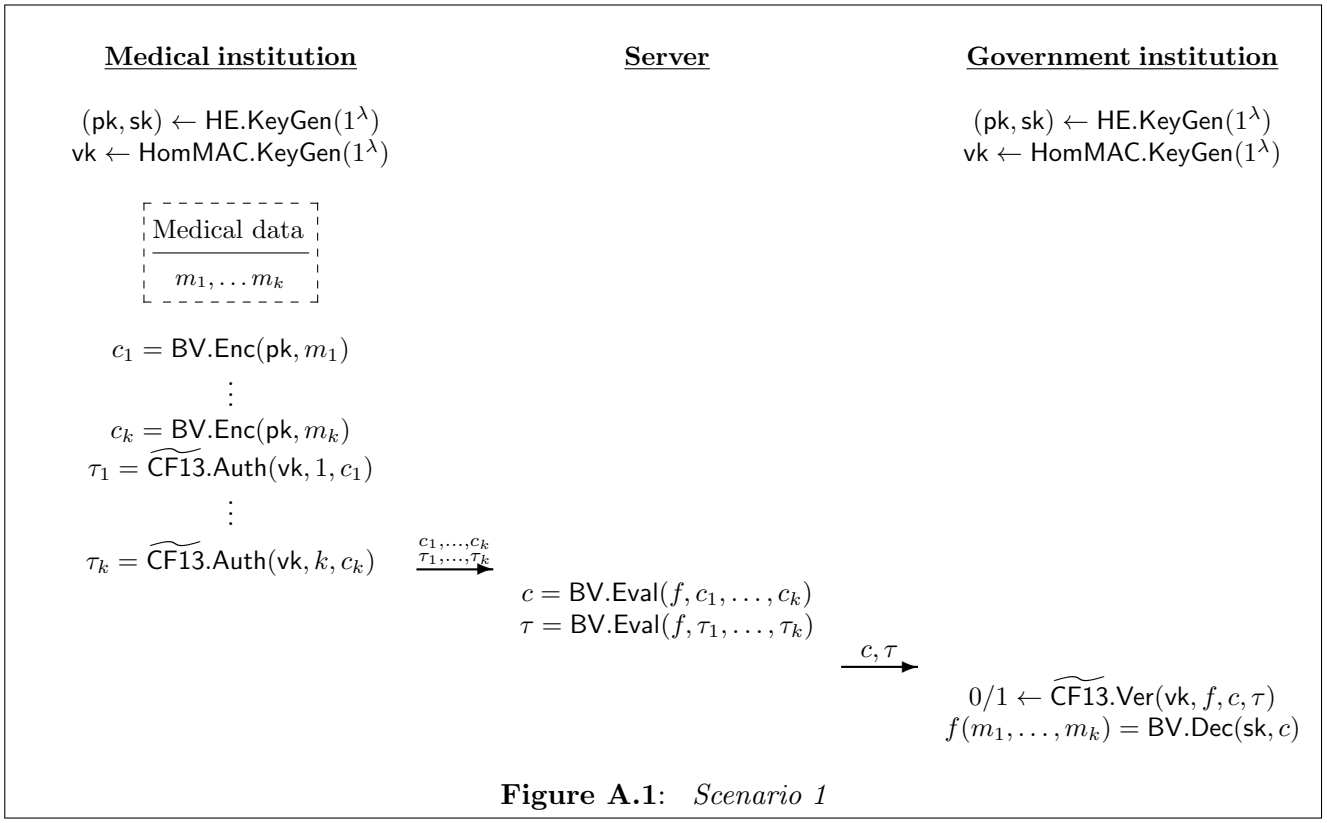


Figure A.1: *Scenario 1*

A.3.2 Scenario 2

Here we consider a one client scenario, where the client can be some government institution that receives encrypted data. This data, for example, can be some statistics about population which should be not shared. In this scenario, the client has limited resources an it cannot decrypt all the data and compute it by itself. Therefore, the client authenticates the data with the $\widetilde{\mathsf{CF13}}$ procedure, which is very efficient, and when the results are given, it only has to run the Ver algorithm and just decrypt the results. This should be more efficient than decrypting all data and computing it.

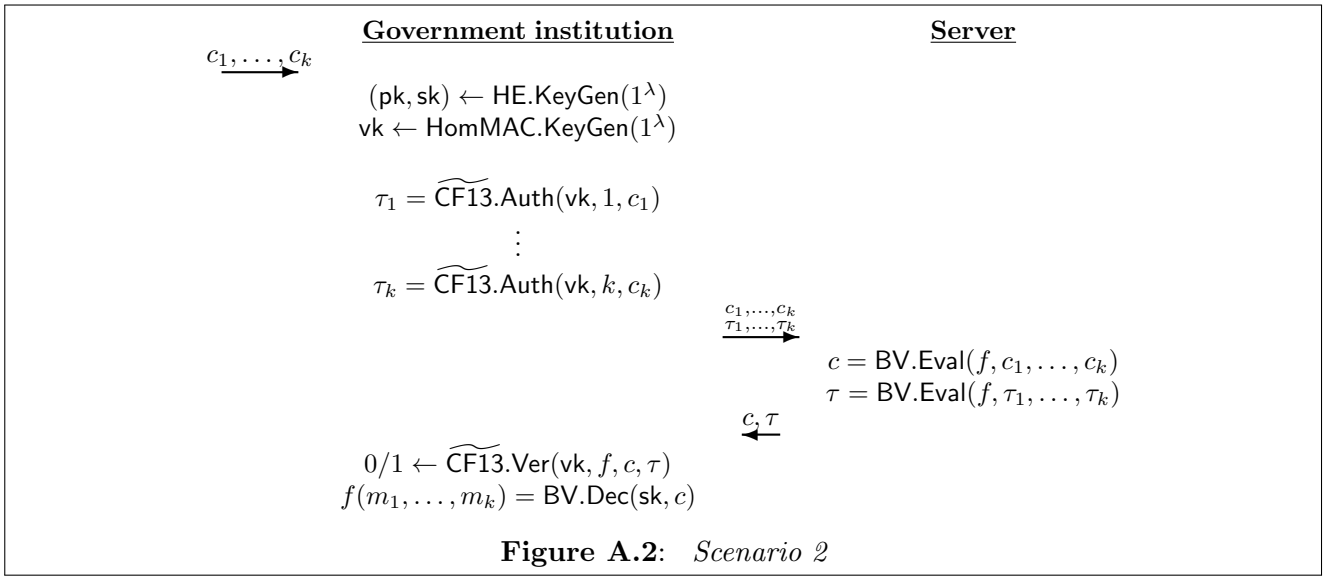


Figure A.2: *Scenario 2*