CHALMERS |  UNIVERSITY OF GOTHENBURG

Verifiable Delegation of Computation in the Setting of Privacy-Preserving Biometric Authentication

Master of Science Thesis in Computer Systems and Networks

JING LIU

Chalmers University of Technology
University of Gothenburg
Department of Computer Science and Engineering
Göteborg, Sweden, October 2015

Verifiable delegation of computation in the setting of privacy-preserving biometric authentication

JING LIU

Examiner: David Sands

Chalmers University of Technology
University of Gothenburg
Department of Computer Science and Engineering
SE-412 96 Göteborg
Sweden
Telephone + 46 (0)31-772 1000

Department of Computer Science and Engineering
Göteborg, Sweden, October 2015

Abstract

Cloud computing has gained popularity due to the growth of internet and the number of devices. Although outsourcing computation tasks to the remote cloud come with great convenience, there are increasing concerns regarding data *privacy* and computation *integrity* since the cloud providers are external third parties. Verifiable computation (VC) is a mechanism to let the client verify the computation result returned by the cloud as an integrity guarantee, which can be widely applied in various scenarios of computation outsourcing. In this thesis work we focus specifically on the setting of biometric authentication systems, where a user is granted access to some service based on biometric templates matching. It is very important to preserve the privacy of these templates as they contain many private information. Privacy-preserving can be achieved by homomorphic encryption, where the computation server only stores and performs computations on encrypted templates. Yasuda et al. proposed a biometric authentication scheme based on such mechanism [3]. However, a template recovery attack was discovered in the scheme as a result of malicious computation server and lack of integrity check [4].

The goal of this theory-oriented thesis is to choose a suitable VC scheme and integrate it into the biometric authentication scheme by Yasuda et al. in order to counter the aforementioned attack. The outcome is a new scheme BVC that allows the client to verify the correctness of the result returned by the computation server while preserving the authentication functionalities and templates privacy. We provided a general scheme description, a protocol description showing the interaction of different parties, and more importantly the actual construction of BVC with security and correctness analyses. In addition, we reflected on the template recovery attack and showed that the order combining a VC and homomorphic encryption is very critical. We presented an attack algorithm for malicious cloud to comprise the privacy of the computation outcome if the order is done in a wrong way.

Acknowledgements

I would like to show my sincere gratitude towards my advisor Katerina Mitrokotsa and her PHD student Elena Pagnin for all the constant support, guidelines and feedbacks they provided through out the thesis project and the positive encouragements when I encountered any obstacles. I could not have accomplished this work without the weekly group discussion meetings, especially the mathematical knowledge support.

I would also like to thank my family and my boyfriend for their patience, understanding and encouragements.

Jing Liu, Stockholm, 2015/08/13

Contents

1

Introduction

1.1 Motivation

NOWADAYS the rapid growth of internet has resulted in changes in the working and living patterns of a great number of people. A clear trend is that more and more tasks are connected by networks of various sizes and among them internet is the largest and most powerful one. If we look back to ten, twenty years ago, the number of digital devices per person was limited and the access to the internet was way less convenient and slower. In consequence, most commonly computation tasks were accomplished locally. To increase computing power or storage space, updating hardware would be the most realistic approach. However, today there are many blooming web services and applications covering areas of office-ware, education, finance, public/personal health, entertainment, environment, etc. For example, we have a private user *Bob*, who can now store all his photos in a remote file hosting server (such as Dropbox) and access them readily from his laptop, mobile phone and tablet as long as there is internet connection. Another example is web-based office software (such as Google Docs) which allows *Bob* to collaborate with other users in editing a document in real time.

The examples mentioned above provided us with a glimpse of the "cloud" technology, which has become a popular term not only for computer scientists but also private internet users. Essentially, cloud computing indicates an internet-based computing paradigm which allows private users or organizations to outsource services, data storage or software to a remote server [1] (a more detailed definition can be found in Chapter 2). This infrastructure enables devices with limited computational power, e.g. mobile phones, to off-load expensive data processing to a remote server by a "pay-per-use" manner using the shared resources, and hence achieve flexibility, scalability and cost-efficiency.

However, The cloud model also raises privacy and security concerns as the remote servers are in essence external third parties. If *Bob* is outsourcing private data to a remote server such as his health data recorded daily from his smart watch, he definitely

will not want the cloud to disclose this information or misuse it in anyway. Moreover, if some calculation is required (e.g. average blood pressure of a year), *Bob* might want to be sure the result returned is actually correct and trustworthy.

Regarding the security aspect of cloud computing, two significant challenges are how to guarantee data/computation *integrity* and *privacy*. The former property integrity addresses the correctness of the computation. Given that the external servers can be malicious, wrong computation results could be returned for the incentive of e.g. saving computational power or in a worse case to initiate some specific attacks. The clients will therefore need some mechanisms to verify the correctness of the delegated computations on the outsourced data. One relevant mechanism to achieve this requirement is verifiable computation (described more in Chapter 2). The second property, privacy, addresses the confidentiality issue where the clients may want to hide the sensitive data through encryption and under certain circumstances even hide the computational functions from the server in order to preserve privacy.

Now we shift our focus to a different area of security: user authentication, which is seen as the "primary line of defence" in the context of computer security [28]. In essence this is the step when a user can confirm his or her claimed identity. Password-based authentication is a widely used approach by mainstream web service and organizations where the client needs to provide both an ID and a secret pass phrase in order to login or be granted certain authorities. A common risk associated with this approach is the password being too simple and thus easily guessed or cracked as it is subject to brute force attack. On the other hand, long and complex passwords are cumbersome to be remembered and deemed inconvenient for certain user groups.

An alternative approach for user authentication is biometric authentication that a user is authenticated through his or her unique phisical characteristic, such as finger prints or retinal pattern. An example flow goes as the following: the user *Bob* first registers his fingerprint in the database. Then upon authentication, he provides his identity and a fresh fingerprint template, which shall be compared with the reference template stored in the database. If the similarity passes a threshold level, the authentication is successful. A more detailed description of biometric authentication systems can be found in Chapter 3.

Secure storage of the reference template is a key concern in any biometric authentication system [2]. Due to the irrevocable nature, leaked templates can cause even more serious consequences compared to cracked passwords.

Now we relate biometric authentication systems back to cloud computing. In order to benefit from the flexibility and other conveniences, the database that stores biometric templates as well as the component responsible for templates comparision can also be outsourced to a cloud service provider. We need to revisit the two security challenges: *integrity* and *privacy*. One way to preserve privacy is to encrypt the stored templates [3]. Nevertheless a template recovery attack has been proven possible [4] even under the encryption technique. The main reason is the lack of integrity check, where the malicious entities in the biometric authentication system are capable to deviate from the protocol and perform incorrect computations. In this special setting integrity is a more trivial

property compared to data privacy. The direct result is the leak of the stored templates.

1.2 Aim

We have argued in the motivation why it is crucial to preserve both integrity and privacy in a biometric authentication system. The aim for the thesis work is to integrate a verifiable computation scheme with a chosen privacy-preserving biometric authentication scheme [3]. The new scheme should maintain the privacy-preserving property of the original protocol and add integrity check for malicious computing entities. The main contribution is that it counters the template recovery attack discovered by Abidin and Mitrokotsa [4].

Another goal of the work is to reflect on the bit-flipping attack [4] for privacy-preserving biometric authentication and study if a similar attack can be exploited in cloud computing generally. The attack should demonstrate that encryption and verifiable computation cannot be simply combined in a naïve approach.

1.3 Scope and limitations

This thesis work focuses on the theory study. The outcome shall include a generic scheme, a protocol, and the scheme construction with appropriate correctness and scrutiny analysis. The new scheme is designed specifically for the Yasuda biometric authentication scheme [3] to counter a specific attack.

Therefore, implementations or any actual prototyping are not in the scope of the thesis. The new scheme in combining the explicitly chosen VC scheme and homomorphic encryption might be extended to be applied on a more generic level, e.g., covering a wider class of VC schemes and homomorphic encryption schemes.

1.4 Notations

Some common mathematical notations used through out this report are listed below:

- $\mathbb{Z}$ is the ring of integers. $\mathbb{Z}_p = \mathbb{Z}/p\mathbb{Z}$ and $\mathbb{Z}_d = \mathbb{Z}/d\mathbb{Z}$ are respectively rings of integers modulo p and rings of integers modulo d.
- The set of n-dimensional vectors with components in $\mathbb{Z}_p$ is donated by $\mathbb{Z}_p^n$.
- For two integers x and d, $[x]_d$ denotes the reduction of x modulo d in the range of $[-d/2,d/2]$.
- Vectors are denoted with bold letters. For example, $\boldsymbol{\sigma}$ denotes a vector of tags and σ denotes one single tag. Moreover, the i-th component of a vector $\mathbf{v} \in \mathbb{Z}_p^n$ is referred to as $v_i \in \mathbb{Z}_p$.
- $x \xleftarrow{\$} X$ denotes selecting x uniformly at random from the range X.

1.5 Thesis structure

The report is organized as follows. Chapter 2 describes the background and related work of cloud computing and verifiable computation. Chapter 3 presents the notions and related work of privacy-preserving biometric authentication, where we focus on one specific scheme by Yasuda et al. [3] and one attack detected in this scheme. Chapter 4 describes some mathematical tools that will be used in the scheme construction. In chapter 5 we present the results of the thesis: a new scheme combining verifiable computation with the biometric authentication protocol [3]. We present in turns the generic scheme, the protocol execution and the detailed construction. Moreover, we show a flawed approach combining verifiable computation and privacy-preserving in a naïve way. Then we wrap up chapter 4 with a discussion. Last but not least, chapter 6 concludes this thesis and outlines some future work directions.

2

Verifiable computation

Verifiable computation (VC) is a mechanism that enables clients to outsource the computation of certain functions to remote servers and gives the possibility to check the operations performed by the servers. This chapter introduces the background of verifiable computation, some formal definitions and current main solutions.

2.1 Background

2.1.1 An overview of cloud computing

Nowadays there is a climbing trend to deliver hosted services over the internet rather than isolating data storage and processing on a local device. The word "cloud computing" is growing popularity rapidly, even among non-computer specialists. A formal definition of cloud computing is that it is a model that enables ubiquitous on-demand network access to a shared pool of configurable computing resources (including storage, processing, memory and network bandwidth) which can be conveniently provided and released with low management effort [5]. It is a promising and adaptive paradigm that grows its influence on both large scale business and private users.

The cloud model is composed of three different service models, including *Software as a Service (SaaS), Platform as a Service (PaaS)* and *Infrastructure as a Service (IaaS)* [5]. The capability provided to the client increases in the aforementioned order. In SaaS, the applications are owned and run by the cloud service provider and can be accessible from the devices owned by the client. PaaS allows the client to deploy self-created applications onto the cloud infrastructure and configure the environment for the hosted application. Finally, IaaS grants the highest capability that the client could even manage the operating systems, the environment for arbitrary software and probably selected network components (e.g. firewall) [5].

Besides, the cloud model can be categorized into four deployment models: *private, community, public* and *hybrid*. Private cloud is provided exclusively to a single organization. Community cloud is provisioned for a certain community or different parties sharing similar concerns. Public cloud, as the name suggests, is provided open to the public. Lastly, hybrid cloud could be a combination of distinct cloud infrastructures.

Cloud computing has a list of essential characteristics. Rather than making investments in purchasing new hardware and managing all the data storage and computation in-house, the client can purchase computation power in desired amount from a cloud service provider (SP). Thus the client is greatly relieved from the effort of hardware/software infrastructure maintenance. Moreover, the size of the client platform (which could be workstations, PC, laptops, mobile phones, tablets, wearable devices etc.) is no longer a constrain for the service it can provide since the heavy data storage and complex computation is outsourced to much more powerful machines. This is very useful as nowadays the amount of digital devices is increasing dramatically and the trend is that the devices form a common network to promote information flow and knowledge sharing. Lastly, the outsourcing capabilities are very dynamic. The computing resource can be very elastically provided and released, which allows the client to cope better with the rapidly scaling market demand. From a service provider's point of view, the resource can be managed in a "pool" manner and dynamically assigned and re-assigned to multiple clients to suit their various levels of demands regardless the physical location of the clients. Hardware maintenance can also be carried out in a centralized way.

Having mentioned some major advantages of cloud computing, this paradigm also comes with risk. As the name "cloud" suggests, the SPs are complex "black boxes" operating distantly from the client. Thus, it is usually more difficult to detect any misconfigurations, corruptions in data storage and transition, and in worse situations even malicious computations that intentionally bring harm to the interests of the clients. The SPs do not always have the strongest incentive to provide integrity guarantee. This raises the concern of the client: we need a mechanism to know that the cloud is trustworthy and performing correct tasks. This is the motivation behind the area verifiable computation.

2.1.2 Background notions

This section explains some cryptographic terms that will help the readers get a better understand of the schemes/protocols to be described.

Public-key encryption

All the encryption in this thesis refer to public-key encryption, which is also called asymmetric encryption. The algorithm requires two separate keys: one *public key* and one *private key*, and encryption and decryption rely on different keys. If the goal to use public-key encryption is to ensure confidentiality so that only the appointed receiver can read the message, the plain text shall be encrypted with the receiver's public key and the cipher text shall be decrypted with the receiver's secret key. This scenario is

the most relevant one to the thesis. In addition, public-key encryption is also the backbone of digital signature (private signing key, public verification key), which guarantees authenticity of the message that the sender cannot deny having sent the message.

Public-key encryption is based on hard to solve mathematical problems such as factorization of large integers or computing a discrete logarithm in a large group [6]. It should be computationally infeasible for an adversary to determine the private key knowing the public key, or to recover the plaintext knowing the public key and the corresponding ciphertext [7].

Homomorphic encryption

Homomorphic encryption is a type of encryption schemes that allows computations to be performed on cipher text. In normal encryption scheme, if the receiver wants to process the message, he or she will need to decrypt the data first. However, this leads to privacy concern, especially in cloud computing where the server is an external party from which the client may wish to hide the sensitive data. For example, homomorphic encryption will allow a cloud service provider to process financial transactions directly on the cipher text level and return the result in encrypted format too. We provide a more formal definition for homomorphic encryption in below [6].

Definition 1 (Homomorphic encryption)**.** *Let $\mathcal{M}$ denote the space of plantexts and $\mathcal{C}$ denote the space of ciphertexts and there is operator $\odot$. An encryption scheme is said to be homomorphic if for a given key k the encryption function E satisfies:*

$$\forall m_1, m_2 \in \mathcal{M}, E(m_1 \odot m_2) \leftarrow E(m_1) \odot E(m_2),$$

where $\leftarrow$ means directly computed from without transitional decryption. Moreover, if the operation is "addition", we say the scheme is additively homomorphic. Similarly, if the operation is "multiplication", we say the scheme is multiplicatively homomorphic.

Somewhat/Fully homomorphic encryption (SHE, FHE)

A homomorphic encryption scheme can also be categorized into somewhat homomorphic or fully homomorphic based on the range of functions it can be applied to. Somwhat homomorphic encryption schemes (SHE) only support a limited number of homomorphic operations, e.g., only additive homomorphism, or multiplication up till a certain degree. On the other hand, fully homomorphic encryption (FHE) does not have any constraint regarding the circuit depth and can evaluate arbitrary functions. This property should make FHE a powerful tool in constructing various privacy-preserving systems. However, in reality FHE comes with a heavy overhead and its poor practicality becomes its drawback. With regard to the efficiency aspect, SHE, if used appropriately, can be much faster and more compact than FHE [8].

Verifiable computation in a nutshell

The notion of verifiable computation is first defined and formalized by Gennaro et al. [9]. A verifiable computation scheme (VC) on a general level is a two-party protocol between a client and a server. The client chooses a function and delegate the function with the corresponding inputs to the server. The server is required to evaluate the function based on the inputs and respond with an output, which in the next step can be verified by the client to ensure it is indeed the correct result of the given function and inputs.

Verifiable computation is the core topic of this chapter. In below we present the related work in this area on a more detailed level and then show one specific scheme, which will be used as a building backbone in our scheme construction in Chapter 5.

2.2 Related Work

2.2.1 Proof-based approaches

The theoretical community has vastly investigated in the topic of verification computation. One fundamental area is proof-based approaches. Namely, it involves two parties: a *prover* and a *verifier*. The goal of the prover is to convince the verifier of some mathematical assertions (called the *proof*) [10]. On the other hand, the verifier should be able to check the proof. The verification procedure is very crucial in defining a proof systems [11]. Probabilistic proof systems are the most relevant in the area of verifiable computation, which include e.g. interactive proofs (IPs) and probabilistic checkable proofs (PCPs).

Figure 2.1 illustrates a theoretical framework where a verifier can check whether the result y returned by the prover is correct for a computation p and specified input x. The figure is a modified version from [10]. There are four important steps as marked in the figure. Step(1) is to compile and express the computation p into a Boolean circuit from a high-level language. In step(2), the prover performs the computation and obtains a transcript, which is essentially an assignment of the input values to the circuit. In step(3), the transcript is encoded to a larger string to assist the verifier in issuing efficient querying/testing. In step (4) the verifier probabilistically send queries regarding the encoded transcript and the actual structure of this step varies in different protocols.

First there are interactive proofs (IPs). The verification procedures for IP systems are randomized and interactive [12]. The proof is probablistic in nature. If the input statement has length n, the proof might be faultily accepted with a small probability of $1/2^n$. On the other hand the legal correctness shall be approved by the verifier with a comparatively much larger probability $(1 - 1/2^n)$. Note that randomness is very crucial in IPs. If the verifier is deterministic (e.g. ask predictable questions), the prover can simply supply a pre-determined sequence of answers that known for sure the verifier will be convinced about [11]. The second property interactivity implies that in order for the verifier to efficiently verify the proof, the verifier must actively ask questions and receive replies from the prover. So these protocols typically run in rounds. Goldwasser et

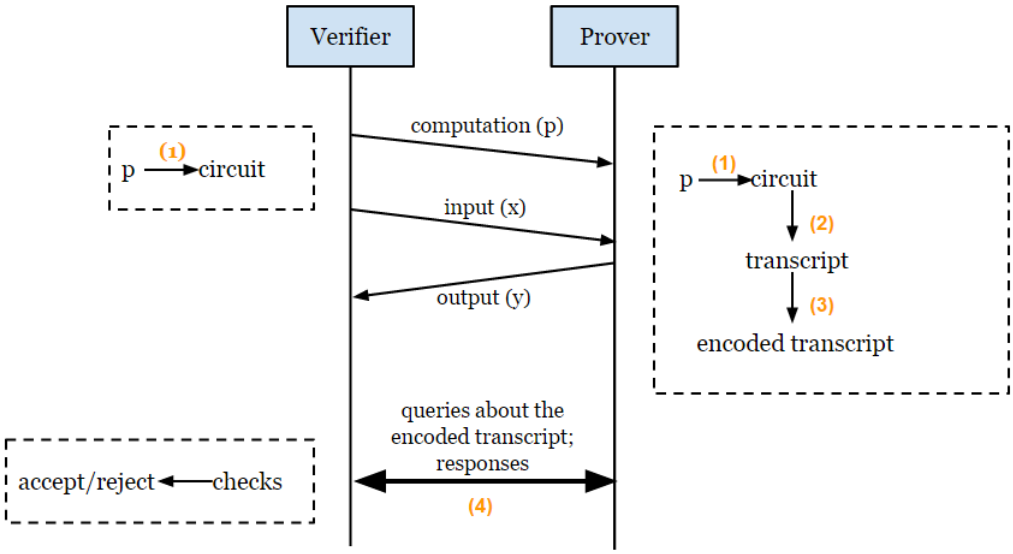

Figure 2.1: A theoretical framework for solving prove-based verifiable computations [10].

al. in [12] gave a computational complexity measurement and also proposed to classify languages depending on the amount of additional information that must be released. However, one drawback of IPs is that it demands a powerful prover (super-polynomial) [9] which takes exponential-time and might be prohibitive [10]. Also it is not saving much work for the verifier.

The work of IPs further on leads to the concept development of probabilistically checkable proofs(PCPs), which is another crucial candidate in the probabilistic proof systems. It works by the prover first preparing and committing to the complete content of the encoded transcript serving as the proof. Then the verifier can choose specific locations in the proof and only requests the value contained in these chosen locations. The prover is expected to answer according to the pre-commited transcript [10]. However, PCPs suffer from the impracticality that the proof could be too long and heavy for the verifier to process [9]. To cope with this problem, Kllian proposed the usage of efficient arguments [13]. An argument in this context means a computationally sound proof, in which the prover is assumed to be computationally bounded. Thus the prover sends the verifier a short commitment of the proof using a Merkle hash tree rather then the complete proof content [9]. The prover will follow PCP theorem and opens up bits at the requested places by the verifier. Applying PCP machinery, Goldwasser et al. in their influential paper built an IP to verify polynomial-time computation in approximately linear time [11]. In the same paper they also contributed on non-interative argument for a confined class of functions.

While the early work depends on PCP, recent proof-based schemes rely on SNARKS (Succinct non-interactive arguments of knowledge), where the length of the proof is

greatly shortened im comparison to the length of the statement [14]. Considering the succinctness of the arguments, the verification step can be carried out efficiently. However, SNARKS would require either the random oracle model or non standard, non-falsifiable assumptions [15]. There have been systems implemented such as Pinocchio [16] and ADSNARK [17] for verifiable computation based on SNARKS that demonstrates its practicality.

2.2.2 FHE as a tool

In 2009 Gentry published his break-through work on the first plausible fully homomorphic encryption (FHE) scheme using lattice as the underlying tool [18]. The scheme supports optional amount of addition and multiplication operations on the ciphertext so that it is possible to process any arbitrary functions. This leads to new constructions for verifiable computation based on FHE.

Gennaro et al. proposed a non-interactive scheme for verifying arbitrary functions based on Yao's garbled circuit and FHE [9]. The whole scheme comprises three stages: *pre-processing*, *input preparation* and *output computation/verification*. The client first garbles the circuit using Yao's construction in the preprocessing stage. After that the client will reveal the random labels affiliated with the inputs x to the worker in input-preparation. The server will then calculate accordingly the output labels (adequately long and random), which allows the client to reconstruct the function f and perform verification check. The role of FHE is to allow the same garbled circuit to be safely reused for different inputs x for the same function. Rather than revealing the input labels in plain text, the labels are encrypted with a FHE scheme. Thanks to the homomorphic properties, the server performs the calculation on the cipher text instead and returns the output labels in encrypted format, which needs to be decrypted by the client to construct the function. Note that a new public key will be generated for different inputs so that the server does not learn any information from the input labels in the case of function-reuse. Regarding efficiency, this scheme uses "amortized notion of complexity" which indicates that the preprocessing is a one-time offline stage which will take time similar to computing the function from scratch. The important requirement is that the next two online stages (which will be repeated) for the client should be more efficient (verified in $O(m \cdot poly(\lambda))$ time where m is the output bit-length and λ the security parameter). This efficiency requirement lets clients with constrained computation power verifies efficiently the result returned by the powerful server in an amortized sense. Furthermore, another great contribution of this paper is that it first defines and formalizes the notion of Verifiable Computation [9].

Following the work by Gennaro et al. [9], Chung et al. designed two constructions that improved the verifiable computation scheme based on FHE [19]. The first construction eradicates the large public key from Gennaro et al. scheme. Their second construction improves the efficiency of the offline stage for the client at the price of having a fixed-size 4 message interaction between the client and the server.

Nevertheless, FHE-based schemes are very expensive due to the complexity of FHE [10]. This line of work is more theoretical then practical.

2.2.3 Homomorphic authenticators

Another direction of investigation is to extend message authentication primitives with homomorphic properties. A primitive is that every data item in the input space is attached with an authenticator. Then for every operation in a computation taking authenticated data as inputs will produce the corresponding output and a new valid authenticator. There are two types of homomorphic authenticators: Homomorphic MACs are adopted in a symmetric setting and homomorphic signatures are used in an asymmetric setting. Essentially the former supports private verification and the latter supports public verification [20].

Gennaro and Wichs introduced homomorphic MACs and provided a fully homomorphic construction that supports arbitrary computation [20]. In this scheme a short tag, independent of the authenticated input, can be generated to authenticate the result of the computation over previously authenticated data. The client can use this tag to verify the correctness of the claimed output of the computation returned by the untrusted server using a secret key. There is no need for the client to keep track of the underlying data (which is outsourced to the "cloud"). Furthermore, this scheme is proven secure in a weaker model where the adversaries are not allowed to issue verification queries. Another homomorphic MACs scheme was later proposed by Catalano and Fiore which supports a limited set of functions (i.e., polynomially-bounded arithmetic circuits) rather than arbitrary functions. [21]. However, as it is not dependent on FHE, this scheme is much more efficient and straightforward to implement and it can tolerate verification queries. Then Backes et al. further built on the scheme and introduced a first practical realization of homomorphic MACs with efficient verification [15], which works for quadratic polynomial over a large number of polynomials. Namely, the verification should take less time to process rather than being proportional to the description of the function. The efficiency is achieved in "amortized closed-form" by having an offline stage (preprocessing) and an online stage. The online stage is when the same function is evaluated on different inputs and it takes constant time. This scheme also introduces the notion of multi-labels that consist of a dataset identifier and an input identifier, which is a crucial component in realizing the amortized closed-form efficiency.

Homomorphic signatures were first proposed by Johnson et al. in 2002 [22]. It is the public-key variant of the homomorphic authenticators, which means that the verification step requires only the public key and thus allows public verification. There have been many homomorphic signature schemes proposed for linear functions both in the random oracle model and in the standard model [15]. But linear functions are very restrictive in verifiable computation. The first construction targeting non-linear multivariant polynomials of constant degree was proposed by Boneh and Freeman [23]. Then Catalano proposed an alternative scheme that improved Boneh and Freeman's scheme from three perspectives [14]: not relying on the random oracle assumption, proven secure in a stronger adaptive model and achieves efficient verification in an amortized sense(same primitive as in [15] described in the previous paragraph).

2.2.4 Privacy-preserving verifiable computations

Fiore et al. pointed out that in the state-of-the art-research of verifiable computation, there is very few work that addresses on data privacy in a restricted model [24]. In many schemes the data is exposed in the clear.

The protocol in [9] was claimed to achieve input/output privacy since it is based on FHE, but it is not secure against verification queries, i.e., if the malicious cloud learns whether the verifier accepts/rejects the result, it can obtain information of the input.

One other scheme that addresses both integrity and privacy was proposed in [25], which uses a functional encryption (FE) scheme to construct a VC scheme. It claims that the scheme operates on encrypted data as inputs and the computation is publicly verifiable. Nevertheless, Fiore et al. pointed out that the security motivation in [25] is not reinforced with any formal definition or proof. Furthermore, They found a number of other issues of the scheme such as unclear adaptivity and inherently bit-based [24].

Meanwhile, Fiore et al. proposed a generic protocol of VC based on FHE. The idea is to encrypt the input x with FHE and the VC scheme is run on the encrypted function [24]. Thus, the acceptance bit is determined before the outcome is decrypted. Besides the theoretical contribution, in their work they also proposed practical constructions suitable for a wide class of functions based on somewhat homomorphic encryption, in specific a simplified version of the BGV encryption [26]. For integrity, homomorphic MAC is used. One other key contribution of the paper is homomorphic hash functions, which compresses a multi-component BGV ciphertex into a single component in a predefined field. This improvement boots the performance of the scheme to a great extent by avoiding a 10^4 overhead [24].

2.2.5 Other approaches

The security community also designed alternative solutions to achieve different aims as the above mentioned ones. One branch is audit-based solutions. In this context, the client is required to check a small portion of the computation returned by the remote server. Belenkiy et al. [27] considered a scenario with one client (the boss) distributing computational tasks to a number of workers (the contracters), some of which might be malicious. The main contribution in [27] is a credit system that sets incentives/fines for rational/malicious contractors. the strategies analyzed are random double-checking by the boss and tasks replication and deployment to multiple contractors. However, the audit-based solutions usually require the clients to redo some parts of the calculations, which is not very feasible for resource-constrained clients [9]. Moreover, audit-based solutions could focus on the scenario with a pool of contributors (multi-party protocols) and the accuracy of the cloud depends on having a threshold value for the fraction of honest contractors. This fault tolerance technique, which is similar to the "Byzantine approach", is not suitable when the remote server is viewed as one whole entity.

2.3 A homomorphic MAC-based VC scheme

Out of all the approaches to realize verifiable computation, we present here the general scheme by Backes et al. [15] based on homomorphic MAC, one of the authenticators mentioned in the related work. This scheme will be the fundamental backbone of our scheme construction in chapter 5. Note that the original name of the scheme is $\mathsf{HomMAC-ML}$, which is a shortened for "homomorphic message authentication codes for multi-label programs". In this thesis report we rename the scheme as VC for the purpose of simplicity and consistency in describing the new scheme.

The VC scheme [15] comprises a number of algorithms: $\mathsf{VC} = (\mathsf{KeyGen}, \mathsf{Auth}, \mathsf{Ver}, \mathsf{Eval})$. We first explain some specific terms used in the scheme (which shall be re-used in the new scheme construction). Then we describe each of the algorithms more precisely.

Multi-label

Multi-labels essentially are useful identifiers for the variables. The term "multi-" indicates there are more than one components in one label. In the particular context, a multi-label is defined in two parts as $L = (\Delta, \tau)$, where Δ is a *dataset identifier* and τ is an *input identifier*. The splitting is useful in identifying both the concrete data items and the variable inputs of a programs [15]. We need a complete pair (Δ, τ) to identify a particular input item. Thus we say a message m can be authenticated with respect to a multi-label L. The label is unique for each message m. In other words, no two different messages m_1, m_2 can share the same multi-label.

We provide one example of multi-labels applied in practice. Suppose we are recording the hourly temperature of a city for a whole year. To identify a specific entry such as "The temperature at 15:00 on Jan 1, 2015", we can use a multi-label $L = (\Delta, \tau)$ where $\Delta =$ "Jan 1, 2015" and $\tau =$ "15:00". The label loses its uniqueness if any of the two parts is missing.

Multi-label program

A Multi-label program is defined as $\mathcal{P} = ((f, \tau_1, \ldots, \tau_n), \Delta)$, which is essentially a function $f : \mathcal{M}^n \to \mathcal{M}$ defined on the range $\mathcal{M}$ that takes in n variables and a set of multi-labels. Each variable is associated a multi-label. The labels in the same program all share the same Δ, the *dataset identifier*, and they vary in τ, the *input identifier*. Thus the i-th variable in f has the label structure $L_i = (\Delta, \tau_i)$.

An example in practice could be $\mathcal{P} = ((f_{avg}, \tau_1, \ldots, \tau_{24}), \Delta$ where f_{avg} calculates the average hourly temperature of a certain date. The date is specified as Δ (e.g. "Jan 1, 2015") and the τs represent every sharp hour.

The general scheme

Finally we present the algorithms in the $\mathsf{VC} = (\mathsf{KeyGen}, \mathsf{Auth}, \mathsf{Ver}, \mathsf{Eval})$ scheme. Note that here we only present the general schemes focusing on the input and output of each

algorithm. In chapter 5 we show how the actual construction of VC contributes in building the new scheme (integrated with privacy-preserving biometric authentication). The details of the complete construction of VC by itself can be found in the original paper [15].

KeyGen$(\lambda) \rightarrow (ek,sk)$: Given the security parameter λ, this algorithm randomly generates a secret sk and a public evaluation key ek.

Auth$(sk, L, m) \rightarrow \sigma$: Given the secret key sk, a multi-label $L = (\Delta, \tau)$ and a message m in the valid input range, this algorithm should output an authentication tag σ .

Ver$(sk, \mathcal{P}_\Delta, m, \sigma) \rightarrow acc$: Given the secret key sk, a multi-label program $\mathcal{P} = ((f, \tau_1, \ldots, \tau_n), \Delta)$, a message m in the valid input range and an authentication tag σ, this verification algorithm returns an acceptance bit acc: "0" for rejection and "1" for acceptance.

Eval $(ek, f, \boldsymbol{\sigma}) \rightarrow \sigma$: Given the public evaluation key ek, a circuit f and a vector of tags $\boldsymbol{\sigma} = (\sigma_1, \ldots, \sigma_n)$, this evaluation algorithm produces a new tag σ.

The scheme satisfies a number of properties [15]. First of all, *authentication/evaluation correctness* requires that the tag σ produced by the *Auth* or *Eval* algorithms are legitimate, with regard to a message m with its multi-label L, or in the later case a circuit f and a vector of authentication tags $\boldsymbol{\sigma}$. In other words, if this tag σ is tested in a Ver algorithm with the corresponding m and multi-label program $\mathcal{P}$, the result should always be "accept". The next property is *succinctness*, which specifies that the size of the tag σ is bounded in the security parameter and does not depend on the number of variables n. Finally the scheme has to follow the notion of *security*, where the authentication tags should be unforgeable for any PPT Adversary.

3

Biometric authentication

This chapter introduces the reader to the area of biometric authentication with the focus on the privacy-preserving property. In the background we provide an overview to this area, present some definitions to better readers understanding and provide a summary of some existing biometric authentication protocols in the current state of research. Finally we focus on one specific scheme and present an attack discovered in this scheme, which is countered in the result section of this thesis.

3.1 Background

Biometric authentication is a way to identify and verify an individual user by examining his or her physical characteristics. Nowadays we see this technology being implemented more and more widely into personal devices. One example is the fingerprint login functionality which is fairly common in touch screen mobile devices. Besides finger prints, other typical traits used in biometric authentication include facial characteristics, hand geometry, retina/iris patterns and voices recognition [28]. The greatest advantage of biometric authentication is that the users do not have to provide extra passwords, which increasingly become a target for attackers to steal or crack. Moreover, whilst short passwords are subject to brute force attack, remembering long and cumbersome passwords is not very user-friendly. Biometric features are unforgeable and unforgettable and the chances are extremely low that they are stolen.

The operation of a biometric authentication system on a generic level is described below and illustrated in Figure 3.1. First the user who is going to be included in the authorized group will be enrolled in the database. This step would require a user identifier and a sensor that captures the biometric characteristic. The output of the sensor will be processed and digitalized into a numerical template and stored in a database. Depends on the type of system, user authentication can be carried out in two ways: *identification* and *verification*. In an identification system, the user only supplies fresh

biometric data through a sensor with no extra information. The outcoming template is compared against a set of entries in the database to check whether there is any successful matching. In a verification system, the user uses the sensor to capture his or her fresh biometric data and supplies an identifier (ID, PIN, etc). The system will extract a feature template and compare it with the corresponding stored template in the database for this specific user. The user is authenticated successfully if a match is found [28]. Note that we focus on the verification system type in this thesis. Essentially biometric authentication is based on pattern recognition [28].

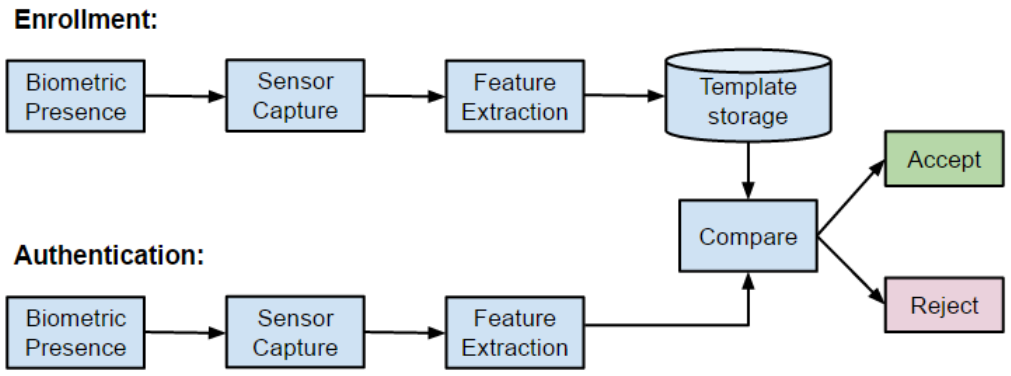

Figure 3.1: The general operation flow of a biometric authentication system.

One concern of biometric authentication is its accuracy. The biometric characteristics are processed into a digital representation and the complexity of the physical traits makes it impossible to find a perfect match [28]. The systems use different algorithms to generate a "matching score" that will be compared against a threshold value to measure the similarity between two templates quantitatively. Nevertheless, sensor noises, aging, injuries and user training can all cause false matches or false non-matches. Thus it is crucial to define an appropriate threshold value. In general, a high-security system demands a very low false match rate. On the contrary, a forensic system will require low false non-match rate to not miss any possible candidates.

Another concern of biometric systems is security and privacy of the biometric templates, which is a problem in focus in this thesis project. Leaked biometric templates, essentially the stored reference data, can reveal sensitive information such as genetic structure or special diseases. To make things worse, whilst compromised passwords can be reset, compromised biometric templates can neither be revoked nor replaced due to its inherent nature [3]. Hence protecting the stored templates and hiding them from the server is a very crucial mechanism in preserving user privacy, especially when the operation is outsourced to a cloud service provider.

3.1.1 Background notations

This section explains some technical terms that will help the readers get a better understand of the schemes/protocols to be described.

Authentication

User authentication is usually considered as the primary line of defence in system security, resource management and access control [28]. In principle authentication can be further divided into two sub-steps: *presenting an identity* and *verifying the identity* (binding between the claimed identity and the actual entity).

There are three known means in authenticating an acclaimed identity regarding an individual [28]. First, there is *something the individual knows*, which includes the most common password-based authentication or answering pre-registered questions. Then there is *something the individual has* (referred to as "token"), such as access cards, smart cards or physical keys. Lastly, there is *something the individual is*, which biometric authentication belongs to (e.g., fingerprint, retina, etc). Each of these means has its advantages and disadvantages. As mentioned earlier, passwords can be forgotten/cracked, tokens can be stolen and biometric authenticators might suffer from high cost, privacy and accuracy issues. While all these means can be used alone, a more common approach is to combine two or more of them to obtain so-called *multi-factor authentication* to compensate for the shortages of each method, and hence achieves better security.

Hamming distance

When comparing whether there is a match between a stored biometric template and a fresh biometric template, it usually involves computing some sort of distance. The Hamming distance is one very common metric used in this context (e.g., adopted by the Yasuda scheme [3], which we will elaborate on in this thesis). The principle is to measure the similarity of two equal-lengthed strings (or vectors) by counting how many positions are different. In other words, the Hamming distance between a string A to a string B is the minimum substitutions of bits in string A required to achieve $A = B$. For example, the Hamming distance between $A = 111000$ and $B = 110001$ is 2 (flipping the third bit and the last bit of A).

Besides Hamming distance, other types of straightforward distances in templates comparison include e.g., the normalized Hamming distance and the Euclidean distance [29].

3.2 Related work

3.2.1 Privacy-preserving Biometric Authentication

As we have described the severe impacts of leaked biometric templates, the privacy-preserving property has been given very high priority in the current state of research

of biometric authentication. In below we present several main approaches for privacy-preserving biometric authentication protocol [3]:

Feature transformation

The essence of this approach is to hide the original biometric data by transforming the data into a random representation in another domain. It usually requires a secret key only known by the client or some sort of pass phrases. One typical example is the *cancelable biometrics* technique, which was first proposed by Ratha et al. in [30]. This technique relies on repeatable distortion of biometric signals following a chosen transform method. Since every enrollment istance can follow different transform methods, this technique achieves unlinkability, i.e., the templates cannot be associated to the right clients. Further more, even if one transform variant is compromised, the transform function will allow a new transform variant to be performed to "re-enroll" the client into the system [30]. One crucial requirement is that the biometric transformation is always a one-way function, so it is computationally hard to trace back and recover the original biometric data from a distorted representation.

Another example of feature transformation approach is called *BioHashing*, which applies two-factor authentication by incorporating two elements: a psydorandom number and the biometric data [31]. A *BioCode* is generated from feeding these two elements into a one-way hashing function. The general mechanism works as follows: in the enrollment phase, the server stores the BioCode and the psydo-random number; in the authentication phase, the server generates a fresh BioCode using the stored number and the fresh biometric data provided upon authentication, which is compared against the stored BioCode (e.g. by computing the Hamming distance) to determine if the authentication is successful. In general, privacy-preserving realized by feature transformation achieves ideal practicality performance-wise, but the security is questioned if the client secret is compromised [3].

Biometric cryptosystems

This approach is based on secure sketch and error tolerance. Secure sketch in the context of biometric authentication requires recovering a template b from any adequately-close template b' and some extra data D. Meanwhile, D is forbidden to disclose too much information of the original template b [31]. The pioneer work of secure sketch was proposed by Juels et al. in [32] named *fuzzy commitment*. This scheme uses error-correcting codes and the application in biometric authentication system is as follows [31]: in the enrollment phase, the client records her biometric b and the server computes and stores the sketch $D = c \otimes b$ as well as $H(c)$, where c is a randomly chosen *codeword* and H is a hash function; in the authentication phase, the client provides a fresh biometric b', the server computes and decodes $c' = D \otimes b'$, checks whether $H(c) = H(c')$ and makes authentication judgement.

Another approach is called *fuzzy vault* [33], which is also based on secure sketch but with a different mechanism. The secret value is "locked" by one party in the vault using a

set of elements A from a public domain. Another party can only retrieve this secret and "unlock" the vault with a set B of the same length as A and overlaps with A extensively [33]. Juels et al. also pointed out in the paper that fuzzy vault is suitable in the setting of biometric authentication as it provides resistance to a certain amount of errors/noises as a result of signal processing.

Homomorphic encryption

This approach protects the biometric templates by cryptographic encryption techniques. We have mentioned the definition of homomorphic encryption in Chapter 2 and we provide a brief revision here: this property allows the similarity of two templates(stored and fresh) to be compared in the encrypted formats using common metrics such as Hamming distances [3]. Thus the original templates are never exposed in public. Only the trusted parties who possess the secret key have the rights to decrypt the cipher text.

The performance/rfficiency of the biometric authentication systems are subject to which specific homomorphic encryption method is applied (e.g., somewhat homomorphic (SHE) or fully homomorphic(FHE)). The heavy overhead of the computation and the size of the ciphertext are considered as the main issues in this approach [3]. This is also the approach that we will focus on in this thesis work.

3.2.2 Outsourced Biometric Authentication

In the last section of the related work, we will focus on the keyword "outsourcing" in the context of biometric authentication, which is the most relevant to the goal of this thesis work. Sedenka et al. recently published a paper and proposed the first efficient privacy-preserving protocols for outsourcable authentication using scaled Manhattan and scaled Euclidean verifiers [34]. The outsourced authentication is visualized as two-parties: the client holding a device, and the outsourced authentication server. The scenario Sedenka et al. considered was malicious client and honest-but-curious authentication server. Their construction is based on garbled circuit and additive homomorphic encryption. In addition, they also evaluated the performance of the scheme by conducting experiments with smartphones and demonstrated accuracy and usability of the protocol.

3.3 The Yasuda scheme

Now we shift our focus to one specific biometric authentication scheme which is relevant to this thesis work. Yasuda et al. proposed an efficient scheme by computing the Hamming distance with homomorphic encryption based on ideal lattices [3]. They adopted somewhat homomorphic encryption to achieve faster computation, which was adapted from the work by Gentry and Halevi [35]. Their main contributions comprise the proposal of new packing methods to pack a feature vector into a single ciphertext rather than running the encryption bit-by-bit, and the proposal of a new privacy-preserving authentication protocol applying SHE. We will introduce the SHE construction and the protocol in below.

3.3.1 Construction

Now we present the algorithms in the construction $\mathsf{SHE} = (\mathsf{KeyGen}, \mathsf{Enc}, \mathsf{Dec})$. Here we only show a brief version. In chapter 5 we will expand the explanation of each variable and show how the actual construction of SHE contributes in building the new scheme (integrated with verifiable computation). The details of the complete construction of SHE and the details of ideal lattices can be found in the original paper [3].

KeyGen$(\lambda) \rightarrow (pk,sk)$: Given the security parameter λ, this algorithm randomly generates a secret sk and a public key pk for the homomorphic encryption. The keys returned have the format $pk = (d, r, n, s) \in \mathbb{Z}^4$ and $sk = \omega \in \mathbb{Z}$. A valid key pair must obey the following conditions: $gcd(\omega, s) = 1$, $gcd(\omega, s) = 1$, and $r^n \equiv -1 \mod d$ [4].

Enc$(m, pk) \rightarrow ct$: This algorithm encrypts a message in the valid message space $m \in \mathbb{Z}_s$ with the public key $pk = (d, r, n, s)$ and ouputs a ciphertext ct. we first need to generate a "noise vector", whose purpose is to introduce randomness and makes the encryption scheme indeterministic. The random vector is chosen as $\mathbf{u} = (u_0,u_1,\ldots,u_{n-1})$ with $u_i \in \{0, \pm 1\}$. u_i has q probability being 0, $(1-q)/2$ probability being 1 and $(1-q)/2$ probability being -1. The corresponding cipher text of m is defined as

$$ct = \left[m + s \sum_{i=0}^{n-1} u_i r^i \right]_d \in \mathbb{Z}_d.$$

Dec$(ct, sk) \rightarrow m$: This algorithm decrypts the ciphertext $ct = Enc(m, pk)$ with the secret key $sk = \omega$ and recovers the plaintext message m. The computation is:

$$m = [ct_{HD} \cdot \omega]_d \cdot \omega^{-1} \mod s.$$

The packing methods

In the Yasuda scheme they assume the feature vectors are 2048-bit binary vectors. If a vector is encrypted bit-by-bit, the resulting ciphertext will be very lengthy. Therefore, the packing methods were proposed to pack the feature vector into one single ciphertext. More specifically, the procedure transforms a binary vector into a polynomial bounded by a ring and then the polynomial is encrypted. There are two types of packing methods which are asymmetric in structure. In addition, in the protocol section we will describe which type of packing methods shall be applied at which stage of the protocol run. The two packing methods are defined as follows:

Definition 2 (packing methods $vEnc_i$)**.** *[3]*

- *Type 1. Given a binary vector A and the public key $pk = (d, r, n, s)$, we let $F_1 : A = (A_0, \ldots, A_{2047}) \mapsto \sum_{i=0}^{2047} A_i x^i \in R = \mathbb{Z}[x]/(f_n(x))$. Then the type 1 packed cipher text is computed as:*

$$ct(A) = vEnc_1(A) = [F_1(A)(r) + su_1(r)]_d = \left[\sum_{i=0}^{2047} A_i r^i + su_1(r)\right]_d \in \mathbb{Z}_d.$$

- *Type 2. Given a binary vector B and the public key $pk = (d, r, n, s)$, we let $F_2 : B = (B_0, \ldots, B_{2047}) \mapsto -\sum_{i=0}^{2047} B_i x^{n-i} \in R = \mathbb{Z}[x]/(f_n(x))$. Then the type 2 packed cipher text is computed as:*

$$ct(B) = vEnc_2(B) = [F_2(B)(r) + su_2(r)]_d = \left[-\sum_{i=0}^{2047} B_i r^{n-i} + su_2(r)\right]_d \in \mathbb{Z}_d.$$

Note that $u_1(x)$ and $u_2(x)$ denote noise polynomials as described in the encryption scheme.

Secure Hamming distance computation

Yasuda et al. have proven that the two types of packing methods allow efficient computation of the secure Hamming distance. Here we describe in brief how the computation of secure Hamming distance is performed. The whole proof can be referred in Proposition 2 in [3].

We let two vectors $A = (A_0, A_1,..., A_n) \in \mathbb{Z}_2^n$ and $B = (B_0, B_1,..., B_n) \in \mathbb{Z}_2^n$ represent two biometric templates, each of length n. The Hamming distance between A and B in plain text is:

$$HD(A, B) = \sum_{i=0}^{n-1} A_i \otimes B_i = \sum_{i=0}^{n-1} (A_i + B_i - 2A_iB_i) \tag{3.1}$$

With the somewhat homomorphic encryption based on ideal lattices, the secure Hamming distance is calculated as follows:

$$ct_{HD} = C_2 * vEnc_1(A) + C_1 * vEnc_2(B) + (-2 * vEnc_1(A) * vEnc_2(B)) \in \mathbb{Z}_d \tag{3.2}$$

C_1 and C_2 are two integers defined below and constructing them would require extracting the value r and d from pk:

$$C_1 := \left[\sum_{i=0}^{n-1} r^i\right]_d \text{ and } C_2 := [-C_1 + 2]_d.$$

Note that if we decrypt ct_{HD} with the proper secret key, we will obtain $HD(A, B)$.

Secure inner product computation

The two packing methods can also lead to efficient computation of the inner product between two templates. Again, we let two vectors $A = (A_0, A_1,..., A_n) \in \mathbb{Z}_2^n$ and $B = (B_0, B_1,..., B_n) \in \mathbb{Z}_2^n$ represent two biometric templates, each of length n. The inner product between A and B in plain text is:

$$P(A, B) = \sum_{i=0}^{n-1} A_i B_i \tag{3.3}$$

The encrypted inner product is:

$$ct_P = vEnc_1(A) \cdot vEnc_2(B). \tag{3.4}$$

The complete proof of Equation 3.2 can be referred in Proposition 3 in [3].

3.3.2 The protocol

Besides the SHE construction, Yasuda et al. also proposed a biometric authentication protocol employing the SHE scheme [3] that targets one-to-one authentication (with respect to a specific $userID$). The protocol uses a distributed setting with three parties: a client server $\mathcal{C}$, a computation server $\mathcal{CS}$ attached to a database $\mathcal{DB}$, and an authentication server $\mathcal{AS}$. In addition, $\mathcal{AS}$ is assumed to be a trusted party that is responsible for key generations. The protocol is divided into three phrases:

Setup Phase: The authentication server $\mathcal{AS}$ runs $\mathsf{KeyGen}(\lambda)$ and generates the public key pk and the secret key sk of the somewhat homomorphic encryption (SHE) scheme. $\mathcal{AS}$ keeps the sk to itself and distributes pk to both the client server $\mathcal{C}$ and the computation server $\mathcal{CS}$.

Enrollment Phase.

1. Upon client registration, the client server $\mathcal{C}$ generates a 2048-bit feature vector A from the client's biometric data (e.g., fingerprints), runs $\mathbf{Enc}(A, pk)$ using the type 1 packing method (see Definition 2), and outputs the encrypted feature vector $vEnc_1(A)$.
2. The computation server $\mathcal{CS}$ stores the tuple $(ID, vEnc_1(A))$ in the database $\mathcal{DB}$. This tuple serves as the reference biometric template for the specific client with ID.

Authentication Phase.

1. The client provides fresh biometric data and the ID upon an authentication request, from which The client server $\mathcal{C}$ generates a feature vector B of 2048 bit. $\mathcal{C}$ runs $\mathbf{Enc}(B, pk)$ and encrypts the feature vector B with the type 2 packing method (note that it is asymmetric to the type 1 packing method, see Definition 2) and outputs $vEnc_2(B)$. Then $\mathcal{C}$ sends $(ID, vEnc_2(B))$ to the computation server $\mathcal{CS}$.
2. The computation server $\mathcal{CS}$ extracts the tuple $(ID, vEnc_1(A))$ from the database $\mathcal{DB}$ corresponding to the client to be authenticated (use ID as the search key). It calculates the encrypted Hamming distance ct_{HD} and sends ct_{HD} to the authentication server $\mathcal{AS}$.
3. The authentication server decrypts ct_{HD} and retrieves the actual Hamming distance $HD(A, B) = Dec(ct_{HD}, sk)$. Finally, the server $\mathcal{AS}$ returns 'AUTHENTICATION SUCCESS' if $HD(A, B) \leq \kappa$ or 'AUTHENTICATION FAILURE' if $HD(A,B) > \kappa$, where κ is a predefined threshold.

3.4 A template recovery attack

Abidin and Mitrokots [4] discovered a simple yet powerful attack in Yasuda et al.'s protocol [3] given the computation server $\mathcal{CS}$ is malicious and does incorrect computation. The attack leads to violation of data integrity and the consequence is severe: the reference biometric template will be leaked in plain text format if the attack is successful. The attack algorithm is shown in below as Algorithm 1.

input : $vEnc_1(A)$ (the stored encrypted template)
output : $A = A_1,...,A_N$ (the reference template)
initialize: $A = 0_1 0_2 ... 0_N$
for $i = 0$ **to** $N - \kappa$ **do**
  Set $A' = 1_1 ... 1_{\kappa+i} 0_{\kappa+i+1} ... 0_N$;
  Compute $ct_P = vEnc_2(A')$;
  Send $vEnc_1(A) \cdot vEnc_2(A')$ to $\mathcal{AS}$;
  if *rejected* **then**
    break
  end
  if i=N **then**
    return centerSearch(A);
  end
  Set $i' = \kappa + i$;
  Set $A_{i'} = 1$;
end
for $i = 1$ **to** $i' - 1$ **do**
  Set $A' = 1_1 ... 1_{i-1} 0_i 1_{i+1} ... 1_{i'} 0_{i'+1} ... 0_N$;
  Compute $vEnc_2(A')$;
  Send $vEnc_1(A) \cdot vEnc_2(A')$ to $\mathcal{AS}$;
  if *accepted* **then**
    $A_i = 1$;
  end
end
for $i = i' + 1$ **to** N **do**
  Set $A' = 1_1 ... 1_{i'-1} 0_{i'} ... 0_{i-1} 1_i 0_{i+1} ... 0_N$;
  Compute $vEnc_2(A')$;
  Send $vEnc_1(A) \cdot vEnc_2(A')$ to $\mathcal{AS}$;
  if *rejected* **then**
    $A_i = 1$;
  end
end
return A;

Algorithm 1: The template recovery attack algorithm [4].

The attack takes place in the authentication phase where the compromised computation server $\mathcal{CS}$ constructs a specially tailored trial vector A'. $\mathcal{CS}$ computes the inner

product $ct_P = vEnc_1(A) \cdot vEnc_2(A')$ rather than the hamming distance ct_{HD} as it is supposed to. The goal of this attack is to recover A from $vEnc_1(A)$ and it takes advantage of the homomorphic property that the decryption of ct_P will be $A \cdot A'$, the inner product of vector A and A'. To be more precise, A' is initialized as a vector of the same length as A with the first κ bits set to 1 and the rest bits set to 0. Recall that κ is the threshold value to compare the similarity between two biometric vectors. $\mathcal{CS}$ sends ct_P to the authentication server $\mathcal{AS}$. If $\mathcal{AS}$ accepts the outcome after decrypting ct_P, it implies that $\sum_{i=1}^{\kappa} A_i \leqslant \kappa$, i.e., the first κ bits in the vector A consist of both $0s$ and $1s$. Then $\mathcal{CS}$ can flip the $\kappa + 1$ bit of A' from 0 to 1 and send $vEnc_1(A) \cdot vEnc_2(A')$ to $\mathcal{AS}$. If the result is accepted again, $\mathcal{CS}$ flips the $\kappa + 2$ bit of A' and repeats the procedure until $\mathcal{AS}$ returns a rejection. At this stage $\mathcal{CS}$ has recovered a portion of the vector A that contains exactly κ number of $1s$, i.e. $\sum_{i=1}^{\kappa+k} A_i = \kappa$ where k is the number of trials. The steps mentioned previously correspond to the first *for* loops in the algorithm and is also visualized in Figure 3.2. Note that under a special case where the loop ends without being rejected, it implies that the reference template A has a Hamming weight less than κ. Abidin and Mitrokotsa pointed out that a center search attack can be mounted at this stage [4], which is depicted in Algorithm 2.

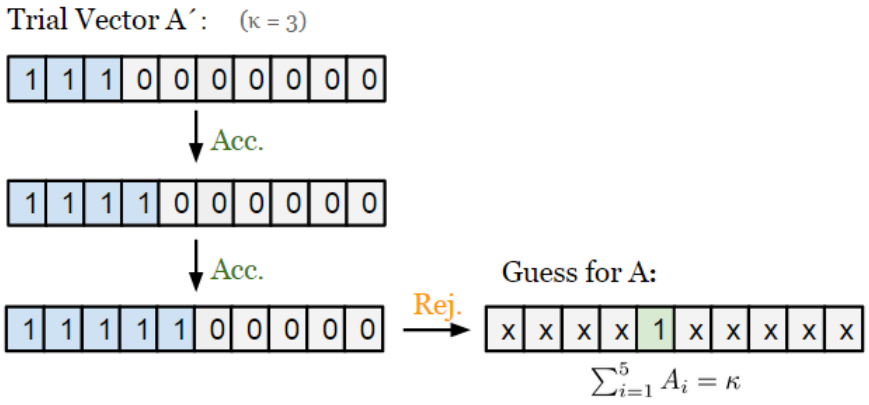

Figure 3.2: An example of the first loop of the template recovery attack.

input : $B = B_1,...,B_N$ (a fresh but matching template)
output: $A = A_1,...,A_N$ (the reference template)
for $i = 1$ **to** N **do**
  Set $B' = \bar{B_1},...,\bar{B_i},B_{i+1},...,B_N$;
  `/* The "bar" means flipping the bit, e.g.,` $B_1 = 0$, $\bar{B_1} = 1$. `*/`
  Send $vEnc_2(B')$ to $\mathcal{CS}$;
  if *rejected* **then**
    break
  end
end
for $i = 1$ *to* N **do**
  Send $vEnc_2(B'_1,...,\bar{B'_i},B'_{i+1},...,B'_N)$ to $\mathcal{CS}$;
  if *accepted* **then**
    $A_i = B_i$;
  end
  else
    $A_i = \bar{B_i}$;
  end
end
return A;

Algorithm 2: The center search attack [4].

We have determined one bit of A. Then the recovery can be divided into two halves: the left hand side and the right hand side of the determined bit. The two halves correspond to the second and third *for* loop in the algorithm and can be visualized in Figure 3.3 and Figure 3.4.

The cause of the Abidin attack is that the authentication server $\mathcal{AS}$ in Yasuda et al's protocol is not equipped with any mechanism to verify the correctness of the computation. It only decrypts the received output (not differentiating ct_{HD} or c_P), compares the decrypted value with κ and announces AUTHENTICATION SUCCESS or AUTHENTICATION FAILURE. Such check in theory can be repeated arbitrary amount of times with different input value. The authentication server $\mathcal{AS}$ is therefore exploited as a decrypting oracle. If these steps are repeated, the complete biometric data A can be learnt in at most $2N - \kappa$ steps, where N is the bit-length of the feature vector A and κ is the threshold value.

The attack will not be possible to execute if the authentication server $\mathcal{AS}$ detects that the incorrect function is computed. Because once $\mathcal{AS}$ detects that it is not the legitimate encrypted hamming distance that is sent over by the computation server, it will terminate the loop of trials.

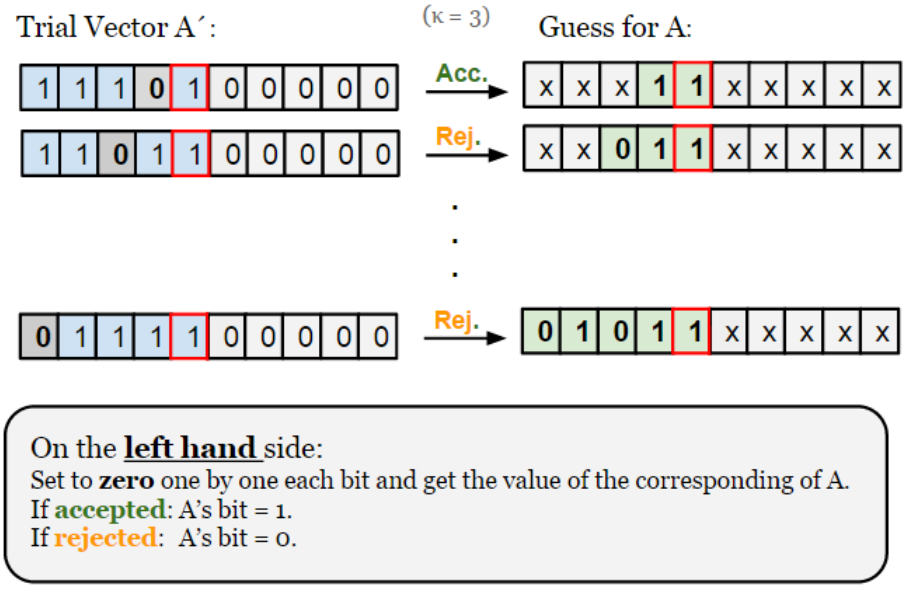

Figure 3.3: An example of the second loop of the template recovery attack.

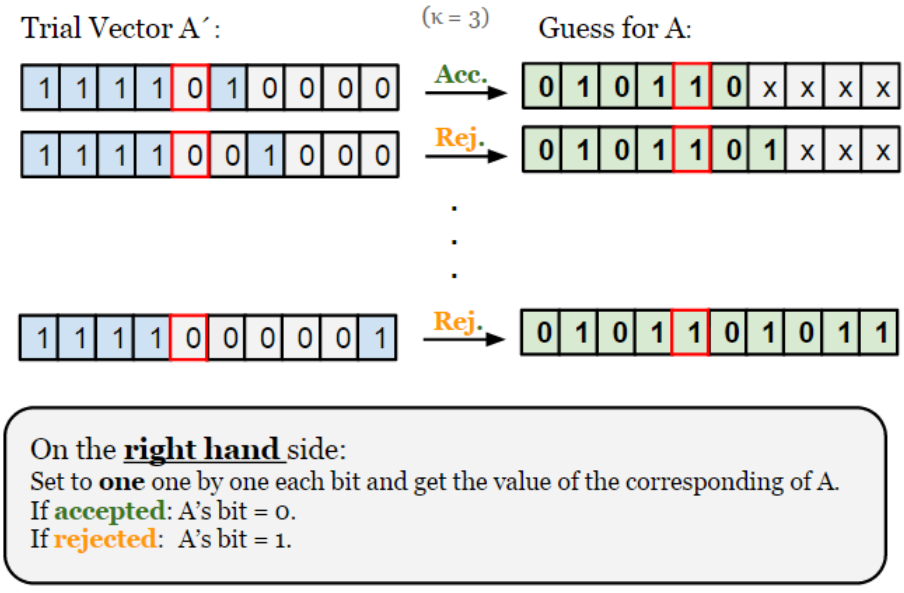

Figure 3.4: An example of the third loop of the template recovery attack.

4

Utilities

THIS chapter provides some mathematical tools that will be used in the instantiation of the verifiable privacy-preserving biometric authentication scheme to be presented in the result chapter.

4.1 Arithmetic circuits

Arithmetic circuits are very widely used algebraic tools for symbolizing polynomial computations. It is very useful in determining the most efficient way for computing a specific polynomial f.

The fundamental elements of arithmetic circuits include a set of variables $X = \{x_1, ..., x_n\}$ defined over a field $\mathbb{F}$, constant values and operators $(+, \times)$, and they form a directed acyclic graph. The notes in the graph are called *gates.* There are three types of gates: *input gates* where in-degree is 0; *output gates* where out-degree is 0, and last but not least *internal gates* with in-degree and out-degree greater than 0. An input gate could be either a *variable* or a *constant.* A variable is represented by a string and can hold any arbitrary value in the range of the pre-defined field $\mathbb{F}$, whilst a constant can only hold a fixed value $c \in \mathbb{F}$. Internal gates, on the other hand, are labelled with arithmetic operators either $+$ or $\times$. Gates marked by a $+$ sign are called the *sum gates*, which sum up the polynomials from the input wires. Similarly, gates marked by a $\times$ are called the *product gates* that calculate the products of polynomials from the input wires. The output of the whole arithmetic circuit (which corresponds to the evaluation of a polynomial) is obtained through the outgoing wire of the output gate. *The degree of a gate* is defined as the entire degree of the polynomials outputted by the gate.

After introducing the specific gate properties, we now describe some other definitions of an arithmetic circuit as a whole. The *size* of an arithmetic circuit is the total number of gates. The *depth* of the circuit is the length of the longest path from an input gate to an output gate. The *degree of the circuit* is the maximal degree of all gates in the entire

circuit [15]. Note that in the setting of the thesis, we will only consider circuits with single output gate and in-degree of each internal gate bounded by 2. In other words, we will only look at circuits that correspond to quadratic polynomials in the scheme instantiation.

4.2 Bilinear maps

The notion of bilinear maps are used in evaluating the authentication tags in the verifiable computation (VC) scheme. Generally speaking, a bilinear map is a function that takes elements from two spaces and then outputs a element in a new third vector space. Furthur more, an important requirement is that it is linear in each of its element.

Now we provide a more formal definition. Given two isomorphic groups $\mathbb{G}$ and $\mathbb{G}_T$ of order p where p is a large prime, there exists a map $e : \mathbb{G} \times \mathbb{G} = \mathbb{G}_T$. We say such a map e is bilinear if the following properties are satisfied [36]:

1. *Bilinear*: the condition $e(aP, bQ) = e(P, Q)^{ab}$ must hold for all $P, Q \in \mathbb{G}$ and all $a, b \in \mathbb{Z}$.

2. *Non-degenerate*: the map e cannot be a trivial map that sends all pairs in $\mathbb{G} \times \mathbb{G}$ to the identity in $\mathbb{G}_\mathbb{T}$. This property implies that if the generator of group $\mathbb{G}$ is g, then the generator for $\mathbb{G}_T$ will be $e(g,g)$.

3. *Computable*: There exists an efficient algorithm to compute $e(P, Q)$ for any P, Q in $\mathbb{G}$.

Note that an extra requirement for the bilinear map e in the setting of VC is that is should be cryptographically secure. Roughly speaking, this map works over groups where the discrete logarithm problem is assumed to be hard, and the function $e(,)$ is one-way, i.e., hard to invert.

Homomorphic Evaluation over Bilinear Groups

In this subsection we show how to perform homomorphic evaluation of arithmetic circuits bounded by degree 2 with bilinear maps. The construction of homomorphic MACs is based on this algorithm [15]. We first define a description of bilinear groups $bgpp = (p, \mathbb{G}, \mathbb{G}_T, e, g)$ where p is a large prime, $\mathbb{G}$ and $\mathbb{G}_T$ are two groups of order p, e is a bilinear map and g is the generator for $\mathbb{G}$. We see that $\mathbb{G}$ is isomorphic to the additive group $(\mathbb{Z}_p, +)$ if we consider $\phi_g(x) = g^x$ for all $x \in \mathbb{Z}_p$. For similar reason $\mathbb{G}_T$ is also isomorphic to $(\mathbb{Z}_p, +)$ if we consider $\phi_{gT}(x) = e(g,g)^x$ for all $x \in \mathbb{Z}_p$. In theory there exist the inverses $\phi_g^{-1} : \mathbb{G} \to \mathbb{Z}_p$ and $\phi_{gT}^{-1} : \mathbb{G}_T \to \mathbb{Z}_p$, but according to the cryptographic constraints these should not be efficiently computable.

As defined in [15], given an arithmetic circuit $f : \mathbb{Z}_p^n \to \mathbb{Z}_p$, there is an algorithm $\mathsf{GroupEval}(f, X_1,...,X_n)$ which homomorphically evaluates f from $\mathbb{G}$ to $\mathbb{G}_T$. It holds:

$$\mathsf{GroupEval}(f, X_1,...,X_n) = \phi_{gT}(f(\phi_g^{-1}(X_1),...,\phi_g^{-1}(X_n))). \tag{4.1}$$

Under the constraint that f has maximal degree 2 and $(X_1,...,X_n) \in \mathbb{G}^n$, GroupEval proceeds in a gate-by-gate manner through the arithmetic circuit. We show the various scenarios how the algorithm works as it passes through either an addition gate or a product gate as follows.

For an addition gate $(f,+)$ there are four cases:

- $X_1 \in \mathbb{G}$, $X_2 \in \mathbb{G}$, output $X = X_1 \cdot X_2 = g^{x_1} \cdot g^{x_2} = g^{x_1+x_2} \in \mathbb{G}$.
- $\hat{X_1} \in \mathbb{G}_T$, $\hat{X_2} \in \mathbb{G}_T$, output $\hat{X} = \hat{X_1} \cdot \hat{X_2} = e(g,g)^{x_1} \cdot e(g,g)^{x_2} = e(g,g)^{x1+x2} \in \mathbb{G}_T$.
- $\hat{X_1} \in \mathbb{G}_T$, $X_2 \in \mathbb{G}$, output $\hat{X} = \hat{X_1} \cdot e(X_2, g) = e(g,g)^{x1+x2} \in \mathbb{G}_T$.
- $X_1 \in \mathbb{G}$, $\hat{X_2} \in \mathbb{G}_T$, output $\hat{X} = e(X_1, g) \cdot \hat{X_2} = e(g,g)^{x1+x2} \in \mathbb{G}_T$.

For a product gate $(f, \times)$ with two variable inputs there is only one single case where $X_1, X_2 \in \mathbb{G}$. It is because doing a multiplication will ascend the computation from $\mathbb{G}$ to $\mathbb{G}_T$. Given the maximal degree of the circuit is 2, we can only take in two degree 1 variables as inputs, i.e., from group $\mathbb{G}$. Thus $X_1 \in \mathbb{G}$, $X_2 \in \mathbb{G}$, output $\hat{X} = e(X_1, X_2) = e(g,g)^{x_1 x_2} \in \mathbb{G}_T$.

For a product gate $(f, \times)$ with one variable and a constant, there is also only one single case regardless whether the variable is in $\mathbb{G}$ or $\mathbb{G}_T$: $X_1 \in \mathbb{G} \cup \mathbb{G}_T$, output $X = (X_1)^c = e(g,g)^{x_1 c} \in \mathbb{G}_T$.

The ultimate output of the GroupEval algorithm is the output of the last gate of the arithmetic circuit.

Backe et al. has proven the homomorphic property of the GroupEval algorithm by induction over the gates of f. The details can be referred to in Theorem 1 in [15].

5

Results

In this section we present the results of this thesis, which start with a generic scheme for verifiable privacy-preserving biometric authentication followed by its application in a protocol. Then we describe the actual construction of the scheme with correctness and security analysis. We also present a flawed approach in combing integity and privacy guarantees in a naive way. Finally we round up this chapter with a discussion.

5.1 The generic scheme

Let SHE = (SHE.KeyGen, SHE.Enc, SHE.Dec) be the somewhat homomorphic scheme used by Yasuda et al [3]. Moreover, let VC = (VC.KeyGen, VC.Auth, VC.Ver, VC.Eval) be the homomorphic message authenticator scheme by Backes et al. [15]. We describe a new scheme BVC= (**KeyGen, Enc, Auth, Comp, Eval, Ver**) in the following that is built upon the two schemes mentioned above (SHE and VC). BVC is tailored for the scenario of privacy-preserving biometric authentication system supporting verifiable computation.

In addition, because of the particular setting of biometric authentication, the circuit to be evaluated/computed is fixed, which is always the function f that computes the encrypted Hamming distance given two packed ciphetext of type 1 and type 2 respectively. Any other circuits will be treated as invalid computations. Thus, the circuit f is not passed in exclusively as parameters in the algorithms of the scheme. On the other hand, the public key pk for the encryption scheme appears in the parameter list for the last three algorithms, and the reason is that pk contains some parameters needed to complete defining the circuit (more specifically, to compute C_1 and C_2). The details can be found in the scheme construction.

KeyGenSHE$(\lambda) \rightarrow (pk, sk)$: Run SHE.KeyGen(λ) to generate the public key and the private key (pk, sk) for somewhat homomorphic encryption.

KeyGenVC$(\lambda_2) \rightarrow (sk_{vc}, ek)$: Run VC.KeyGen(λ) to generate the secret key and the evaluation key(sk_{vc}, ek) for verifiable computation.

Enc$(A, pk, phase) \rightarrow ct(A)$: Compute $ct(A) = \mathsf{SHE}.\mathrm{Enc}(A, pk)$ with the appropriate packing method. So $ct(A) = vEnc_1(A)$ (when $phase = 0$, "enrollment") or $ct(A) = vEnc_2(A)$ (when $phase = 1$, "authentication").

Auth$(sk_{vc}, L, ct(A)) \rightarrow \sigma$: Run VC.Auth$(sk_{vc}, L, ct(A))$ to authenticate $ct(A)$ given the secret key sk_{vc} and a label $L = (\Delta, \tau)$ (explained in Chapter 2). The authentication algorithm outputs a tag σ.

Comp$(vEnc_1(A), vEnc_2(B), pk) \rightarrow (ct_{HD}, \ell_d)$: Compute the encrypted Hamming distance ct_{HD} and a parameter ℓ_d given the two packed cipher texts and the circuit f.

Eval$(ek, \sigma_A, \sigma_B, pk) \rightarrow \sigma_{HD}$: Given the input of the evaluation key ek , the two tags σ_A, σ_B and f implicitly, the evaluation algorithm outputs a new tag σ_{HD} for the encrypted Hamming distance.

Ver$(sk_{vc}, sk, \mathcal{P}_\Delta, ct'_{HD}, \sigma_{HD}, pk) \rightarrow acc_{vc}, acc_{hd}$:

- Run VC.Ver$(sk_{vc}, \mathcal{P}_\Delta, ct'_{HD}, \sigma_{HD})$ given the secret key sk_{vc}, a multi-labelled program $\mathcal{P}_\Delta = ((f, \tau_A, \tau_B), \Delta)$, the received result of encrypted Hamming distance ct'_{HD} and a tag σ_{HD}. The acc_{vc} bit is set to either *0 (reject) or 1 (accept)* indicating if the computation is performed correctly. If $acc_{vc} = 1$ (so $ct'_{HD} = ct_{HD}$), the algorithm proceeds to the next step. Otherwise, the scheme terminates and outputs (0, 0) and a message 'INVALID COMPUTATION'.
- Compute $HD(A, B) = \mathsf{SHE}.\mathrm{Dec}(ct_{HD}, sk)$. The acc_{hd} bit is set to either 0 (reject) or 1 (accept) indicating the authentication result of the biometric data. The scheme terminates and outputs either (1, 1) for 'AUTHENTICATION SUCCESS' or (1, 0) for 'AUTHENTICATION FAILURE'.

5.2 The protocol

The protocol serves as an improvement to the original protocol using SHE scheme based on ideal lattices by Yasuda et al. [3]. The new protocol incorporates the feature of computation verification. It comprises four distributed parties: a client server $\mathcal{C}$, a computation server $\mathcal{CS}$, an authentication server $\mathcal{AS}$, and a database $\mathcal{DB}$. In the original protocol, the authentication server $\mathcal{AS}$ is assumed to be a trusted party who manages the secret key for SHE. In this protocol we preserve this assumption and furthermore assume the client server $\mathcal{C}$ and the database $\mathcal{DB}$ are also trusted parties. $\mathcal{C}$ is responsible

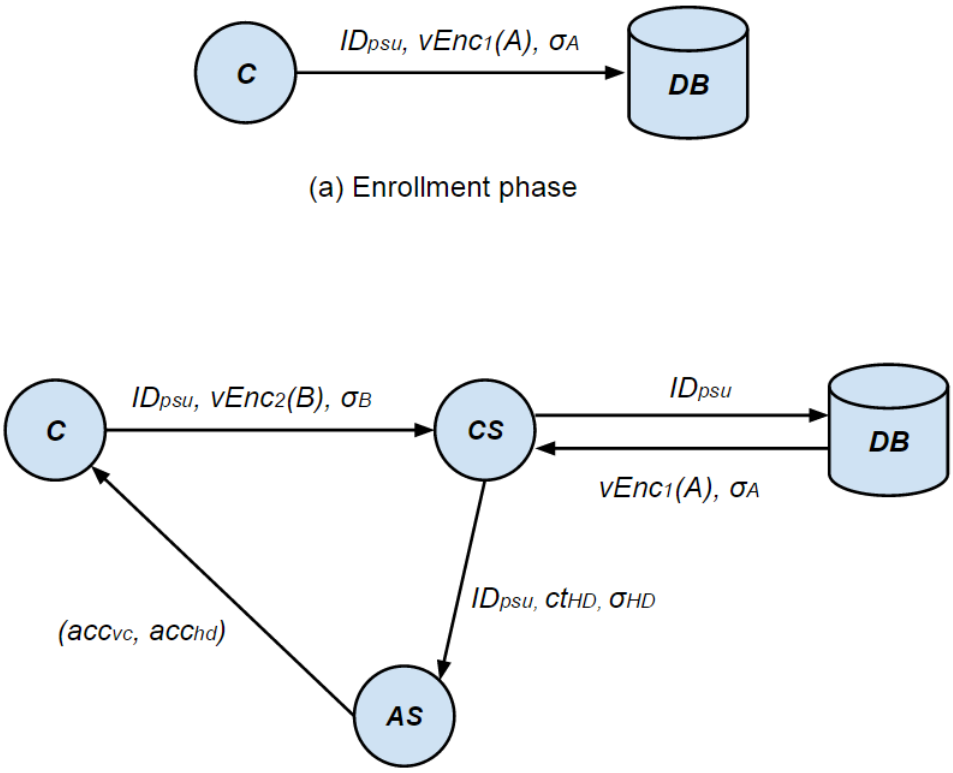

Figure 5.1: Modified Yasuda et al. biometric authentication scheme including verifiable computation [4].

to manage the secret key sk_{vc} for verifiable computation and $\mathcal{DB}$ stores the encrypted reference biometric templates with the pseudonyms of the corresponding clients. On the other hand, the computation server $\mathcal{CS}$ can be malicious and cheat with flawed computation. We describe the improved protocol in the following and illustrate the main phases in Figure 5.1. Moreover, we show which parties generate the various keys required in the protocol as well as the distribution of the keys in Table 5.1.

Key Name	Generated by	Owned by
pk (for SHE)	$\mathcal{AS}$	$\mathcal{AS}$, $\mathcal{C}$, $\mathcal{CS}$
sk (for SHE)	$\mathcal{AS}$	$\mathcal{AS}$
ek	$\mathcal{C}$	$\mathcal{C}$, $\mathcal{CS}$
sk_{vc}	$\mathcal{C}$	$\mathcal{C}$, $\mathcal{AS}$

Table 5.1: Keys Generation and Ownership.

Setup Phase.
The authentication server $\mathcal{AS}$ runs **KeyGenSHE**(λ) to generate the public key pk and the secret key sk of the somewhat homomorphic encryption (SHE) scheme. $\mathcal{AS}$ keeps the sk to itself and distributes pk to both the client server $\mathcal{C}$ and the computation server $\mathcal{CS}$.

Enrollment Phase.

1. Upon client registration, the client server $\mathcal{C}$ runs **KeyGenVC**(λ_2) to generate the evaluation key ek and the secret key sk_{vc} for verifiable delegation of computation. $\mathcal{C}$ distributes ek to the computation server $\mathcal{CS}$ and sk_{vc} to the authentication server $\mathcal{AS}$ (e.g., through a sensor).
2. The client server $\mathcal{C}$ generates a 2048-bit feature vector A from the client's biometric data (e.g., fingerprints), runs **Enc**$(A, pk, 0)$ using the type 1 packing method (see Definition 2), and outputs the encrypted feature vector $vEnc_1(A)$.
3. The client server $\mathcal{C}$ authenticates $vEnc_1(A)$ by running **Auth**$(sk_{vc}, L_A, vEnc_1(A))$ and outputs a tag σ_A. Then $\mathcal{C}$ sends the three-tuple $(ID_{pse}, vEnc_1(A), \sigma_A)$ to the database $\mathcal{DB}$. ID_{pse} is a pseudonym for the client to be enrolled. This three-tuple serves as the reference biometric template for the specific client with ID_{pse}.

Authentication Phase.

1. The client provides fresh biometric data upon an authentication request, from which The client server $\mathcal{C}$ generates a feature vector B of 2048 bit. $\mathcal{C}$ runs **Enc**$(B, pk, 1)$ and encrypts the feature vector B with the type 2 packing method (note that it is asymmetric to the type 1 packing method, see Definition 2) and outputs $vEnc_2(B)$.
2. The client server $\mathcal{C}$ authenticates $vEnc_2(B)$ by running **Auth**$(sk_{vc}, L_B, vEnc_2(B))$ and outputs a tag σ_B. Then $\mathcal{C}$ sends $(ID_{pse}, vEnc_2(B), \sigma_B)$ to the computation server $\mathcal{CS}$.
3. The computation server $\mathcal{CS}$ extracts the tuple $(ID_{pse}, vEnc_1(A), \sigma_A)$ from the database $\mathcal{DB}$ corresponding to the client to be authenticated (use ID_{pse} as the search key). It calculates the encrypted Hamming distance ct_{HD} by running **Comp**$(vEnc_1(A), vEnc_2(B), pk)$ and generates an authentication tag σ_{HD} by running **Eval**$(ek, \sigma_A, \sigma_B, pk)$. Then $\mathcal{CS}$ sends $(ID_{pse}, ct_{HD}, \sigma_{HD})$ to the authentication server $\mathcal{AS}$.

4. The authentication server $\mathcal{AS}$ runs $\mathbf{Ver}(sk_{vc}, sk, \mathcal{P}_\Delta, ct'_{HD}, \sigma_{HD}, sk)$ to perform verification. First it verifies if the computation done by the computation server $\mathcal{CS}$ is legitimate, i.e. the correct encrypted Hamming distance is returned. If the computation verification fails, the result 'INVALID COMPUTATION' is returned (indicated by the acceptance bits $(0, 0)$) . Otherwise, $\mathcal{AS}$ proceeds to decrypt ct_{HD} and retrieves the actual Hamming distance $HD(A, B)$. Finally, the server $\mathcal{AS}$ returns 'AUTHENTICATION SUCCESS' if $HD(A, B) \leq \kappa$ or 'AUTHENTICATION FAILURE' if $HD(A, B) > \kappa$, where κ is a predefined threshold. The acceptance bits to be returned are $(1, 1)$ respectively $(1, 0)$.

5.3 Construction

This section gives the detailed construction of our BVC scheme. For the ease of reading, the output of each algorithm is underlined.

KeyGenSHE$(\lambda) \rightarrow (pk,sk)$:

- The first step is to generate the public key and the secret key (pk,sk) for the somewhat homomorphic encryption (SHE) scheme. From the security parameter λ, we generate three parameters (n,s,t). $n = 2^m$ is a 2-power integer for $m \in \mathbb{Z}$ denoting the dimension of lattice L. s is the plain text space size. Then we define $R := \mathbb{Z}[x]/(f_n(x))$ denoting a polynomial ring modulo $f_n(x) := x^n + 1$ and choose an n-dimensional vector $\mathbf{v} = (v_0, v_1,..., v_{n-1}) \in \mathbb{Z}^n$ where v_i is chosen randomly and satisfies $|v_i| \leq 2^t$. We set $v(x) = \sum_{i=0}^{n-1} v_i x^i \in R$ as a generating polynomial and the parameter t is the bit length of the coefficients of $v(x)$.
- Using the extended Euclidean greatest common divisor (gcd) algorithm for polynomials, we calculate $w(x)$, the scaled inverse of $v(x)$ modulo $f_n(x)$ and the following equation should be satisfied:

$$w(x) \times v(x) \equiv d \mod f_n(x)$$

  d is the resultant of $v(x)$ and $f_n(x)$, which also equals to the determinant of the lattice L.
- We need to check two conditions to ensure that the vector $\mathbf{v}$ is qualified. Note that both conditions should be fulfilled.

  Condition I: d and s must be relatively prime, i.e. $gcd(d,s) = 1$. If not, the cipher text can be decrypted without the secret key.

  Condition II: Given $w(x) = \sum_{i=1}^{n} w_i x^i$, we check that $r := (w_1 * w_0^{-1}) \mod d$ should satisfy $r^n \equiv -1 \mod d$. If so, the lattice L generated by $v(x)$ will contain

a vector of the form $(-r,1,0,\ldots,0)$. The details can be referred in **Definition 1** in [3].

If both conditions are fulfilled, we proceed to the next step. Otherwise a new vector $\mathbf{v}$ shall be generated and the test for checking both conditions is rerun.

– We set $\underline{pk = (d, r, n, s)}$ and $\underline{sk = \hat{w}}$, where $\hat{w}$ is chosen among the coefficients of $w(x)$ satisfying $gcd(\hat{w}, s) = 1$.

KeyGenVC$(\lambda_2) \rightarrow (sk_{vc}, ek)$:

– Next we generate the keys (sk_{vc}, ek) for verifiable computation (VC). Given the security parameter λ_2, we generate the description of bilinear groups $bgpp = (p, \mathbb{G}, \mathbb{G}_{\mathbb{T}}, e, g)$. Let the message space $\mathcal{M}$ be $\mathbb{Z}_p$.

– We select a random value θ uniformly from $\mathbb{Z}_p$, run a key generation to obtain random (K, pp), where K is the seed of a pseudo-random function (PRF) $F_K : \{0,1\}^* \times \{0,1\}^* \rightarrow \mathbb{G}$ and pp is some public parameters that specify the domain $\mathcal{X}$ and the range $\mathcal{R}$ of the PRF.
We set $\underline{sk_{vc} = (bgpp, pp, K, \theta)}$ and $\underline{ek = (bgpp, pp)}$.

Enc$(A, pk, phase) \rightarrow ct(A)$:

– Given the assumption that the plain text A to be encrypted is a 2048-bit feature vector of biometric data, we shall apply the appropriate packing method to first transform the vector into a polynomial and then proceed with the encryption. Recall there are two types of packing methods (see Definition 2):

i. If A is provided in the **enrollment phase** as a reference template (indicated by $phase = 0$), then the type 1 packing method should be applied. We compute

$$F_1 : A = (A_0, \ldots, A_{2047}) \mapsto \sum_{i=0}^{2047} A_i x^i \in R = \mathbb{Z}[x]/(f_n(x)).$$

$$ct(A) = vEnc_1(A) = [F_1(A)(r) + su_1(r)]_d = \left[\sum_{i=0}^{2047} A_i r^i + su_1(r)\right]_d \in \mathbb{Z}_d.$$

ii. If A is provided in the **authentication phase** as a fresh biometric feature vector to be compared with a stored template (indicated by $phase = 1$), then the type 2 packing method should be applied. We compute

$$F_2 : A = (A_0, \ldots, A_{2047}) \mapsto -\sum_{i=0}^{2047} A_i x^{n-i} \in R = \mathbb{Z}[x]/(f_n(x)).$$

$$ct(A) = vEnc_2(A) = [F_2(A)(r) + su_2(r)]_d = \left[-\sum_{i=0}^{2047} A_i r^{n-i} + su_2(r)\right]_d \in \mathbb{Z}_d.$$

Note that $u_1(x)$ and $u_2(x)$ denote noise polynomials. The coefficients of the polynomail can be seen as a random vector constructed as $\mathbf{u} = (u_0, u_1, \ldots, u_{n-1})$ with $u_i \in \{0, \pm 1\}$. u_i has q probability being 0, $(1-q)/2$ probability being 1 and $(1-q)/2$ probability being -1.

– Finally we output the encrypted packed cipher text $ct(A)$.

Auth$(sk_{vc}, L, ct(A)) \rightarrow \sigma$:

– $ct(A)$ is the message to be authenticated and $L = (\Delta, \tau)$ is the corresponding multi-label, in which Δ is the *data set identifier* (e.g., the client's pseudonym) and τ is the *input identifier* (e.g., "stored biometric template" or "fresh biometric template"). Regarding data ranges, Δ and τ can be two arbitrary strings, i.e.,$\Delta \in \{0,1\}^\lambda$ and $\tau \in \{0,1\}^\lambda$.

– Given $sk_{vc} = (bgpp, pp, K, \theta)$, we compute $R = F_K(\Delta, \tau)$ and $(y_0, Y_1) \in \mathbb{Z}_p \times \mathbb{G}$, where we set

$$y_0 = ct(A) \text{ and } Y_1 = (R \cdot g^{-ct(A)})^{1/\theta}.$$

And then we output the authentication tag $\sigma = (y_0, Y_i)$.

Comp$(vEnc_1(A), vEnc_2(B), pk) \rightarrow (ct_{HD}, \ell_d)$:

– Recall Equation 3.2, the encrypted Hamming distance is calculated as:

$$ct_{HD} = C_2 * vEnc_1(A) + C_1 * vEnc_2(B) + (-2 * vEnc_1(A) * vEnc_2(B)) \in \mathbb{Z}_d,$$

where C_1 and C_2 are two integers defined below and constructing them would require extracting the value r and d from pk:

$$C_1 := \left[\sum_{i=0}^{n-1} r^i\right]_d \text{ and } C_2 := [-C_1 + 2]_d.$$

– The purpose of this step is to solve the range problem, which is described in detail in Section 5.4. Let c be the result of the Hamming distance without the final modulo d. Given

$$c' = c \mod p \text{ and } c = lp + c',$$
where c' is a component in the authentication tag and l is a dividend.

We compute $\ell_d = l \mod d = (c - [c \mod p])/p \mod d$.

– If the computation is performed correctly, the encrypted Hamming distance ct_{HD} and ℓ_d shall be returned.

Eval$(ek, \sigma_A, \sigma_B, pk) \rightarrow \sigma_{HD}$:

– The evaluation is performed homomorphically. The inputs include the evaluation key $ek = (bgpp, pp)$, the two tags σ_A and σ_B, and implicitly the arithmetic circuit f for calculating the encrypted Hamming distance (require public parameters from pk).

– The circuit is evaluated gate-by-gate. Every input gate accepts either two tags σ_A, $\sigma_B \in (\mathbb{Z}_p \times \mathbb{G} \times \mathbb{G}_T)^2$, or one tag and a constant σ, $c \in ((\mathbb{Z}_p \times \mathbb{G} \times \mathbb{G}_T) \times \mathbb{Z}_p)$. The output of a gate is a new tag $\sigma' \in (\mathbb{Z}_p \times \mathbb{G} \times \mathbb{G}_T)$, which will be fed into the next gate in the circuit as one of the two inputs. The operation stops when the final gate of f is reached and the resulting tag σ_{HD} shall be returned.

– We let a tag has the format $\sigma^{(i)} = (y_0^{(i)}, Y_1^{(i)}, \hat{Y}_2^{(i)}) \in \mathbb{Z}_p \times \mathbb{G} \times \mathbb{G}_T$ for $i = 1, 2$ (indicating the two input tags), which correspond respectively to the coefficients of (x^0,x^1,x^2) in a polynomial. If $\hat{Y}_2^{(i)}$ is not defined, it is assumed that it has value $1 \in \mathbb{G}_T$. Note that the two tags σ_A and σ_B fall under the case because they have degree up to 1. Next we define the specific operation for different types of gates:

- **Addition.** The output tag $\sigma' = (y_0,Y_1,\hat{Y}_2)$ is calculated as:
$$y_0 = y_0^{(1)} + y_0^{(2)}, \quad Y_1 = Y_1^{(1)} \cdot Y_1^{(2)}, \quad \hat{Y}_2 = \hat{Y}_2^{(1)} \cdot \hat{Y}_2^{(2)}.$$
- **Multiplication.** The output tag $\sigma' = (y_0,Y_1,\hat{Y}_2)$ is calculated as:
$$y_0 = y_0^{(1)} \cdot y_0^{(2)}, \quad Y_1 = Y_1^{(1)} \cdot Y_1^{(2)}, \quad \hat{Y}_2 = e(\hat{Y}_1^{(1)},\hat{Y}_1^{(2)}).$$
Note that since the circuit f has maximum degree 2, The two input tags to a multiplication gate can only have maximum degree 1 each. i.e. $\hat{Y}_2^{(1)} = \hat{Y}_2^{(2)} = 1$.
- **Multiplication with constant.**The two inputs are one tag σ and one constant $c \in \mathbb{Z}_p$.The output tag $\sigma' = (y_0,Y_1,\hat{Y}_2)$ is calculated as:
$$y_0 = c \cdot y_0^{(1)}, \quad Y_1 = (Y_1^{(1)})^c, \quad \hat{Y}_2 = (\hat{Y}_2^{(1)})^c.$$

We output the final tag $\sigma_{HD} = (y_0, Y_1, \hat{Y}_2)$.

Ver$(sk_{vc}, sk, \mathcal{P}_\Delta, ct'_{HD}, \sigma_{HD}, \ell_d, pk) \rightarrow (acc_{vc}, acc_{hd})$:

– The first step is to verify **computation integrity check**, i.e. the encrypted Hamming distance has been calculated correctly. $\mathcal{P}_\Delta$ is a multi-labeled program defined by Backes et al. [15]. Δ is the *data set identifier* and in $\mathcal{P} = (f,\tau_A,\tau_B)$, f is the arithmetic circuit for calculating ct_{HD} and τ is the *input identifier* . We have ct'_{HD} as the "claimed to-be" encrypted Hamming distance to be verified and $\sigma_{HD} = (y_0,Y_1,\hat{Y}_2)$ is the corresponding tag. We compute:

$$R_A = F_K(\Delta, \tau_A); \quad R_B = F_K(\Delta, \tau_B).$$
$$W = \mathsf{GroupEval}(f,R_1,R_2) \in \mathbb{G}_T$$

Note that GroupEval (see Equation 4.1) is an algorithm that homomorphically evaluates a circuit f over bilinear maps.
We can then check the following two conditions. If both checks are passed, the acc_{vc} bit is set to 1 and the scheme proceeds to the next step. Otherwise the algorithm returns $(acc_{vc}, acc_{hd}) = (0, 0)$ and message 'INVALID COMPUTATION'.

$$ct_{HD} = y_0 \mod d + \ell_d \tag{5.1}$$
$$W = e(g,g)^{y_0} \cdot e(Y_1,g)^{\theta} \cdot (\hat{Y}_2)^{\theta^2} \tag{5.2}$$

Note that in the original VC scheme in [15], the first checking equation is $ct_{HD} = y_0$ rather than equation (5.1). The reason of this modification is the different ring ranges of ct_{HD} and y_0. A detailed explanation and the definition of ℓ_d can be found in Section 5.4 Ring range problem.

– The next step is to perform the **biometric authentication check**. The actual Hamming distance $HD(A, B)$ is recovered from decrypting ct'_{HD} (which is equal to ct_{HD} since the computation has passed the check). We compute:

$$HD(A,B) = [ct_{HD} \cdot sk]_d \mod s.$$

Let κ be the pre-defined threshold for authenticating biometric data.

i. If $HD(A,B) \leq \kappa$, the message 'AUTHENTICATION SUCCESS' is returned, indicated by $\underline{(acc_{vc}, acc_{hd}) = (1, 1)}$.

ii. If $HD(A,B) > \kappa$, the message 'AUTHENTICATION FAILURE' is returned, indicated by $\underline{(acc_{vc}, acc_{hd}) = (1, 0)}$.

5.4 Ring range problem

The most significant challenge in combining the VC scheme based on homomorphic MAC with the SHE scheme is the range of the base rings. Even though the function types match perfectly (quadratic polynomials), the base ring ranges chosen in these schemes are different. While the VC scheme proposed by Backes et al. [15] handles all operations in $\mathbb{Z}_p$ where p is a prime, the operations in Yasuda et al.'s SHE scheme are handled in $\mathbb{Z}_d$ where d is the resultant of two polynomials. Moreover, the VC scheme defines the reduction of z modulo p in the interval $[0,p)$ and the SHE scheme defines the reduction of z modulo d in the interval $[-d/2, d/2)$. Therefore in our BVC we need to tweak the input data if there is a mismatch in the ranges. Similarly, the range factor should be taken into consideration in performing final verification.

First we show how to tweak the input data. In the formula for calculating the Hamming distance, there is a constant term -2, which lives in $[-d/2,d/2)$ but not in $[0, p)$. In order to verify and generate proper tags, we can write -2 as $D = (d - 2) \mod p$. Note that the -2 will be re-written as D in the proof of correctness.

Then we need to check the impact of the range difference to the verification carried out by the client. The first equation the client needs to check in the scheme of Backes et al. [15] is

$$ct_{HD} = y_0^{(HD)}, \text{ where } ct_{HD} \in \mathbb{Z}_d \text{ and } y_0^{(HD)} \in \mathbb{Z}_p.$$

ct_{HD} is the encrypted Hamming distance calculated by the computation server $\mathcal{CS}$ and $y_0^{(HD)}$ is a component of the final authentication tag. If $d = p$, this check will succeed as long as $\mathcal{CS}$ performs the correct computation. But the case of $d = p$ is very unlikely, and the difference in the reduction has a direct impact on the equality of the verification equation (e.g., $(10 \mod 5 = 0) \neq (10 \mod 7 = 3)$).

The actual value of p and d depends on the implementation. We present a solution on a general level regardless of implementation and specific value of p and d. Thus we assume that $p < d$ (as the tag size should be ideally small). Nevertheless, the solution also applies when $p > d$ and it would require swapping the place of p and d.

The solution relies on keeping track of the dividend. Given we have the encrypted stored template $\alpha \in \mathbb{Z}_d$ and encrypted fresh template $\beta \in \mathbb{Z}_d$, we have

$$\alpha = \alpha' + mp, \alpha' = \alpha \mod p \in \mathbb{Z}_p.$$
$$\text{Similarly, } \beta = \beta' + kp, \beta' = \beta \mod p \in \mathbb{Z}_p$$

Assume $SWHD(x,y)$ is the arithmetic circuit for calculating the encrypted Hamming distance without the final modulo d. Let $c = SWHD(\alpha, \beta)$ and $c' = SWHD(\alpha', \beta')$ We can derive

$$SWHD(\alpha, \beta) \mod p = SWHD(\alpha', \beta') \mod p;$$
$$c = l * p + c'.$$

The value l is the dividend. However, in order to perform the comparison we would track $l \mod d$ instead of tracking l directly. The reason is that l contains more information. Relating back to comparing ct_{HD} and $y_0^{(HD)}$, we have:

$$ct_{HD} = c \mod d \in \mathbb{Z}_d;$$
$$y_0^{(HD)} = c' \in \mathbb{Z}_p$$

Given $c = l * p + c'$,

$$\begin{aligned} ct_{HD} &= c \mod d \\ &= (l * p + c') \mod d \\ &= c' \mod d + (l \mod d) * (p \mod d) \\ &= y_0^{(HD)} \mod d + (l \mod d) * (p \mod d) \end{aligned} \tag{5.3}$$

Thus, if we define $\ell_d = (l \mod d) * (p \mod d)$, the verification equation will be

$$ct_{HD} = y_0^{(HD)} \mod d + \ell_d$$

5.5 Correctness analysis

Backes et al.[15] proved the correctness of the VC scheme in a general way using proofs by induction. However, in our case of study, the scheme (BVC) makes use of a deterministic quadratic function and is described by only two variables. This relative simple setting allows us to write the correctness proof by walking through the arithmetic circuit step by step.

Figure 5.2 depicts the arithmetic circuit for calculating the encrypted Hamming distance. A and B denote the encrypted stored and fresh biometric templates respectively. C_1 and C_2 are the constants in the function as defined in the **Comp** algorithm. D should indicate the -2 in the function, but since -2 is not in the valid range $\mathbb{Z}_p$ required by the original VC scheme, we need to have an intermediate transformation of $D = d - 2$. All A, B, C_1 and C_2 are in $\mathbb{Z}_d$. Finally, the σs are the outcome tags of the form $\sigma^{(i)} = (y_0^{(i)}, Y_1^{(i)}, \hat{Y}_2^{(i)}) \in \mathbb{Z}_p \times \mathbb{G} \times \mathbb{G}_T$ after each gate operation, and the Rs are values in either $\mathbb{G}$ or $\mathbb{G}_\mathbb{T}$, which are used for homomorphic evaluation over bilinear groups (The GroupEval algorithm in [15]) (see Equation 4.1).

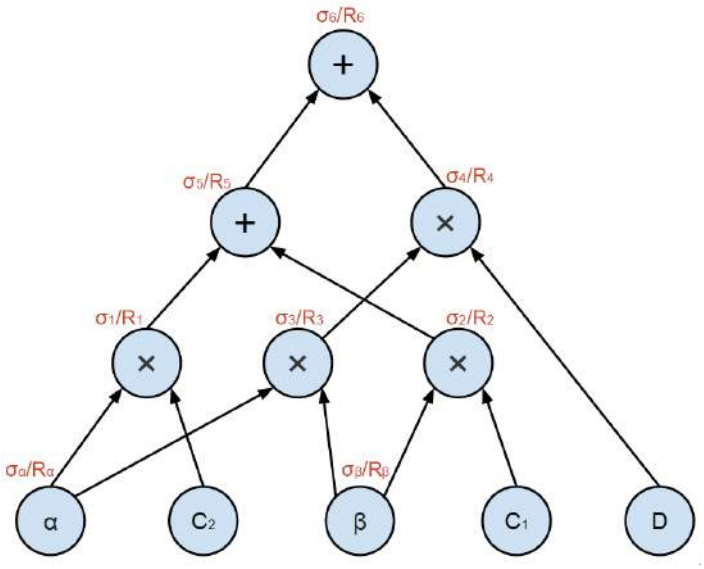

Figure 5.2: The arithmetic circuit for calculting the encrypted Hamming distance.

For the simplicity to show the calculation, we let α and β be $vEnc_1(A)$ and $vEnc_1(B)$ and they each has a tag where $\sigma_\alpha = (y_0^{(A)}, Y_1^{(A)}, 1)$ and $\sigma_\beta = (y_0^{(B)}, Y_1^{(B)}, 1)$. These two tags are generated by the **Auth** algorithm, which specifies that $y_0^{(A)} = \alpha$ and $Y_1^{(A)} = (R_\alpha \cdot g^{-\alpha})^{1/\theta}$. Similarly, we have $y_0^{(B)} = \beta$ and $Y_1^{(B)} = (R_\beta \cdot g^{-\beta})^{1/\theta}$.

To verify the correctness of our BVC scheme, we need to check that the two equations specified in the **Ver** algorithm (5.1) and (5.2) are satisfied if the computation is performed correctly. Let $\sigma_{HD} = (y_0^{(HD)}, Y_1^{(HD)}, \hat{Y}_2^{(HD)})$ be the final tag (which is equivalent to σ_6 in the arithmetic circuit depicted in Figure 5.2). To recall, the verification equations (5.1) and (5.2) are:

$$ct_{HD} = y_0 \mod d + \ell_d;$$
$$W = \mathsf{GroupEval}(f, R_\alpha, R_\beta) = e(g,g)^{y_0^{HD}} \cdot e(Y_1^{HD}, g)^\theta \cdot (\hat{Y}_2^{(HD)})^{\theta^2}.$$

The first step is to derive the tags for the intermediate calculation and eventually the final tag. If we run the **Eval** algorithm homomorphically through the circuit, we will get the following outcome tags $\sigma_1, \ldots, \sigma_6$.

$$\sigma_1 = (C_2 \cdot y_0^{(A)}, (Y_1^{(A)})^{C_2}, 1);$$
$$\sigma_2 = (C_1 \cdot y_0^{(B)}, (Y_1^{(B)})^{C_1}, 1);$$
$$\sigma_3 = (y_0^{(A)} \cdot y_0^{(B)}, (Y_1^{(A)})^{y_0^{(B)}} \cdot (Y_1^{(B)})^{y_0^{(A)}}, e(Y_1^{(A)}, Y_1^{(B)}));$$
$$\sigma_4 = (D \cdot y_0^{(A)} \cdot y_0^{(B)}, (Y_1^{(A)})^{y_0^{(B)} \cdot D} \cdot (Y_1^{(B)})^{y_0^{(A)} \cdot D}, e(Y_1^{(A)}, Y_1^{(B)}))^D;$$

$\sigma_5 = (C_2 \cdot y_0^{(A)} + C_1 \cdot y_0^{(B)}, (Y_1^{(A)})^{C_2} \cdot (Y_1^{(B)})^{C_1}, 1);$

$\sigma_6 = (C_2 \cdot y_0^{(A)} + C_1 \cdot y_0^{(B)} + D \cdot y_0^{(A)} \cdot y_0^{(B)}, (Y_1^{(A)})^{y_0^{(B)} \cdot D + C_2} \cdot (Y_1^{(B)})^{y_0^{(A)} \cdot D + C_1}, e(Y_1^{(A)}, Y_1^{(B)})^D);$

We let σ_{HD} denote the final authentication tag. Thus we have $\sigma_{HD} = (y_0^{(HD)}, Y_1^{(HD)}, \hat{Y}_2^{(HD)}) = \sigma_6$. For reading simplicity, we can write down each component of σ_{HD} separately as follows:

$y_0^{(HD)} = C_2 \cdot y_0^{(A)} + C_1 \cdot y_0^{(B)} + D \cdot y_0^{(A)} \cdot y_0^{(B)};$

$Y_1^{(HD)} = (Y_1^{(A)})^{y_0^{(B)} \cdot D + C_2} \cdot (Y_1^{(B)})^{y_0^{(A)} \cdot D + C_1};$

$\hat{Y}_2^{(HD)} = e(Y_1^{(A)}, Y_1^{(B)})^D.$

The next step is to run $GroupEval(f, R_\alpha, R_\beta)$ and execute the bilinear gate operations. Recall that R_α and R_β correspond to R_A and R_B in the notation used in the construction, which are the results of the pseudo-random function in the Ver algorithm. Similar to the tag derivation, we show the intermediate calculations and the final result of the R value, as depicted in Figure 5.2.

$R_1 = R_\alpha^{C_2};$

$R_2 = R_\beta^{C_1};$

$R_3 = e(R_\alpha, R_\beta);$

$R_4 = e(R_\alpha, R_\beta)^D;$

$R_5 = R_\alpha^{C_2} \cdot R_\beta^{C_1};$

$R_6 = e(R_\alpha^{C_2} \cdot R_\beta^{C_1}, g) \cdot e(R_\alpha, R_\beta)^D.$

R_6 is the final result of running *GroupEval* over the arithmetic circuit. By the definition of W given in equation (5.2), we have:

$$W = R_6 = e(R_\alpha^{C_2} \cdot R_\beta^{C_1}, g) \cdot e(R_\alpha, R_\beta)^D.$$

Now we show the proofs for the two verification equations. First we need to prove Equation (5.1) : $ct_{HD} = y_0 \mod d + \ell_d$. This equation has been proved in section 5.4 Ring range problem with equation (5.3). The end result is

$$ct_{HD} = y_0^{(HD)} \mod d + (l \mod d) * (p \mod d).$$

As we define $\ell_d = (l \mod d) * (p \mod d)$, we can derive equation (5.1).

Then we prove Equation (5.2) : $W = GroupEval(f, R_\alpha, R_\beta) = e(g,g)^{y_0^{HD}} \cdot e(Y_1^{HD}, g)^\theta \cdot (\hat{Y}_2^{(HD)})^{\theta^2}$. The approach is to start with the righthand side of the equation. There are

in total three factors. We in turn expand each one of the factors and finally compute the product of the results, evaluating it against W.

The first factor can be expanded as:

$$\begin{aligned} e(g,g)^{y_0^{HD}} &= e(g,g)^{C_2 \cdot y_0^{(A)} + C_1 \cdot y_0^{(B)} + D \cdot y_0^{(A)} \cdot y_0^{(B)}} \\ &= e(g,g)^{C_2\alpha + C_1\beta + \alpha\beta D}. \end{aligned} \tag{5.4}$$

The second factor is expanded as:

$$\begin{aligned} e(Y_1^{HD}, g)^{\theta} &= e((Y_1^{(A)})^{y_0^{(B)} \cdot D + C_2} \cdot (Y_1^{(B)})^{y_0^{(A)} \cdot D + C_1}, g)^{\theta} \\ &= e((R_\alpha \cdot g^{-\alpha})^{(\beta D + C_2)/\theta} \cdot (R_\beta \cdot g^{-\beta})^{(\alpha D + C_1)/\theta}, g)^{\theta} \\ &= e((R_\alpha \cdot g^{-\alpha})^{\beta D + C_2} \cdot (R_\beta \cdot g^{-\beta})^{\alpha D + C_1}, g) \\ &= e(R_\alpha^{\beta D + C_2} \cdot R_\beta^{\alpha D + C_1} \cdot g^{-2\alpha\beta D - \alpha C_2 - \beta C_1}, g) \\ &= e(R_\alpha, g)^{\beta D + C_2} \cdot e(R_\beta, g)^{\alpha D + C_1} \cdot e(g,g)^{-2\alpha\beta D - \alpha C_2 - \beta C_1}. \end{aligned} \tag{5.5}$$

The third factor is expanded as:

$$\begin{aligned} (\hat{Y}_2^{(HD)})^{\theta^2} &= e(Y_1^{(A)}, Y_1^{(B)})^{D\theta^2} \\ &= e((R_\alpha \cdot g^{-\alpha})^{1/\theta}, (R_\beta \cdot g^{-\beta})^{1/\theta})^{D\theta^2} \\ &= e(R_\alpha \cdot g^{-\alpha}, R_\beta \cdot g^{-\beta})^{D} \\ &= e(R_\alpha, R_\beta \cdot g^{-\beta})^{D} \cdot e(g^{-\alpha}, R_\beta \cdot g^{-\beta})^{D} \\ &= e(R_\alpha, R_\beta)^{D} \cdot e(R_\alpha, g)^{-\beta D} \cdot e(R_B, g)^{-\alpha D} \cdot e(g,g)^{\alpha\beta D}. \end{aligned} \tag{5.6}$$

Finally we calculate the product of the three factors by using the freshly derived equations (5.4), (5.5) and (5.6). To simplify the visualization of the equation, we create a temporary variable and let $P = e(g,g)^{y_0^{HD}} \cdot e(Y_1^{HD}, g)^{\theta} \cdot (\hat{Y}_2^{(HD)})^{\theta^2}$. The last equality sign in equation (5.7) proves the correctness of the second verification equation (5.2). Thus by far we have proved the correctness of the BVC scheme.

$$\begin{aligned} P &= e(g,g)^{C_2 \cdot \alpha + C_1 \cdot \beta + D \cdot \alpha \cdot \beta} \cdot e(R_\alpha, g)^{\beta D + C_2} \cdot e(R_\beta, g)^{\alpha D + C_1} \cdot e(g,g)^{-2\alpha\beta D - \alpha C_2 - \beta C_1}. \\ &\quad e(R_\alpha, R_\beta)^{D} \cdot e(R_\alpha, g)^{-\beta D} \cdot e(R_B, g)^{-\alpha D} \cdot e(g,g)^{\alpha\beta D} \\ &= e(g,g)^{C_2\alpha + C_1\beta + \alpha\beta D - 2\alpha\beta D - \alpha C_2 - \beta C_1 + \alpha\beta D} \cdot e(R_\alpha, g)^{\beta D + C_2 - \beta D} \cdot e(R_\beta, g)^{\alpha D + C_1 - \alpha D} \cdot e(R_\alpha, R_\beta)^{D} \\ &= e(g,g)^{0} \cdot e(R_\alpha, g)^{C_2} \cdot e(R_\beta, g)^{C_1} \cdot e(R_a, R_b)^{D} \\ &= e(R_\alpha^{C_2}, g) \cdot e(R_\beta^{C_1}, g) \cdot e(R_a, R_b)^{D} \\ &= e(R_\alpha^{C_2} \cdot R_\beta^{C_1}, g) \cdot e(R_a, R_b)^{D} \\ &= W. \end{aligned} \tag{5.7}$$

5.6 Security analysis

In this section we analyse the security of the new BVC scheme. We first show that after integrating a VC scheme into the privacy-preserving biometric authentication scheme by Yasuda et al. [3], the template recovery attack discovered in [4] is countered. Then we analyse other types of threats in specific security models.

5.6.1 Countering the template recovery attack

We define the view of a malicious computation server $\mathcal{CS}$ (the Adversary) who tries to recover the stored biometric template of a client in the following. Note that the malicious property implies the Adversary does not need to follow the protocol steps faithfully (in contrast to honest-but-curious, details in below). Furthermore, it is assumed that all other participants ($\mathcal{AS}$, $\mathcal{C}$ and $\mathcal{DB}$) are trusted parties (e.g., no key leakage or data corruption). We show that a probabilistic polynomial-time (PPT) Adversary has negligible probability to achieve the goals with the listed input and output.

Input: the client's pseudonym ID_{pse}, The public key for SHE pk, the public evaluation key for VC ek, the encrypted stored template $vEnc_1(A)$ and the authentication tag σ_A, the encrypted fresh template $vEnc_2(B)$ and the authentication tag σ_B.

Output: the inner product $ct_P = vEnc_1(A) * vEnc_2(A')$ and the final authentication tag σ'_{HD}.

Goals: let the authentication server $\mathcal{AS}$ accept the inner product computation and return $Ver(sk_{vc}, sk, \mathcal{P}_\Delta, ct_P, \sigma_{HD'}, \ell_d, pk) = (1,0)$ or $(1,1)$.
Eventually, the ultimate goal is to recover the stored biometric template $A \in \mathbb{Z}_2^{1024}$.

We show that the malicious computation server $\mathcal{CS}$ cannot forge a tag $\sigma_{HD'}$ that lets $\mathcal{AS}$ pass the check for both Equation (5.1) and Equation (5.2) in the verification step. If we look at these two equations as separate checks, it is possible for the Adversary to cheat on Equation (5.1). As this check only verifies that the returned computation result (ct_{HD} or ct_P) aligns with the arithmetic circuit used to generate the tag (σ_{HD} or $\sigma_{HD'}$), $\mathcal{CS}$ can cheat by substituting the arithmetic circuit of Hamming distance with the inner product. However, to check Equation (5.2), $\mathcal{AS}$ needs to calculate $W = GroupEval(f, R_\alpha, R_\beta)$. Given $\mathcal{AS}$ is a trusted party, the arithmentic circuit in the argument will be the one to calculate the Hamming distance. If the computation server $\mathcal{CS}$ calculates other functions (such as the inner product), the equation check will fail. Moreover, the template recovery attack is a type of hill-climbing attack where multiple trails are required (to adjust the trail vector A'). Since $\mathcal{AS}$ checks the circuit, all the trails will be denied and $\mathcal{CS}$ in consequence can no longer learn any information of the stored biometric template A.

5.6.2 Other threat models

In this section we go beyond the assumptions made previously and analyse different kinds of threats the BVC protocol is facing if the involved entities are malicious. We use the biometric privacy/security analysis framework proposed by Simoens et al. [37] (which is also used in the security analysis in [4]). We consider only the *internal adversaries*, which are corrupted components in the scope of the system. *External adversaries*, defined in [37] as those who only have access to the communication channel, are not in the scope of the analysis. Furthermore, we differentiate the capabilities of the adversaries: $honest - but - curious$ and *malicious*. The former type only allows the Adversary to eavesdropp the communication that concerns itself without deviating from the protocol specification, i.e., modify, remove or generate new information. dmalicious Adversary has more power and can modify existing communication involving itself and disobeys the protocol execution. In this security analysis we consider malicious adversaries. The following threats to privacy in a biometric authentication system are in focus [4]:

- **Biometric reference privacy:** The Adversary should not be able to recover the stored biometric template A.
- **Biometric sample privacy:** The Adversary should not be able to recover the fresh biometric template B.
- **Identity privacy:** The Adversary should not be able to associate a biometric template to a specific user ID.
- **Intractability:** The Adversary should not be able to distinguish whether two authentication attempts are from the same user.

First we make analysis that are relevant for all the entities ($\mathcal{C}$, $\mathcal{CS}$, $\mathcal{AS}$ and $\mathcal{DB}$). One update of BVC compared to the original Yasuda protocol [3] is substituting user ID with a user pseudonym ID_{pse}. One way to implement it will be to hash the original ID and use the outcome digest. Since the user identifier needs to be accessed by all the entities and it is linked to certain biometric templates (stored or fresh), exposing it in plain text violates *identity privacy*. The use of pseudonym can alleviate this issue. Nevertheless, we still fail to achieve *intractability* as the pseudonym is deterministic, which implies that two authentication requests from the same user can be distinguished by all the entities.

Next we analyse the security of the BVC protocol from the perspective of each entity: what shall go wrong if any of them shows malicious behaviours. We use A to denote the reference template and B to denote the fresh template.

- **The client server $\mathcal{C}$:** $\mathcal{C}$ is responsible to capture the reference template A in the enrolment phase and the fresh template in the authentication phase. $\mathcal{C}$ also performs the corresponding encryption. If $\mathcal{C}$ turns malicious, in the worst case A and B will be leaked directly. When Abidin and Mitrokotsa performed the security analysis of the Yasuda protocol in [4], they assumed the client server $\mathcal{C}$ knew the user ID and B. $\mathcal{C}$ can thus initiate a *center search attack* (see Algorithm 2 in

Chapter 3) and recover the stored template A by simulating the computation server $\mathcal{CS}$. The attack procedure in brief is that the Adversary iteratively flips the bits of B until the Hamming distance of A and B is right above the threshold and the result ct_{HD} is rejected. Then the Adversary changes the newly rejected vector bit-by-bit, observes the acceptance result and eventually recover all the bits of A. This attack has been discovered in the Yasuda et al.'s protocol by [4]. Unfortunately, the same attack is still possible in the new BVC protocol. Since the malicious client server $\mathcal{C}$ is not cheating with the actual computation, this attack cannot be detected by the verifiable computation (VC) scheme. In fact, the attack is not dependent on the encryption method but the choice of using Hamming distance to measure the similarity of two templates [4]. This argument applies in the security analysis of BVC too.

A new concern with BVC is the key generation for VC. In the protocol we let the client server $\mathcal{C}$ generate the private key pk_{vc}, the evaluation key ek_{vc} and the authentication tags because we assume $\mathcal{C}$ is a trusted party. If $\mathcal{C}$ turns malicious, it could give fake sk_{vc} to the authentication server $\mathcal{AS}$ and initiates the template recovery attack with the inner product by simulating the computation server $\mathcal{CS}$. Since the Adversary controls sk_{vc}, the computation verification step becomes meaningless and the false computation will no longer be detected.

- **The computation server $\mathcal{CS}$:** the significant improvement of BVC over the original Yasuda protocol [3] is to counter the malicious computation server $\mathcal{CS}$. Unlike the client server $\mathcal{C}$, $\mathcal{CS}$ only has access to the encrypted templates $vEnc_1(A)$ and $vEnc_2(B)$ and the user pseudonym. Moreover, $\mathcal{CS}$ cannot modify the secret key of the VC scheme. We have analysed in the beginning of this chapter how the template recovery attack conducted by $\mathcal{CS}$ can be countered and hence we shorten the discussion here.

  In contrast to the original protocol, $\mathcal{CS}$ needs to calculate an extra value ℓ_d to solve the range issue after integrating VC. However, ℓ_d is still operated on the ciphertext level and is not involved in the second equation (5.2). Thus learning ℓ_d will not give advantage to the Adversary in learning A or B.

- **The database $\mathcal{DB}$:** another update in BVC is to let $\mathcal{DB}$ be a separate entity as opposed to be controlled by $\mathcal{CS}$. The motivation behind is to distribute the responsibilities even more. Once the client server $\mathcal{C}$ generates a tag for a reference template A in the enrolment phase, it could update the entry in the database without involving the computation server $\mathcal{CS}$. A malicious database $\mathcal{DB}$ in theory could simulate the computation server $\mathcal{CS}$ and launches a template recovery attack with the inner product, but again this attack will be circumvented by the VC scheme. In consequence, a malicious database that returns wrong entries (e.g., invalid encrypted templates or authentication tags) will interfere the functionality of the biometric authentication system but from a solely privacy-preserving perspective there is no obvious negative consequence.

- **The authentication server $\mathcal{AS}$:** a malicious $\mathcal{AS}$ will completely break down the templates privacy of the system since it controls the secret key to the somewhat homomorphic encryption scheme (SHE). If $\mathcal{AS}$ successfully eavesdrops the communication between $\mathcal{C}$ and $\mathcal{DB}$ (or $\mathcal{C}$ and $\mathcal{CS}$) and obtains the encrypted templates $vEnc_1(A)$ or $vEnc_2(B)$ with a user pseudonym, it can recover the biometric templates in plain text.

5.7 A flawed approach

Privacy and integrity are the two significant properties we are trying to achieve in a privacy-preserving biometric system. A common way to provide privacy is through a (fully) homomorphic encryption scheme ((F)HE) and integrity is guaranteed through a verifiable computation scheme (VC). The motivation of obtaining both of these properties in one system leads to the trials of combining a FHE and a VC. Fiore et al. in [24] stressed on this problem in the setting of cloud computing. Our privacy-preserving biometric scheme is also reliable on this scheme combination.

Simply put, there are two general ways to combine a VC scheme and a FHE scheme. The first way is to run VC on top of FHE conveying that the original message is first encrypted to preserve privacy and then encoded to generate an authentication proof. Our construction of BVC follows this principle. The second way is to reverse the order and running FHE on top of VC. Namely, this approach will encode the original message first and then encrypt the encoded data. Succinctly, the difference is as follows:

$$\text{The first approach: } \mathit{Encrypt}(\mathit{Encode}(x));$$
$$\text{The second approach: } \mathit{Encode}(\mathit{Encrypt}(x)).$$

We can raise a questoin from here: are the two approaches equally secure? We have studied Abidin and Mitrokotsa's template recovery attack [4]. The attacker (the malicious computation server $\mathcal{CS}$) takes advantage of the homomorphic properties and abuses the authentication server $\mathcal{AS}$ as a decrypting oracle by sending verification queries. Inspired by this attack present in the field of biometric authentication systems, we could try to see if this can be correlated back to cloud computing - is there a "naive" way of combining a VC scheme and a FHE scheme that can be exploited by an attacker to leak privacy? The answer is "yes" and the order we combine them is indeed a determining factor. In below we will show that the **second** approach: $\mathit{Encode}(\mathit{Encrypt}(x))$ is a flawed approach. This problem has also been mentioned in [9] in brief and it is claimed in the paper that their scheme is not chosen cipher text attack (CCA) secure because of this flaw.

Attack scenario

We first provide a general definition of the attack. The privacy-preserving property is defined as the following in this context:

- **Input privacy.** The malicious cloud should not be able to recover any input entries of a computation outsourced by the client in plaintext (either encoded or the original message).
- **Output privacy.** The malicious cloud should not be able to recover the output of a computation in plaintext (either encoded or the original message).

In other words, the cloud should only have access to and should perform operations on the ciphertext level. The scheme is not privacy-preserving if the Adversary breaks any of the two rules mentioned above.

In this attack, we show that the malicious cloud (Adversary) has non-neligible advantage in recovering a computation result in encoded format, hence violating the *output privacy* property.

For example, a specific scenario could be that the cloud is supposed to calculate the average blood pressure over a year of a patient and it stores the daily value in a database in encrypted formats. To preserve output privacy, the cloud should return the computation result encrypted, whose validity is verifiable by the client. With this attack, the malicious cloud will have the advantage to recover the patient's average blood pressure in encoded format.

Attack description

We will describe in detail how the flawed scheme is not resistant against the attack mentioned above conducted by a malicious cloud. We first describe the flawed scheme and then the attack algorithm.

Assume we have:

$$\mathsf{FHE} \text{ scheme - } (KeyGen_{FHE}, Enc, Dec, Eval);$$
$$\mathsf{VC} \text{ scheme - } (KeyGen_{VC}, ProbGen, Compute, Ver).$$

The FHE scheme has been introduced in Chapter 2. It allows computations be to performed on ciphertexts for arbitrary functions. Here a short explanation of the VC algorithms is provided below. $KeyGen_{VC}$ outputs the private key sk_{vc} and public key pk_{vc} for the VC scheme; $ProbGen$ takes sk_{vc} and the plain text x as input and outputs the encoded value σ_x; $Compute$ takes the circuit f, the encrypted encoded input and outputs the encoded version of the output; last but not least, Ver is performed by the Verifier to verify the correctness of the computation given the secret key sk_{vc} and the encoded output σ_y.

The main idea of the flawed approach is to first encode the data in plain text and then encrypt the encoded data. It can be represented by:

$$\hat{x} = Enc(ProbGen(x)),$$
where $\hat{x}$ is what the cloud server gets access to.

The flawed approach can be written as an algorithm FHEVC-FLAWED between a Verifier and an Adversary (the malicious cloud) as depicted in Figure 5.3. The computation could be an arbitrarily chosen arithmetic circuit f. The goal of the Adversary is

to recover every bit of σ_y, i.e., the encoded version of the result of the computation f. More formally speaking, FHEVC-FLAWED is privacy-preserving for all PPT Adversary $\mathcal{A}$ if:

$$Adv_{[\mathsf{FHEVC-FLAWED},\mathcal{A}]} = |Prob(\sigma'_y = Dec(sk_{FHE}, Enc(\sigma_y)))| \text{ is negligible.}$$

After we describe all the phases of the algorithm, we will show that the privacy-preserving property is broken if $q \geq n$, where q is the number of repetitions of the learning phase execution and n is the length of encoded result σ_y.

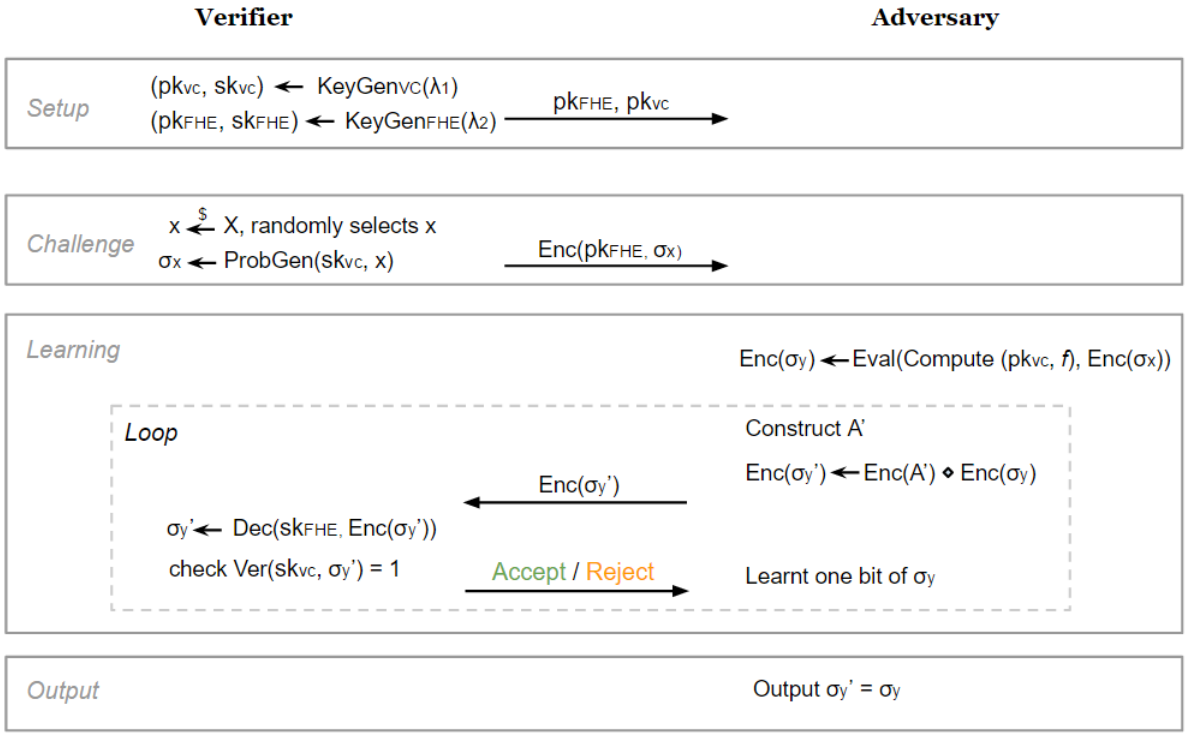

Figure 5.3: The algorithm FHEVC-FLAWED, a flawed approach combining verifiable computation and homomorphic encryption. The details of how to construct A' and how the Adversary can crack one bit of σ_y at the end of each loop is described in the "Learning phase" text as well as illustrated in Figure 5.4.

The algorithm FHEVC-FLAWED is composed of several phases and goes as follows:

Setup phase. The Verifier generates the public and private keys $pk_{vc}, sk_{vc}, pk_{FHE}, pk_{FHE}$ and gives pk_{vc}, pk_{FHE} to the Adversary.

Challenge phase. The Verifier determines the challenge by generating the encoded version σ_x for the input x, which is uniformly selected at random from the input range

X. Then the Verifier encrypts the encoded input and sends $Enc(\sigma_x)$ to the Adversary (the malicious cloud).

Learning phase. In the learning phase the Adversary can use the Verifier as a decryption oracle by sending verification queries, which can be further divided into the following steps:

1. The Adversary performs honest computation on the cipher text level and derives the encrypted result $Enc(\sigma_y)$.

2. The Adversary constructs a vector $A' \in \mathbb{Z}_2^n$ of the same length as σ_y. It is plausible because the length of the tag is known. A' is initialized with the last bit set to 0 and the rest of the bits set to 1 (the 1^{st} trial). For the i^{th} trial, we set $A' = (1_1,1_2,\ldots,0_i,1_{i+1},\ldots,1_{n-1},1_n)$, i.e., set the i^{th} bit to 0 and the rest bits to 1. (The construction of A' is illustrated in Figure 5.4).

3. The Adversary encrypts the specially tailored vector A' and reuses the encrypted honest result $Enc(\sigma_y)$ calculated from step 1. Then he performs the following computation according to **Definition** 3 and sends the result $\sigma_{y'}$ back to the Verifier for verification.

$$Enc(\sigma_{y'}) = Enc(A') \diamond Enc(\sigma_y).$$

4. The Verifier decrypts $Enc(\sigma'_y)$. Thanks to the homomorphic properties that operation performed on the cipher text will match the operation performed on the plain text, the Verifier can derive:

$$Dec(Enc(\sigma_{y'})) = Dec(Enc(A')) \diamond Dec(Enc(\sigma_y));$$
$$\sigma_{y'} = A' \diamond \sigma_y.$$

   The Verifier checks the computation based on the encoded result (after decryption) $\sigma_{y'}$ and sends either *accept* if $Ver(sk, \sigma_{y'}) = 1$ or *reject* if $Ver(sk, \sigma_{y'}) = 0$ to the Adversary.

5. Finally the Adversary obtains the acceptance bit from the Verifier. Essentially A' is acting as a "mask". It copies all the bit values of σ_y into $\sigma_{y'}$ except for the i^{th} bit, which is always set to zero. Consequently, if the output of the verification is *accept*, the Adversary will learn that $\sigma_y = \sigma_{y'}$ as well as $Enc(\sigma_y) = Enc(\sigma_{y'})$, which reveals that the i^{th} bit of σ_y equals to 0. Similarly, if the output of the verification is *reject*, the Adversary learns that the i^{th} bit of σ_y is 1. In both cases, one bit of σ_y is leaked.

Output phase. Assume that the length of σ_y is n. The Adversary sends q verification queries and cracks one bit of σ_y from each query. After $q \geq n$ trials, the Adversary learns every bit of σ_y, outputs the complete $\sigma'_y = \sigma_y$ that passes the verification check

$Ver(sk,\sigma'_y) = 1$ and hence achieves the attack goal. In other words, the advantage of the Adversary: $Adv_{[\mathsf{FHEVC-FLAWED},\mathcal{A}]} = |Prob(\sigma'_y = Dec(sk_{FHE}, Enc(\sigma_y)))|$ is NOT negligible and the security of the scheme FHEVC-FLAWED is broken if $q \geq n$.

Definition 3 (Hadamard product for binary vectors)**.** *We define the operation Hadamard product for binary vectors (one dimension matrix) as* $\diamond : \mathbb{Z}_2^n \diamond \mathbb{Z}_2^n \rightarrow \mathbb{Z}_2^n$*. Given two equal-sized binary strings denoting the stored and the fresh biometric templates:* A *and* $B \in \mathbb{Z}_2^n$*, we compute* $A \diamond B = C \in \mathbb{Z}_2^n$ *as* $C_i = A_i * B_i \in \mathbb{Z}_2$ *for* $i = 1,2,\ldots,n$*. It means multiplying each bit of* A *with the corresponding bit of* B *and the output is a new vector* C *of size* n*. This operation is similar to calculating the inner product, but it outputs a vector rather than an integer.*

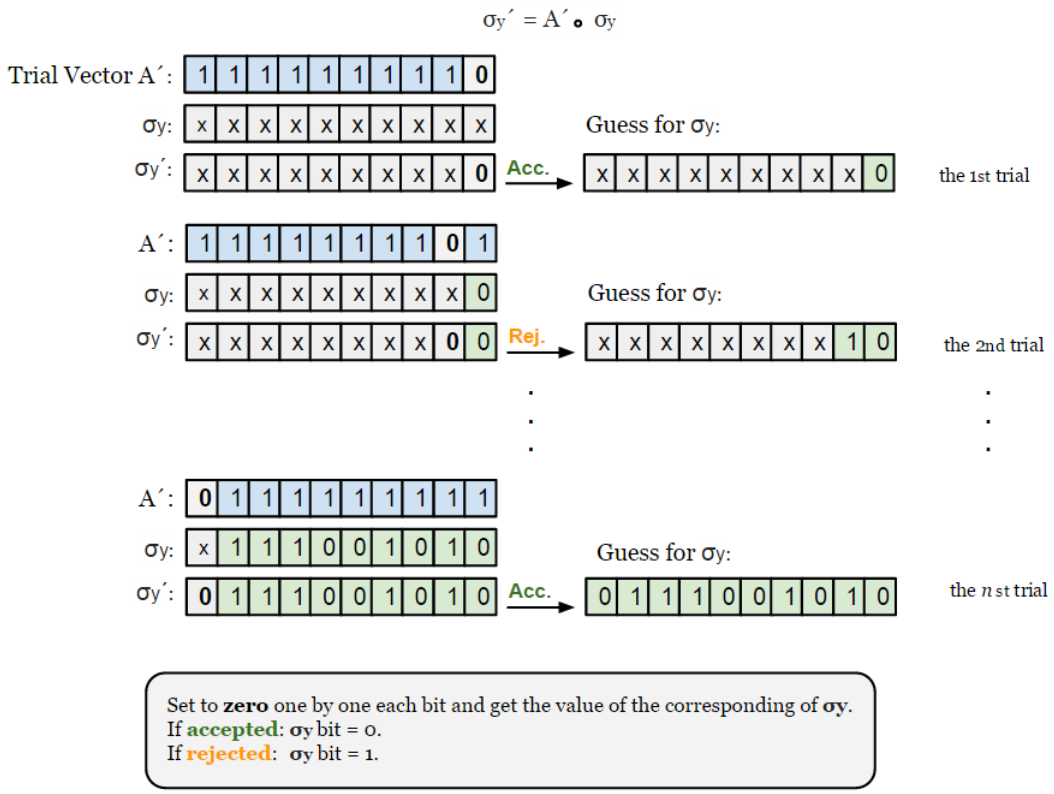

Figure 5.4: The hill climbing attack of the flawed approach combining verifiable computation and homomorphic encryption.

In summary, the attack procedure proceeds in a hill climbing fashion similar to the biometric template recovery attack (see Figure 5.4). The cloud sets to zero one by one each bit of A', computes $Enc(A')$ and $Enc(\sigma_{y'})$, waits for the client to decrypt $Enc(\sigma_{y'})$ and send back the acceptance bit. One verification result implies the recovery of one bit in σ_y and the complete σ_y is recovered after n trails, given n is the length of σ_y.

The attack demonstrates that the order of combining a VC and a (F)HE is very crucial. Since in FHEVC-FLAWED we first perform the encoding and then encryption, the client must decrypt the received result before it can determine whether the computation is accepted or not. The client is therefore exploited as a decrypting oracle and allows the malicious cloud to send verification queries. Privacy is therefore broken. In other words, the procedure is not chosen-cipher text attack (CCA) secure as the Adversary learns information of the plain text (σ_y) from the cipher text ($Enc(\sigma_y)$). If the data stored in the cloud (and the operations performed on them) are sensitive, e.g. average blood pressure value of a patient or some transaction records, the attack can lead to devastating result.

On the other hand, the opposite ordering where VC is run on top of the encryption scheme does not suffer from the attack. In this approach, the Verifier can make the judgement whether the computation is correct or not BEFORE decrypting the received result from the Adversary. Both the verifiable computation scheme on encrypted data by Fiore et al. [24] and our BVC scheme follow this ordering. We will come back to this problem in the discussion section.

Last but not least, we could make a brief analysis what shall go wrong if the flawed approach is used in our BVC scheme. In our case the compromised σ_y value corresponds to the encoded Hamming distance between a fresh template and a reference template. At first sight it is not directly related to the actual templates leakage. However, Pagnin et al. in [29] proves that biometric authentication schemes based on Hamming distance suffer from leakage of information, where a centre search attack can be mounted to recover the stored reference template (defined in $\mathbb{Z}_q^n, q \geqq 2$). On the other hand, from a practical point of view it is inherently difficult to combine the flawed approach with the Yasuda scheme because of the packing methods, where the template vectors are first compacted into one element and then encrypted. The two types of packed ciphertexts are crucial intermediate steps that contribute to the computation and efficient decryption of Hamming distance. Applying the flawed approach will imply generating the tags on the original template vectors and thus affects the application of the packing methods. In order to have a correct scheme, a lot of changes must be adapted.

5.8 Discussion

In this section we will discuss the motivation behind the BVC scheme. We explain why the Backer's homomorphic MAC scheme [15] is the chosen candidate as the fundamental building block. We will also consider some issues that are not addressed in the previous sections, such as the efficiency property and the problems of using pseudonyms as user IDs.

The choice of the VC scheme

In the theory study phase of the project we looked at many different verifiable computation (VC) schemes ranging from purely proof-based to message authenticators (see

the details Chapter 2). They differ in the fundamental mechanism, the complexity level (both theoretical and practical) and target function groups. Some schemes, such as the ones based on fully homomorphic encryption (FHE), are very powerful from a theoretical point of view and suit arbitrary functions. But the drawback is the high complexity. On the other hand, some schemes are designed for a smaller set of function types but the construction is in comparison much easier to understand. We need to make proper trade-offs in choosing the best-fit VC scheme for the Yasuda et al. SHE biometric scheme [3].

We choose the homomorphic MAC scheme by Beckers et al. [15] as the building block because the paper was published rather recently, the construction is relatively simple to follow and it matches the desired function type. The computation in the biometric authentication scenario is not considerably demanding compared to general cloud-computing tasks. In our context the computation function is fixed (Hamming distance calculation) so no function secrecy is required, and the number of variables is also limited. In this sense FHE-based schemes are too heavy to be applied. The homomorphic MAC scheme can be applied to class of computations of quadratic polynomials over a large number of variables [15], which is seen as a good candidate as we need only to handle computation up-to degree 2.

We have also considered developing the BVC scheme based on the work in [24]. This is a recently published paper that properly addresses on verifiable computation on encrypted data. Their approach is to combine a FHE scheme with homomorphic MAC and appears to be the most relevant to our biometric context where privacy-preserving is the highest priority. Moreover, their scheme also supports multi-variate polynomials of degree 2. Nevertheless, the main reason we did not follow Fiore et al.'s construction was its high complexity. As mentioned in the previous paragraph, FHE may be an overpowerful tool to apply. For example, the specific FHE scheme chosen in [24] is a variant of the BGV homomorphic encryption [38], where the ciphertext lives in a ring of polynomial and is composed of several entries. Fiore et al. [24] developed a homomorphic hashing technique that compresses the ciphertext into a single entry in $\mathbb{Z}_p$. In our scenario, the ciphertext outcome from the somewhat homomorphic encryption by Yasuda et al. [3] is already a single entry in $\mathbb{Z}_d$. Even though our candidate VC scheme by Backes et al. [15] did not focus on privacy-preserving, we got the inspiration from [24] and run the homomorphic MAC scheme on top of the SHE scheme described in [3], and hence preserves data privacy. The biggest challenge encountered in bridging the schemes was the data range problem, but we have provided a solution for it.

Lastly, the BVC scheme does not suffer from the output privacy leakage described previously in the "encode-then-encrypt" flawed approach. The reason is that in BVC we do the opposite: "encrypt-then-encode", same as in [24], and the verifier (in our context the authentication server $\mathcal{AS}$) can check the integrity of the computation before decrypting the computed result (the Hamming distance). If the VC check fails, $\mathcal{AS}$ will not even proceed with the decryption. Thus the Adversary cannot benefit from sending verification queries as in the flawed game and abuses the $\mathcal{AS}$ as a decrypting oracle.

Efficiency

Efficiency is a crucial property that verifiable computation schemes try to improve in the cloud computing realm. Even the VC scheme we adopted in this thesis provides alternative algorithms to improve the efficiency of the verifier to perform the check. The mechanism is to use multi-labels and achieve amortized closed-form efficiency. We did not use these algorithms considering the computation function in our context is relatively simple. The verification will run in time $O(|f|)$ and the computation power overhead should be bearable.

In the context of biometric authentication, we prioritized security/privacy over efficiency. Due to the limited time of the project, we chose to eliminate efficiency analysis from the main result. However, the current definition of the multi-labels where the client pseudonym is the *dataset identifier* and the "reference" / "fresh" label is the *input identifier*, should support the addition of amortized closed-form efficiency. If this is the case, the authentication will require one single VC secret key sk_{vc} to manage ALL the biometric registered users since the registered user records will be treated as different data sets outsourced by one single client. If this secret key sk is leaked, the function verification functionality will collapse for all the users. In the current version of BVC we generate one VC secret key per registered user. If some of the keys are leaked, only a portion of the users will be affected. However, the trade-off will be the key and resource management when the user database increases in size. To conclude, we leave out efficiency as an requirement in the scope of this project but a proper efficiency analysis and improvement strategy could be a direction for future work.

Using pseudonym

One update from the original Yasuda et al. protocol is substituting user ID with pseudonyms. In the security analysis we mentioned that this improvement could to a certain extend protect *identity privacy* by increasing the difficulty of linking a template to the right template owner. Further more, another use of the pseudonym is to construct the multi-label required in the VC scheme. However, the introduction of pseudonyms does not contribute in preserving *intractability*. Abidin and Mitrokotsa in [4] proposed to add private information retrieval (PIR) technique as a countermeasure. It will typically require the registered user to send an index and encrypted queries for the database to search for the right entry. We did not integrate a PIR in the current BVC but this can be another branch of the future work.

6

Conclusion and Future Work

In this thesis work we covered two main areas which do not seem very related at first sight: verifiable computation and biometric authentication systems. Thanks to the rapid growth of internet and the proliferation of digital devices, the demand for cloud computing has also risen tremendously. Verifiable computation (VC) enables clients to perform integrity checks for the outsourced computing tasks given the possibility that the cloud server could be malicious. In the other area, biometric authentication has also gained popularity nowdays compared to the traditional passwords based authentication. preserving the privacy of the biometric templates is highly prioritized due to its irrevocable nature. While there are many existing works that focus on privacy, the integrity property is in comparison easier to be ignored. A template recovery attack [4] has been discovered in a privacy-preserving biometric authentication protocol based on somewhat homomorphic encyrption (SHE) [3]. The enabling factor of this attack is the presence of malicious computation server that on purpose performs incorrect computation. If this service is outsourced, this issue is equivalent to malicious cloud. The main research question of this thesis was therefore how to employ a suitable verifiable computation technique to counter the attack due to malicious entities.

The main contribution of this theoretical thesis work is a new scheme named BVC which adds the verifiable computation (VC) feature based on homomorphic message authenticators [15] to the biometric authentication scheme based on SHE [3]. We presented a general scheme description which gave an overview of the algorithms, a detailed construction and an improved distributed biometric authentication protocol supporting VC. We complemented the scheme with a correctness analysis and a security analysis and showed that the template recovery attack is successfully countered in the BVC scheme. In addition, we got inspired by the attack mechanism and conducted a parallel study, whether a similar attack can be achieved in the setting of cloud computing where data privacy is required. The finding revealed that the order combining a VC and a homomorphic encryption scheme was very crucial. If the wrong order is adopted, the bridging

process will become a naive approach and allows the malicious cloud to break the input/output privacy. Last but not least, we discussed several design motivations behind the BVCscheme, in which we explained why Backes et al's VC scheme [15] was chosen to be applied. The main reasons were the matching function types of the two schemes (quadratic polynomials) and its ease to replicate.

This thesis topic can be further developed in several directions, some of which we have touched upon briefly in the discussion section. First of all, some implementation and prototyping of the new BVC scheme could be made. We more or less excluded efficiency analysis due to the prioritizing of security and the simplicity of the Hamming distance function. A proof-of-concept prototype will be a great support to the existing theories. For example, from the implementation we could analyse quantitatively how much computation power is saved if we apply the online/offline stages and amorized efficiency paradigm. Second of all, we can study the feasibility of introducing a privacy-information-retrieval (PIR) layer to the current scheme to replace the user pseudonyms. This improvement will preserve identity privacy. If the study continues towards a theoretical direction, we could make a more generic scheme that is not constrained to the bridging of specific VC and FHE/SHE schemes (e.g., a specific key generation algorithm), but rather can be adapted to a more general level. Last but not least, another direction in the theory field will be to make formal security proofs in proper cryptographic game notation to measure the accurate security of the new scheme (such as the impact to the security of the scheme by introducing ℓ_d to solve the range problem).

Bibliography

[1] V. Beal, cloud computing (the cloud).
URL http://www.webopedia.com/TERM/C/cloud_computing.html

[2] J.-P. Linnartz, P. Tuyls, New shielding functions to enhance privacy and prevent misuse of biometric templates, in: Audio-and Video-Based Biometric Person Authentication, Springer, 2003, pp. 393–402.

[3] M. Yasuda, T. Shimoyama, J. Kogure, K. Yokoyama, T. Koshiba, Practical packing method in somewhat homomorphic encryption, in: Data Privacy Management and Autonomous Spontaneous Security, Springer, 2014, pp. 34–50.

[4] A. Abidin, A. Mitrokotsa, Security aspects of privacy-preserving biometric authentication based on ideal lattices and ring-lwe.

[5] P. Mell, T. Grance, The nist definition of cloud computing.

[6] C. Fontaine, F. Galand, A survey of homomorphic encryption for nonspecialists, EURASIP Journal on Information Security 2007 (2007) 15.

[7] W. Stallings, Cryptography and Network Security: Principles and Practice, 5th Edition, Prentice Hall Press, Upper Saddle River, NJ, USA, 2010.

[8] M. Naehrig, K. Lauter, V. Vaikuntanathan, Can homomorphic encryption be practical?, in: Proceedings of the 3rd ACM workshop on Cloud computing security workshop, ACM, 2011, pp. 113–124.

[9] R. Gennaro, C. Gentry, B. Parno, Non-interactive verifiable computing: Outsourcing computation to untrusted workers, in: Advances in Cryptology–CRYPTO 2010, Springer, 2010, pp. 465–482.

[10] M. Walfish, A. J. Blumberg, Verifying computations without reexecuting them, Communications of the ACM 58 (2) (2015) 74–84.

[11] O. Goldreich, Probabilistic proof systems: A primer, Now Publishers Inc, 2008.

[12] S. Goldwasser, S. Micali, C. Rackoff, The knowledge complexity of interactive proof-systems, in: Proceedings of the seventeenth annual ACM symposium on Theory of computing, ACM, 1985, pp. 291–304.

[13] J. Kilian, Improved efficient arguments, in: Advances in Cryptology—CRYPT0'95, Springer, 1995, pp. 311–324.

[14] D. Catalano, D. Fiore, B. Warinschi, Homomorphic signatures with efficient verification for polynomial functions, in: Advances in Cryptology–CRYPTO 2014, Springer, 2014, pp. 371–389.

[15] M. Backes, D. Fiore, R. M. Reischuk, Verifiable delegation of computation on outsourced data, in: Proceedings of the 2013 ACM SIGSAC conference on Computer & communications security, ACM, 2013, pp. 863–874.

[16] B. Parno, J. Howell, C. Gentry, M. Raykova, Pinocchio: Nearly practical verifiable computation, in: Security and Privacy (SP), 2013 IEEE Symposium on, IEEE, 2013, pp. 238–252.

[17] M. Backes, M. Barbosa, D. Fiore, R. M. Reischuk, Adsnark: nearly practical and privacy-preserving proofs on authenticated data, in: Proc. 36th IEEE Symposium on Security & Privacy (S&P), page to appear. IEEE Computer Society Press, 2015.

[18] C. Gentry, A fully homomorphic encryption scheme, Ph.D. thesis, Stanford University (2009).

[19] K.-M. Chung, Y. Kalai, S. Vadhan, Improved delegation of computation using fully homomorphic encryption, in: Advances in Cryptology–CRYPTO 2010, Springer, 2010, pp. 483–501.

[20] R. Gennaro, D. Wichs, Fully homomorphic message authenticators, in: Advances in Cryptology-ASIACRYPT 2013, Springer, 2013, pp. 301–320.

[21] D. Catalano, D. Fiore, Practical homomorphic macs for arithmetic circuits, in: T. Johansson, P. Nguyen (Eds.), Advances in Cryptology – EUROCRYPT 2013, Vol. 7881 of Lecture Notes in Computer Science, Springer Berlin Heidelberg, 2013, pp. 336–352.
URL `http://dx.doi.org/10.1007/978-3-642-38348-9_21`

[22] R. Johnson, D. Molnar, D. Song, D. Wagner, Homomorphic signature schemes, in: Topics in Cryptology—CT-RSA 2002, Springer, 2002, pp. 244–262.

[23] D. Boneh, D. M. Freeman, Homomorphic signatures for polynomial functions, in: Advances in Cryptology–EUROCRYPT 2011, Springer, 2011, pp. 149–168.

[24] D. Fiore, R. Gennaro, V. Pastro, Efficiently verifiable computation on encrypted data, in: Proceedings of the 2014 ACM SIGSAC Conference on Computer and Communications Security, ACM, 2014, pp. 844–855.

[25] S. Goldwasser, Y. Kalai, R. A. Popa, V. Vaikuntanathan, N. Zeldovich, Reusable garbled circuits and succinct functional encryption, in: Proceedings of the forty-fifth annual ACM symposium on Theory of computing, ACM, 2013, pp. 555–564.

[26] Z. Brakerski, C. Gentry, V. Vaikuntanathan, (leveled) fully homomorphic encryption without bootstrapping, in: Proceedings of the 3rd Innovations in Theoretical Computer Science Conference, ACM, 2012, pp. 309–325.

[27] M. Belenkiy, M. Chase, C. C. Erway, J. Jannotti, A. Küpçü, A. Lysyanskaya, Incentivizing outsourced computation (2008) 85–90.
URL `http://doi.acm.org/10.1145/1403027.1403046`

[28] W. Stallings, L. Brown, Computer Security: Principles and Practice, Always learning, Pearson, 2012.
URL `https://books.google.se/books?id=eQlEkgEACAAJ`

[29] E. Pagnin, C. Dimitrakakis, A. Abidin, A. Mitrokotsa, On the leakage of information in biometric authentication, in: Progress in Cryptology–INDOCRYPT 2014, Springer, 2014, pp. 265–280.

[30] N. K. Ratha, J. H. Connell, R. M. Bolle, Enhancing security and privacy in biometrics-based authentication systems, IBM systems Journal 40 (3) (2001) 614–634.

[31] R. Belguechi, V. Alimi, E. Cherrier, P. Lacharme, C. Rosenberger, An overview on privacy preserving biometrics, Recent Application in Biometrics (2011) 65–84.

[32] A. Juels, M. Wattenberg, A fuzzy commitment scheme, in: Proceedings of the 6th ACM conference on Computer and communications security, ACM, 1999, pp. 28–36.

[33] A. Juels, M. Sudan, A fuzzy vault scheme, Designs, Codes and Cryptography 38 (2) (2006) 237–257.

[34] J. Sedenka, S. Govindarajan, P. Gasti, K. S. Balagani, Secure outsourced biometric authentication with performance evaluation on smartphones, Information Forensics and Security, IEEE Transactions on 10 (2) (2015) 384–396.

[35] C. Gentry, S. Halevi, Implementing gentry's fully-homomorphic encryption scheme, in: Advances in Cryptology–EUROCRYPT 2011, Springer, 2011, pp. 129–148.

[36] D. Boneh, M. Franklin, Identity-based encryption from the weil pairing, in: Advances in Cryptology—CRYPTO 2001, Springer, 2001, pp. 213–229.

[37] K. Simoens, J. Bringer, H. Chabanne, S. Seys, A framework for analyzing template security and privacy in biometric authentication systems, Information Forensics and Security, IEEE Transactions on 7 (2) (2012) 833–841.

[38] Z. Brakerski, V. Vaikuntanathan, Fully homomorphic encryption from ring-lwe and security for key dependent messages, in: Advances in Cryptology–CRYPTO 2011, Springer, 2011, pp. 505–524.