Priv-Ware

Privacy aware processing of encrypted signals for treating sensitive information

List and definition of relevant primitives

Dario Catalano, Mario Di Raimondo, Dario Fiore
University of Catania

Tiziano Bianchi, Alessandro Piva
University of Firenze

Mauro Barni
University of Siena

Sponsored by
MIUR

Contract no. 2007JXH7ET

Deliverable 2.1
10 August 2009
Contents

1 Introduction 3

2 Cryptographic Tools 4
  2.1 Secure Multi-party Computation .......................... 4
    2.1.1 Oblivious Transfer ..................................... 4
    2.1.2 Garbled circuits ........................................... 5
  2.2 Homomorphic Encryption ..................................... 6
    2.2.1 Early mechanisms .......................................... 6
    2.2.2 More efficient constructions .................................. 7
    2.2.3 Extensions of Paillier’s scheme ............................ 7
    2.2.4 Algebraically Homomorphic schemes ........................ 8
  2.3 Searchable Encryption ......................................... 9
    2.3.1 Extensions ................................................ 10
  2.4 Deterministic Encryption ..................................... 14
    2.4.1 Security and Definitions .................................. 14
    2.4.2 Constructions ............................................... 16
    2.4.3 Efficiently Searchable Encryption .......................... 19
    2.4.4 Deterministic Encryption with Hidden Hash Mode ........... 20
  2.5 Zero-Knowledge Sets ........................................... 22
    2.5.1 Zero-Knowledge Sets from mercurial commitments ............ 22
  2.6 Biometrics as Cryptographics Keys .......................... 25
    2.6.1 Fuzzy Extractors .......................................... 25
    2.6.2 Reusable Fuzzy Extractors ................................ 26
    2.6.3 Authentication using Biometrics ............................ 27

3 Signal Processing Basics 28
  3.1 Signal Representation ........................................ 28
    3.1.1 Floating Point Representation .............................. 29
    3.1.2 Fixed Point Representation .................................. 30
    3.1.3 Algebraic Integers ......................................... 31
  3.2 Point Operations .............................................. 32
  3.3 Linear Filtering ................................................ 37
  3.4 Linear Transforms ............................................. 39
  3.5 Non Linear Filters ............................................. 40
  3.6 Statistical Analysis ........................................... 42
  3.7 Signal Compression ............................................. 47
  3.8 High Level Signal Processing ................................. 49

4 Pattern recognition in the encrypted domain 50
  4.1 Two very common tasks ......................................... 50
  4.2 Pattern matching: the verification problem ..................... 51
  4.3 Pattern matching: the identification problem ................... 52
  4.4 The classification problem .................................... 53
5 Conclusion

Abstract

Signal processing algorithms that work directly over encrypted data would be very useful in a variety of practical applications. The contribution of this deliverable is two-fold. First we survey on available tools from cryptography and signal processing. Second we briefly discuss how such tools can be used in the context of pattern recognition in the encrypted domain.

1 Introduction

In recent years the proliferation of the Internet as a new media opened the way to many new possibilities for information and communication. If the opportunities are many, so are the challenges. In particular, recent technological developments are raising serious issues concerning the security of multimedia contents. Such concerns become of crucial importance when treating sensitive information subject to privacy protection regulations. A typical example is the storage and manipulation of medical data. Doctors should clearly be allowed to access data concerning their own patients, but other users should not be able to access such data.

The classical way to protect sensitive information from unauthorized users is to encrypt it as soon as it is created. This basic solution, however, tacitly assumes that the owner of the data and the party in charge of processing it trust each other. This is because, basic encryption schemes, require one to decrypt in order to process the plaintext. The possibility of performing computations directly in the encrypted domain is a central one in cryptography. During the last thirty years a number of beautiful and general methodologies have been proposed in order to solve this problem in different scenarios and settings. Most known techniques, however, share the basic prerequisite that the underlying data have to be expressed as integers in order to be properly processed. This creates some non trivial difficulties if the data one wants to protect is a signal, as signals are often expressed as real numbers. The main goal of this project is to further the interplays between signal processing and cryptography by focusing on specific applications that foster the advancement of signal processing tools that work directly on encrypted signals.

The goal of this deliverable is to survey on (some of) the main cryptographic and signal processing tools, in order to provide a list and definition of relevant primitives, that could be used to perform signal processing in the encrypted domain.

This document is organized as follows. In Section 2 we survey on some of the cryptographic primitives that allow to perform operations over encrypted (or secret) data. This includes a short description of standard and well known techniques as well as a slightly more detailed description of emerging tools that are relevant in our context. Section 3 discusses the main (basic) signal processing operations. As a proposed application, in Section 4, we discuss the problem of pattern recognition in the encrypted domain. In particular, we analyze the most common pattern recognition tools involved in biometrics applications. Then we discuss the security requirements associated to them and we identify a restricted number of primitives that are needed to apply these tools to encrypted data and signals.
2 Cryptographic Tools

2.1 Secure Multi-party Computation

Secure multi-party computation (MPC) [88, 51] allows a set of mutually mistrusting parties to compute a function together while keeping their inputs private.

Mistrust among parties is usually modeled by assuming the existence of an adversary that is allowed to corrupt some partial set of the parties. By corruption, one usually assume that the adversary can actually read (and possibly modify) the internal memory of the corrupted players. The weak reliability of communication, is modeled by allowing the adversary to control the communications involving corrupted players. The MPC paradigm allows many settings and concerns to be modeled and is a strong tool in showing that solutions exists to very general cryptographic problems. The power of the framework is that under partial corruptions assumption (and various settings and constraints) it is possible to compile any polynomial size function into a protocol that maintains input privacy. Input privacy is assured facing an adversary that is assumed to control the entire state (memory) of corrupted parties (passive adversary) and one that in addition may corrupt the memory arbitrarily (malicious adversary).

The notion of MPC is very general and allows numerous variants protocols. Here we briefly recall just some of them. A basic distinction is between the computational setting [51] where all communication is available to the adversary, and the information-theoretic one [9, 31, 81], where point-to-point communication links are completely protected, but the adversary is not restricted to probabilistic polynomial time. Other notions consider different classes of possible adversaries (such as general non-threshold adversaries [56], partial communication graph adversaries [41] or static vs. adaptive adversaries [24]), whereas more recent work investigates several composability issues [69, 26, 27]. Finally, additional works study the case of asynchronous protocols [81] or more advanced security requirements such as non-coercibility [25], or security in the absence of authentication [3].

Even though the results above are very interesting from a theoretical point of view, their generality often comes at the price of efficiency. Indeed general multiparty computation protocols allow to securely compute any efficient function but the efficiency of the computation process is linear in the size of the circuit implementing the function to be computed. This often results in protocols that are simply too inefficient to be used in practice. For this reason, efficient ad-hoc solutions have to be designed to solve specific cryptographic problems.

2.1.1 Oblivious Transfer

Oblivious transfer (OT) is a two party protocol that involves a sender, holding two strings $s_0$ and $s_1$, and a receiver, holding a bit $b$. At the end of the protocol the receiver gets the string $s_b$ but does not get any information about $s_{1-b}$. The sender learns nothing at all about $b$. Oblivious transfer was introduced by Rabin [80] and it is a primitive of central importance in cryptography. Indeed it can be used as a basis for several cryptographic constructions. In particular, any secure multiparty computation protocol can be based on oblivious transfer [87, 51, 63] (see next section). There exists
several variants of the basic primitive, all of them have been proved to be equivalent.
The study of oblivious transfer has been motivated by both theoretical and practical
needs. From a theoretical perspective several papers investigated the primitive in order
to better understand what assumptions are needed to build OT. Nowadays, known con-
structions for oblivious transfer relies either on specific number theoretic assumptions
([80, 6, 71, 2]) or on more generic assumptions, such as the existence of enhanced
trapdoor permutations ([42, 50, 55]). On the negative side we know also that oblivious
transfer cannot be reduced in a black box way to seemingly weaker primitive such as
one way functions [58]. From a practical point of view, the study of oblivious transfer
has been motivated by the fact that it is considered the main bottleneck, in terms of the
amount of computation required, for many secure multiparty computation protocols.

2.1.2 Garbled circuits

Yao [87] proposed a method to carry out secure two-party computation in constant
number of rounds: here both the parties (Alice and Bob) have private inputs \((a \text{ and } b)\)
and they would compute in a “secure way” \(F(a, b)\), where \(F\) is a known function ex-
pressed as a circuit of boolean gates. The basic idea is to make possible the evaluation
of a “garbled copy” of the circuit using “obfuscated inputs”, in particular: one of the
parties (e.g., Alice) is entrusted to choose the way to obfuscate the inputs and to garble
the circuit in a correlated manner, and then the other party (e.g., Bob) can evaluate the
circuit.

More specifically, given the truth table associated to a specific gate of the circuit
(e.g., an AND one) the garbled version is obtained as follows:

1. for each possible value (0 or 1) of each column (i.e., wire) a random uniform
   string is chosen; this is (independently) done for each input/output wire;
2. for each row of the truth table, the value on the output column is (symmetrically)
   encrypted using as key the strings associated to the inputs;
3. finally the garbled version of the truth table will consist of the column with the
   encrypted outputs; a further random permutation of the rows is also applied.

Using such table it is possible to evaluate the garbled gate on the obfuscated inputs
obtaining the resulting (obfuscated) output\(^1\).

In the two-party computation protocol, Alice chooses a random obfuscation for all
the (inputs/output) wires of the circuit: this translation is applied to her inputs and
the circuit is garbled gate-by-gate as explained before. Her obfuscated inputs as well
as a copy of the garbled circuit are sent to Bob. In order to evaluate the circuit he
requires the obfuscation of his inputs: it is obtained using, for each bit of the input, an
Oblivious Transfer protocol (see Section 2.1.1). Note that all the instances of the OT
can be parallelized. The evaluation of the garbled circuit is done by Bob following a
topological order in the circuit. The last phase requires that Alice reveals to Bob the

\(^1\)It is required the following additional property on the underlying encryption scheme: the decryption
process should fail (with high probability) if the key is different than the one used in the encryption step.
obfuscation information related to the output wires: in this way Bob is able to gain (and share with Alice) the output value of the circuit in clear.

The resulting scheme uses a constant number of rounds but the computational complexity (as well as the required bandwidth) can rapidly grow. It is secure only against a passive attacker, but solutions against malicious adversaries are also available [64, 72].

### 2.2 Homomorphic Encryption

Homomorphic encryption schemes have the remarkable property according to which the encryption procedure preserves the algebraic structure of the message space. This allows to perform operations on the encrypted plaintext operating directly on the corresponding ciphertexts. For the case of additively homomorphic encryption schemes this is usually achieved as follows. The message space is set as a ring $M$ of modular residues. The ciphertext space is a multiplicative group $G$ of invertible elements in a ring of integers modulo a hard to factor number $N$. The encryption of a plaintext $m$ is a group element $E(m, r) = g^{mr}e \in G$, where $e$ is a public value (typically a divisor of the order of $G$), $g$ is public element in $G$ and $r$ is a random value in some subgroup $R$ of $G$. The resulting schemes have the desired homomorphic property as an encryption of $m_1 + m_2$ can be obtained from encryptions of $m_1$ and $m_2$ as

$$E(m_1, r_1) \cdot E(m_2, r_2)$$

#### 2.2.1 Early mechanisms

The idea described above was first proposed by Goldwasser and Micali [52] in their seminal paper on probabilistic encryption. The Goldwasser-Micali cryptosystem is based on quadratic residues. It sets $M = \mathbb{Z}_2$, $G = R = \mathbb{Z}_N^*$ for some hard to factor modulus $N = pq$ and $e = 2$. The public element $g$ is set as a random pseudo-square modulo $N$. The security of the scheme relies on the so called quadratic residuosity assumption

$$\text{2A number } a \in \mathbb{Z}_N^* \text{ is a quadratic residue (modulo } N) \text{ if there is another element } b \in \mathbb{Z}_N^* \text{ such that } a \equiv b^2 \mod N. \text{ Let } J_N \text{ be the subset of } \mathbb{Z}_N^* \text{ formed by all the element having Jacobi symbol equal to 1 (the reader is referred to any introductory book on number theory for a definition of the Jacobi symbol). It is possible to prove that the set of quadratic residues modulo } N \text{ is a (proper) subset of } J_N. \text{ The quadratic residuosity assumption states that, unless the factorization of } N \text{ is available, it is computationally infeasible to distinguish a quadratic residue modulo } N \text{ from quadratic non residues in } J_N. \text{ We point out that the limitation on the size of } e \text{ comes from the fact that the}$$

$$\text{3The prime residuosity assumption generalizes the quadratic residuosity assumption as follows. Let } p \text{ be a prime dividing } \phi(N), \text{ we say that } a \text{ is a } p\text{-residue (modulo } N), \text{ if there is another element } b \in \mathbb{Z}_N^* \text{ such that } a \equiv b^p \mod N. \text{ The prime residuosity assumption states that, unless the factorization of } N \text{ is given, it is computationally infeasible to distinguish random } p\text{-residues from random elements in } \mathbb{Z}_N^*.}$$
decryption procedure is very inefficient as it requires exhaustive search to retrieve the message.

2.2.2 More efficient constructions

In 1998 Naccache and Stern [70] suggested a variant of basic Benaloh-Fischer scheme that allows for higher bandwidth. The new scheme sets $e$ as the product of $k$ small primes $e_1, \ldots, e_k$ such that each $e_i$ divides $\phi(N)$ but $e_i^2$ does not. The basis $g$ is a non $e_i$-th residue for all $i = 1, \ldots, k$. Interestingly, the security of the new scheme still relies on the prime residuosity assumption. Independently, Okamoto and Uchiyama [74] proposed a completely different approach to generalize the Benaloh-Fischer scheme. The underlying idea of the new scheme is to switch to a group with a different structure. They set $\mathcal{G} = \mathcal{R} = \mathbb{Z}_N^*$, but this time $N = p^2q$, rather than $N = pq$ as in previous proposals. The message space $\mathcal{M}$ is set as $\mathbb{Z}_p$, the public exponent is $e = N$ and $g$ is such that $g^e$ has order $p - 1$ modulo $p^2$. The scheme can be proven secure under the so called $p$-Subgroup assumption, moreover it reaches an encryption bandwidth similar to that of Naccache-Stern while allowing for a more efficient decryption mechanism. On the negative side, there exists a trivial chosen ciphertext attack that allows to recover the factorization of the modulus.

In 1999 Paillier [77] proposed an improvement of the Okamoto Uchiyama scheme that retains the nice properties of the latter while removing the bad aspects. In Paillier’s scheme $N$ is set as the usual RSA modulus. All the operations, however, are carried over $\mathbb{Z}_{N^2}^*$. Letting $N = pq$ one sets $\mathcal{R} = \mathbb{Z}_N^*$, $\mathcal{M} = \mathbb{Z}_N$, $\mathcal{G} = \mathbb{Z}_{N^2}^*$, $e = N$ and $g$ is an element whose order is a multiple of $N$. The security of the scheme relies on the Composite residuosity assumption, that states that should be (computationally) infeasible to distinguish $N$-th powers (i.e. elements obtained by raising to the $N$-th power some other element of $\mathbb{Z}_{N^2}^*$) from random elements in $\mathbb{Z}_{N^2}^*$. Perhaps surprisingly, Paillier’s scheme also allows for a better encryption ratio with respect to all the previously mentioned schemes. This is why it is probably the best option when an additively homomorphic encryption scheme is needed.

2.2.3 Extensions of Paillier’s scheme

In 2001 Damgård and Jurik [38] proposed an extension of Paillier’s scheme that allows to consider $\mathcal{M} = \mathbb{Z}_{N^s}$, for arbitrary $s$. The resulting scheme retains the useful homomorphic property and it achieves a better ciphertext size/message size ratio (with respect to the original scheme). In 2002 Cramer and Shoup [36] gave a general framework to achieve security against chosen ciphertext attacks for a large class of encryption schemes. Among the others, they show how to appropriately modify Paillier’s scheme to fit the framework. Clearly the resulting scheme is not homomorphic anymore. Interestingly, the new scheme allows for a double decryption mechanism: one can retrieve the plaintext either if the factorization of the modulus is available or if

---

4Informally the assumption states that, without the factorization of $N = p^2q$, it should be infeasible to (efficiently) distinguish random elements in $\mathbb{Z}_{N^2}^*$, from elements whose order in $\mathbb{Z}_{p^2}^*$ is $p - 1$.

5Indeed, it is very easy to show that any homomorphic encryption scheme cannot be secure against an adaptive chosen ciphertext attack.
some specific discrete log information, related to the public key, is given. Building on
this Bresson et al. [22] showed that by slightly modifying the key generation algorithm
of the Cramer-Shoup variant one gets a scheme that, while retaining all the nice prop-
erties of the original variant, also allows for additional applications (see [22] for details).
In 2002, Fouque, Stern and Wackers [45], showed how to extend Paillier’s encryption
scheme to work over the rationals. Their methods employ lattice-based techniques to
"extract" the encrypted rational in the decryption procedure. Such a solution allows
to use Paillier’s cryptosystem to perform some limited crypto computing over the ra-
tionals. The limitation comes from the fact that one has to impose some restrictions
on the size of the encrypted rationals, in order for the lattice-based decryption proce-
dure to work. Cramer, Damgaard and Nielsen [35] show how to do efficient multiparty
computation using any (additively) homomorphic encryption scheme and in particular
using Paillier’s cryptosystem.

2.2.4 Algebraically Homomorphic schemes

An encryption scheme defined over a ring $\mathcal{M}$ is said to be algebraically homomorphic
if the encryption procedure is actually a ring homomorphism. In other words an al-
gebraically homomorphic encryption is a encryption mechanism where both the inner
laws of the underlying ring $\langle \mathcal{M}, +, \cdot \rangle$ are supported. This means that the cipher-
ertext space is equipped with two (efficiently computable) operations $\text{plus}$ and $\text{times}$ such
that, for all $m_1, m_2 \in \mathcal{M}$ and $r_1, r_2 \in \mathcal{R}$ one has that

$$E(m_1, r_1) \text{ plus } E(m_2, r_2) = E(m_1 + m_2, r)$$
$$E(m_1, r_1) \text{ times } E(m_2, r_2) = E(m_1 \cdot m_2, r')$$

for some $r, r' \in \mathcal{R}$.

Thus such schemes would allow to significantly extend the range of functionali-
ties that can be efficiently computed over encrypted data. The problem of constructing
an algebraically homomorphic encryption scheme was posed by Rivest, Adleman and
Dertouzos [83]. Later, Feigenbaum and Merritt [43] pointed out that having an alge-
braically homomorphic cryptosystem over $\mathbb{Z}_2$ would allow to implement secure circuit
evaluation in a complete non interactive way.

In [16] Boneh and Lipton proved that any deterministic algebraically homomorphic
encryption is subject to a subexponential attack. The attack follows from a subexpo-
nential solution to the so-called black box field problem (BBFP, for short). Informally,
this can be stated as follows. Let $p$ be a prime and $\mathbb{F}_p$ a finite field represented as a black
box field$^7$. The BBFP problem consists in finding, given a black box representation of
a field element $a$, an integer $b \in \mathbb{Z}_p$ such that $b \equiv a \mod p$. Boneh and Lipton show
that under the so called smoothness assumption one can construct a subexponential (in $|p|$)
algorithm for the BBFP. Moreover, they prove that any deterministic algebraically

$^6$Notice that the notation assumes $E$ to be a probabilistic encryption scheme. Of course the definition
extends to deterministic schemes as well. In such a case, no randomness is involved in the process and all
the $r$’s above are set as the empty string.

$^7$Informally this means that no specific representation for its element is assumed, thus in order for algo-
rithms to work with such an algebraic structure, appropriate oracles for the various operators are provided.
homomorphic encryption scheme can be seen as an instance of the BBFP. However, we stress that, in order for their reasoning to work, it is crucial to assume that the encryption scheme is deterministic.

In 1999 Sander, Toung and Yung [84], presented an encryption scheme that is homomorphic over a semigroup. The construction is both AND and XOR homomorphic. The cryptosystem, however, is mainly of theoretical interest as the size of the ciphertext grows exponentially in the number of XOR operations performed. More recently Boneh, Goh and Nissim [15] described a homomorphic scheme that allows for an unbounded number of additions and one single multiplication. The scheme, however, is quite inefficient as the decryption procedure is exponential in the length of the message space (and thus it can be used only for rather small message spaces). The scheme works over finite groups of composite order that support a bilinear map. The basic idea of the scheme is to build an additively homomorphic scheme using ideas similar to those described for Paillier’s cryptosystem. The bilinear map then allows for one multiplication of the encrypted data.

In 2009 Craig Gentry proposed a fully algebraically homomorphic encryption scheme based on ideal lattices. The scheme builds upon the following observation. To construct an encryption scheme that allow the evaluation of arbitrary circuits, it is enough to come up with an encryption scheme that can compute a (slightly modified) version of its own decryption circuit. Such a scheme is called bootstrappable. Next, an implementation of a bootstrappable encryption scheme from ideal lattices, is given. The scheme, however, is only of theoretical interest as it is too inefficient to be considered practical.

In conclusion, it is fair to say that no fully satisfactory algebraically homomorphic encryption schemes is known to exist. The quest for such schemes started shortly after the invention of public key cryptography and, in spite of thirty years of considerable research efforts, it remains one of the most important challenges in cryptography.

2.3 Searchable Encryption

Imagine that Alice wants a gateway to route her (encrypted) email to different locations (e.g. her desktop, laptop or pager) based on a keyword attached to the email, but for privacy she wants the keyword encrypted. A Public Key Encryption Scheme with keyword search (PEKS) [13] allows to do exactly this. Alice has a public-key pk and secret key sk. Using sk she can compute a trapdoor \( T_{W'} \) for any keyword \( W' \), and provide it to the gateway. A sender can apply an encryption function \( \text{Enc} \) to the public-key \( \text{pk} \) and a keyword \( W \) to get a ciphertext \( C \) which is sent to the gateway. The latter applies a function \( \text{Test}(\text{pk}, T_{W'}, C) \) to determine whether or not \( W' = W \).

Two requirements are made for PEKS. The first (security condition) is that the gateway learns nothing about \( W' \) beyond whether it equals \( W \) or not. The second requirement (consistency condition) guarantees that the primitive realizes the desired functionality. In our context this means that if \( W' = W \) then \( \text{Test}(\text{pk}, T_{W'}, C) \) should output 1 while it should output 0, if \( W' \neq W \). These requirements were first form-
ized in [13], where a connection between PEKS and identity-based encryption (IBE) was noticed. This connection appears very natural: in an IBE scheme the master secret key (msk) can be used to derive "local" secret keys which allow decryption of ciphertexts generated with respect to specific identities. The construction given in [13] builds a PEKS from the Boneh-Franklin IBE [14] as follows. To encrypt a given keyword \( W \) one encrypts some known string \( \lambda \) (say the string of all 0's) with identity \( W \). The master secret key can be used to compute \( T_W \) by setting \( T_W \) the local secret key corresponding to identity \( W \). Anybody in possession of \( T_W \) can test if a given ciphertext \( C \) contains \( W \) by simply checking if decrypting \( C \), using \( T_W \), leads to \( \lambda \). A more general connection between PEKS and IBE was given by Abdalla et al. [1] where it was proven that the PEKS primitive is actually equivalent to a class of IBEs known as anonymous identity-based encryption schemes. In a nutshell, an identity-based encryption scheme is said to be anonymous if the produced ciphertexts do not reveal the identity of the recipient they were constructed for. Abdalla et al. also generalized the notion of consistency for PEKS and corrected a mistake (concerning consistency) from [13].

**SEARCHABLE ENCRYPTION WITH DECRYPTION**

Notice that a PEKS scheme, by itself does not allow the gateway to decrypt the keyword (but only to check if a given keyword is present or not). In some situations, however it might be desiderable to have such a feature (for example, once the user gets her emails from the gateway, she might want to be able to sort them by keyword, or simply be able to know all the keywords associated to her emails). A naive solution to the problem can be to add a (standard) public key encryption (PKE) scheme, on top of the PEKS. The user can then obtain the value of each keyword by a simply decryption of the corresponding ciphertext. This solution however is not fully satisfactory as it does not guarantee consistency (a malicious sender can send two different keywords: one embedded in the PEKS and the other one encrypted using the PKE). This led to the notion of searchable encryption scheme with decryption (PEKSD for short). Several constructions implementing the new primitive are known [46, 57].

### 2.3.1 Extensions

Recently there has been a trend for users to store their private data over the Internet. However, to guarantee privacy, one should encrypt the data before storing them. Using a traditional encryption scheme (either symmetric or asymmetric) users cannot search over their own data (without decrypting them first). In this setting PEKS can be seen as an encryption mechanism that allows to perform equality tests directly in the encrypted domain (like the schemes in Section 2.4). It is natural then to ask if one can construct encryption primitives realizing more expressive predicates. Along this line of research, Goyal et al. [54] introduced the notion of *attribute-based encryption* (ABE). In an ABE scheme a ciphertext is labeled with a set of attributes and secret keys are associated with predicates. A user can decrypt a given ciphertext \( C \) if and only if the attributes of the ciphertext satisfy the predicate of the key. Such a solution is interesting because it can be used to check if a ciphertext \( C \) contains a message satisfying some given predicate without needing to first decrypting \( C \). The construction given in [54] is quite general.
as it allows to consider basically any predicate that can be expressed as a circuit with threshold gates. Unfortunately, however, it only guarantees security of the encrypted message. In particular, the set of attributes are explicitly revealed by the ciphertext. Using a more technical terminology the scheme guarantees payload security but it is not attribute hiding.

In 2007 Boneh and Waters [17] proposed a (public-key) encryption scheme that supports comparison queries \((x \geq y)\) on encrypted data as well as more general queries such as subset queries (is \(x \in S\)). These systems support arbitrary conjunctive queries \((P_1 \land \ldots \land P_k)\) without leaking information on individual predicates. Going back to the email gateway example, the Boneh and Waters solution allows Alice to ask the gateway to route only the emails satisfying some predicate (such as comparison or conjunctions) in a way such that the gateway does not learn anything beyond the value of the predicate (for example if the predicate is a conjunction \(P_1 \land P_2\), in the case it is false, the gateway should not learn which of the two conjuncts \(P_1\) or \(P_2\) is false). The main technical tool used to realize the new primitive is the notion of Hidden Vector Encryption (HVE). Very informally this primitive can be seen as a generalization of the notion of anonymous identity-based encryption. More precisely an HVE can be described as follows. Let \(\Sigma\) be a (finite) set and let \(*\) be a special symbol not in \(\Sigma\). Let \(\Sigma^* = \Sigma \cup \{\ast\}\) where the \(*\) symbol can be interpreted as a wildcard (or a don’t care symbol). One can typically set \(\Sigma = \{0, 1\}\). For \(\sigma = (\sigma_1, \ldots, \sigma_\ell) \in \Sigma_\ell^*\) one can define a predicate \(P^{HVE}_\sigma\) over \(\Sigma^\ell\) as follows. For \(x = (x_1, \ldots, x_\ell) \in \Sigma^\ell\) set:

\[
P^{HVE}_\sigma(x) = \begin{cases} 1 & \text{if } \forall i = 1, \ldots, \ell \ (\sigma_i = x_i) \lor (\sigma_i = \ast) \\ 0 & \text{otherwise} \end{cases}
\]

Now letting \(\Phi^{HVE} = \{P^{HVE}_\sigma\text{ for all } \sigma \in \Sigma_\ell^*\}\), we say that an HVE is a (secure) \(\Phi^{HVE}\)-searchable encryption. In other words an Hidden Vector Encryption is a public-key encryption scheme that allows to evaluate predicates of the form mentioned above, directly in the encrypted domain. The construction given in [17] is rather inefficient as it requires a bilinear group\(^9\) of order \(N\) where \(N\) is an hard to factor composite. Moreover the size of the produced ciphertexts grows linearly with the size of the universe where the predicate is defined. To be more precise this means that, if one wants to use the encryption scheme to implement subset queries of the form “is \(x \in A\)” for an arbitrary subset of \(\{1, \ldots, n\}\), the ciphertext produced has size \(O(n)\). Even worse, if one wants to implement subset conjunctions such as \((x_1 \in A_1) \land \ldots \land (x_k \in A_k)\), this would require a ciphertext of size \(O(wn)\). Thus the scheme is not efficient enough to be of practical interest and should be seen merely as a feasibility result. A more efficient solution was recently provided by Iovino and Persiano [59]. The main difference between the Iovino-Persiano solution and the scheme given in [17] is that the former works over bilinear groups of prime order. This implies that the resulting parameters are considerably shorter than those required in the Boneh-Waters scheme.

In 2008, Katz, Sahai and Waters [62] introduced the notion of predicate encryption. This can be seen as a significant generalization of the notion of identity-based encryption.

---

\(^9\)Here by bilinear group we mean a group equipped with a bilinear map and where the discrete logarithm problem is conjectured to be intractable. Examples of such maps are the Weil or Tate pairing over certain group of points of elliptic curves.
encryption. Similarly to attribute-based encryption, in predicate encryption ciphertexts are associated with attributes while secret keys are associated to predicates. A secret key (corresponding to predicate \( f \)) can be used to decrypt a ciphertext \( C \), associated to attribute \( I \), iff \( f(I) = 1 \). The main difference between attribute-based encryption and predicate encryption is that the former notion does not consider the issue of hiding the attributes (and thus only payload security is taken into account). The construction of predicate encryption given in [62] is significantly more expressive than previous work as it supports disjunctions, polynomial equations and inner products. At the same time it is totally impractical as (among other inefficiencies) it works on bilinear groups whose order \( N \) is the product of three large primes (for nowadays security requirements this means \( |N| \approx 4500 \) bits!). The basic idea of the construction given in [62] is the following. First they show how to construct a predicate encryption scheme that supports inner products predicates. This is done by implementing a predicate encryption where the set of attributes is \( \Sigma = \mathbb{Z}_N \) (recall that here \( N \) is composite modulus obtained as the product of three large primes). The class of supported predicates is \( F = \{ f_v | v \in \mathbb{Z}_N^\ell \} \) with \( f_v(x) = 1 \) iff \( \langle v, x \rangle = 0 \mod N \) (where \( \langle \cdot, \cdot \rangle \) denotes the inner product). Thus a secret key associated with a predicate \( f_v \) can be used to decrypt a ciphertext \( C \) associated with attributes \( x \) iff \( \langle v, x \rangle = 0 \mod N \).

Next, they show that using the inner product construction as basic building block one can also implement many other functionalities and primitives (such as anonymous identity-based encryption, Hidden Vector Encryption, Predicate Encryption supporting polynomial evaluation, exact thresholds, disjunctions, conjunctions and the evaluation of CNF and DNF formulas). This is done as follows.

**Anonymous Identity-based encryption**  An anonymous IBE (AIBE) can be constructed from an inner product encryption scheme (setup, keyder, enc, dec) as follows. The master secret key and the public-key of the AIBE are generated using the setup algorithm that generates the public parameter and the master secret key for the inner product encryption scheme. To generate the secret key corresponding to identity \( I \in \mathbb{Z}_N \) one sets \( \bar{T} = (1, I) \) and output the secret key for the predicate \( f_{\bar{T}} \). To encrypt a message for the identity \( I' \leftarrow \mathbb{Z}_N \), one sets \( I'' \leftarrow (\bar{T}'', 1) \) and encrypts the message using the encryption algorithm from the predicate encryption scheme. Since \( \langle \bar{T}, \bar{T}' \rangle = 0 \) if and only if \( I = I' \), correctness and security follow from the analogous properties of the underlying predicate encryption scheme.

**Hidden Vector Encryption**  Hidden vector encryption with \( \Sigma = \mathbb{Z}_N \) can be realized from any inner product encryption (setup, keyder, enc, dec), of dimension \( 2\ell \), as follows. The setup algorithm is the same as setup. To generate a secret key corresponding to the predicate
\[
\psi^{\text{HVE}}_{(a_1, \ldots, a_\ell)}(x_1, \ldots, x_\ell) = \begin{cases} 
1 & \text{if } \forall i = 1, \ldots, \ell \ (a_i = x_i) \lor (a_i = \ast) \\
0 & \text{otherwise}
\end{cases}
\]

one computes a vector \( \vec{A} = (A_1, \ldots, A_{2\ell}) \) as follows. If \( a_i \neq \ast \) \( A_{2i-1} \leftarrow 1 \) and \( A_{2i} \leftarrow a_i \). Otherwise, \( A_{2i-1} \leftarrow 0 \) and \( A_{2i} \leftarrow 0 \). The secret key is obtained by running keyder on input \( f_{\vec{A}} \).
To encrypt \( m \) for the attribute \((x_1 \ldots x_\ell)\), one chooses (at random) \((r_1 \ldots r_\ell) \in \mathbb{Z}_N^\ell\) and defines \((\overline{X}) = (X_1, \ldots X_{2\ell})\), where \(X_{2i-1} \leftarrow r_i \mod N\) and \(X_{2i} \leftarrow r_i\) for all \(i\)'s. The ciphertext is computed as \(\text{enc}((\overline{X}), m)\). Correctness can be easily verified and security follows from the security of the underlying inner product encryption scheme.

**Predicate encryption supporting polynomial evaluation**  From an inner product encryption scheme \((\text{setup}, \text{keyder}, \text{enc}, \text{dec})\) (of dimension \(d + 1\)), it is also possible to construct predicate encryption schemes for polynomial evaluation. Let \(\phi_{\leq d}^{\text{poly}} = \{f_p \mid p \in \mathbb{Z}_N[x], \deg(p) \leq d\} \) where

\[
  f_p(x) = \begin{cases} 
    1 & \text{if } \forall i = 1, \ldots, \ell \ (a_i = x_i) \lor (a_i = *) \\
    0 & \text{otherwise}
  \end{cases}
\]

The construction is as follows. The setup algorithm is the same as setup. The secret key corresponding to \(p(x) = a_d x^d + \ldots + a_0\) is generated by setting \(\overline{p} \leftarrow (a_d, \ldots, a_0)\) and by running \(\text{keyder}(\overline{p})\). Encryption of a message \(m\), with respect to attribute \(w \in \mathbb{Z}_N\), is obtained by setting \(\overline{w} \leftarrow (w^d \mod N, \ldots, w^0 \mod N)\) and then outputting \(C \leftarrow \text{enc}(\overline{w}, m)\).

Correctness (and security) follow from the fact that \(p(w) = 0\) if and only if \(\langle \overline{p}, \overline{w} \rangle = 0\).

**Disjunctions, conjunctions and evaluating DNF and CNF formulas**  The construction described above can be generalized to deal with disjunctions, conjunctions, and evaluating DNF and CNF. The basic \(\text{OR}_{I_1, I_2}\) predicate (\(\text{OR}_{I_1, I_2} = 1\) iff \(x = I_1 \lor x = I_2\)), can be encoded using an univariate polynomial \(p(x) = (x - I_1)(x - I_2)\). Similarly the \(\text{AND}_{I_1, I_2}\) predicate (\(\text{AND}_{I_1, I_2} = 1\) iff \(x = I_1 \land x = I_2\)), can be encoded using an univariate polynomial \(p(x) = r(x - I_1) + (x - I_2)\) for some random \(r\).

Such constructions can be easily extended to more complex combinations of disjunctions and conjunctions. For the case of boolean variables this means that a predicate encryption supporting polynomial evaluation allows to handle arbitrary CNF and DNF formulas (this does not generalize to the case of non boolean variables because of negation, something that, at the moment is not clear how to handle).

**Exact thresholds**  One can construct a special form of fuzzy encryption scheme, from any inner product encryption \((\text{setup}, \text{keyder}, \text{enc}, \text{dec})\), of dimension \(\ell + 1\). By special here we mean that to decrypt a ciphertext, constructed with respect to a given set \(S\) of attributes, one needs to be in posses of a key, that "matches" some subset (fixed in size) of \(S\). The construction is as follows. The setup algorithm is the same as setup. Fix the set \(A = \{1, \ldots, \ell\}\) and let \(\psi = \{\psi_S \mid S \subseteq A\}\), where for a given subset \(S'\) of \(A\) we set \(\psi(S') = 1\) if and only if \(|S \cap S'| = t\). To generate a secret key corresponding to the predicate \(\psi_S \in \psi\), we define a vector \(\overline{\psi} \in \mathbb{Z}_N^{\ell+1}\) as follows. For \(1 \leq i \leq \ell\) \(v_i = 1\) iff \(i \in S\), \(v_{\ell+1} = 1\). The secret key is then obtained as the output of \(\text{keyder}(\overline{\psi})\).

To encrypt a message \(m\) for some attribute \(S' \subseteq A\), we define a vector \(\overline{v'} \in \mathbb{Z}_N^{\ell+1}\)

\[10\] Notice also that generalizing these constructions to their corresponding multivariate extensions is straightforward.
as follows. For $1 \leq i \leq \ell$ $v'_i = 1$ iff $i \in S'$, $v'_{i+1} = -t \mod N$. The ciphertext is produced as $C \leftarrow \text{enc}(\overrightarrow{v'}, m)$.

Correctness and security follow from the fact that $|S \cap S'| = t$ iff $\langle \overrightarrow{v}, \overrightarrow{v}' \rangle$.

In a very recent paper by Shen, Shi and Waters [85] the issue of predicate privacy was considered. In a nutshell, predicate privacy requires that tokens reveal no information about the encoded query predicate. Such a feature is inherently impossible to achieve in the public-key setting. For this reason the authors consider predicate encryption in the symmetric-key setting and present a symmetric-key predicate encryption scheme which supports inner product queries. The scheme achieves both plaintext privacy and predicate privacy, but it is extremely inefficient as it works on bilinear groups whose order is a composite hard to factor $N$ obtained as the product of four large primes.

In conclusion, searchable encryption and its generalizations, is a fascinating and promising direction of research, where, however, the most powerful solutions are at the moment too inefficient to be considered practical.

2.4 Deterministic Encryption

The main feature of a deterministic encryption scheme is that a plaintext is associated to one and only one ciphertext. This kind of tool matters in practical contexts like the remote data storage in outsourced databases: the information are stored in encrypted form and the owner would be able to retrieve records without revealing any sensible information to the data-storage manager (except the fact that the queried record is present or not into the database). Assuming the use of a public-key infrastructure\textsuperscript{11}, it is possible to use an encryption scheme with keyword-search (see Section 2.3) to satisfy these requirements; unfortunately these solutions allows the search process to be at most linear in the number of records and thus the retrieval operation becomes prohibitive when applied to very big databases.

The use of deterministic encryption permits faster search operations using the ciphertext as key of the search: e.g. using a tree-based data structure it is needed time logarithmic in the number of the stored records.

2.4.1 Security and Definitions

The use of deterministic encryption to protect the confidentiality of our data permits to design very efficient database structures but, on the other hand, it opens several security issues: no privacy is possible when the plaintext is known to come from a small space. Indeed, it is always possible to retrieve the encrypted plaintext comparing the target ciphertext with the encryptions of every possible plaintext. It is important to note that if the encryption process is deterministic this attack is unavoidable, so we can guarantee privacy only when the plaintext is drawn from a space of large min-entropy.

\textsuperscript{11}This section covers only the public-key setting where the encryption of the records is done using the public-key of the database-owner such that he is the only one allowed to decrypt them. The data-storage manager is supposed to know only information of public domain.
Cryptographers community has spent a lot of efforts in the field of the provable security\textsuperscript{12}: here the provided schemes have a guarantee that if the alleged security properties are (efficiently) violated then one is able to solve instances of well-known hard-problems. The first step to accomplish this is to formally state what is the meaning of “secure scheme”. In the specific case of encryption we have to establish the formal notion of privacy (or confidentiality). Many definitions have been proposed but the classical notions of privacy are only two: indistinguishability and semantic security. If an encryption scheme has the indistinguishability property, then an adversary\textsuperscript{13} will be unable to distinguish if a given ciphertext is the encryption of $m$ or $m'$ where $m$ and $m'$ are two messages adaptively chosen by him. This is formalized by checking if the probability of success of an adversary in the game above is not significantly better than that of random guessing ($1/2$). If a cryptosystem is semantically secure it means that it is infeasible for an adversary to derive significant information about a message (plaintext) when given only its ciphertext and the corresponding public encryption key. These notions make sense only if we fix the “rules of the game” for the adversary, we usually consider: the chosen-plaintext attack (CPA) and the chosen-ciphertext attack (CCA). In the former the adversary, in the intent to accomplish his goal, can obtain only encryption of any chosen plaintext (this is naturally implied by the public-key model). In the CCA game the adversary gains the further possibility to obtain decryption of any chosen ciphertext that is different from the challenge one. The semantic security notion is only considered in the CPA game (semantic secure under CPA), conversely the indistinguishability is usually considered in both the contexts: indistinguishability under CPA (IND-CPA) and under CCA (IND-CCA) attacks. Semantic security and indistinguishability have been proven equivalent under the CPA.

Let’s come back to the deterministic encryption topic: the notions of security described above can be met only when the encryption algorithm is randomized. Indeed, if a deterministic encryption algorithm is used, the adversary is always able to gain information about the equality of plaintexts comparing the relative ciphertexts (that are uniquely determined). This implies that we need a different notion of privacy which should be more suitable for our specific case. Bellare et al. provide the PRIV notion of privacy in [4]: here we measure the capability of an hypothetic adversary $A$ in extracting information from a given ciphertext. The adversary is also entitled to influence the distribution of the encrypted plaintext. More formally an adversary is thought as a pair of algorithms $A = (A_m, A_g)$: $A_m$ generates a plaintext $x$ and some sort of related side-information $t$; $A_g$, given the public-key $pk$ and the encryption of $x$ (that is, $c = \text{Enc}(pk, x)$), is challenged to guess the side-information $t$. Both $A_m$ and $A_g$ are not allowed to exchange any information during the experiment: $A_g$ has to guess the side-information $t$ using only the encryption of something that $A_m$ has previously chosen. It is important to note that the distribution of the plaintexts generated by $A_m$ has to be independent from the public-key $pk$ otherwise, when using a deterministic encryption, it could be always possible to use the encryption of $x$ as side-information (that is, $t = \text{Enc}(pk, x)$).

The definition uses a two-experiment structure: a first real experiment $\text{Exp}_1$ where

\textsuperscript{12}Here we focus on schemes with a formal security proof in the reductionist style.

\textsuperscript{13}The adversaries that we consider are, if not differently specified, computationally-bounded.
the game follows the rules described above and a second ideal experiment $\text{Exp}_0$ where
the adversary is “cheated” giving to $A_g$ the encryption of a plaintext different from
the one related to the side-information to guess (this is accomplished by respecting
the distribution of the inputs given to the adversary). It should be clear that there is
no chance for the adversary to win in this ideal world. We define the advantage of $A$,
$\text{Adv}(A)$, as the difference between the probabilities of success in both the experiments,
that is:

$$
\text{Adv}(A) = \Pr(A \text{ wins in } \text{Exp}_1) - \Pr(A \text{ wins in } \text{Exp}_0)
$$

We say that an encryption scheme is PRIV secure if $A$’s advantage is negligible for
any computationally-bounded $A$. We stress the fact that this notion makes sense
only if the plaintext space has high min-entropy.

It is important to note that the PRIV notion does not strictly require a deterministic
encryption algorithm and in fact in Section 2.4.2 we report a PRIV secure randomized
encryption scheme that is able to satisfy our logarithmic research requirement.

In [4] this notion is also expressed in terms of plaintext vectors: $A_m$ generates a
vector of plaintexts and $A_g$ receives in input a vector of ciphertexts. They show that
this variant is strictly stronger than the one reported here (that they name as PRIV-1).
In what follows we will use the term PRIV to indicate the stronger vector-based notion.
Other further details about these definitions are available in [4].

Further works have investigated on these definitions. In [5] several notions of pri-
vacy are weighed up: a definition that uses the indistinguishability style (simple and
easy to use in proofs) and six variants in the semantic security style (like the already
reported PRIV notion, that coincides with one of the six). Without going in further
details, the authors formally prove that these notions are all equivalent among them,
namely they express the same privacy level.

In a different work Boldyreva et al. [12] propose a slightly weaker notion of security
for their deterministic encryption schemes (see Section 2.4.4). In that variant there is an
additional requirement on the distribution of plaintexts: in a vector of messages each
of them has to be a-priori hard-to-guess given the others; the original PRIV notion
requires only a space with high min-entropy. Nevertheless, the authors underline that
this is suitable for a variety of practical applications, for example the encryption of
high-entropy data containing social security or phone numbers. In such examples,
messages can depend on one another, e.g. share a common prefix, yet the foregoing
condition is satisfied. The definition is introduced both in the single message setting
and in the vector of messages one: they prove the equivalence of both the versions. In
the follow we will name their variant as weak-PRIV.

2.4.2 Constructions

There was been a comparable effort in designing secure and efficient deterministic
encryption schemes. The first three secure constructions have been proposed in the
Bellare et al.’s work [4]. A first generic and natural framework named “encrypt-with-
hash” that deterministically encrypts a plaintext $x$ by applying the encryption algorithm
of a randomized encryption scheme but using as randomness a hash of $x$. The second
construction is a deterministic variant of the well-known RSA-OAEP [8, 47] with the
desirable feature of being length-preserving (i.e., the length of the ciphertext equals the one of the plaintext). This scheme is named RSA-DOAEP and is presented below. The last construction “hash-and-encrypt” in [4] is the formalization of a scheme from the database literature: its encryption algorithm is randomized but it is an example of efficiently searchable encryption (ESE). These schemes can also be non-deterministic but permit fast (i.e. logarithmic time) search using an auxiliary deterministic function of the plaintext that can also be computed from the ciphertext (without the knowledge of the secret key): it serves as a “tag” for the storage process and for the records retrieve. This further scheme is presented in Section 2.4.2. All these three schemes are proven in [4] to be PRIV secure in the Random Oracle (RO) model.

In [5], the authors propose two general frameworks to build a deterministic encryption scheme proven to be PRIV secure in the standard (non-RO) model. The former scheme is based on general trapdoor one-way permutations and requires independent, uniformly distributed messages. The second construction allows deterministic encryption of independent, high min-entropy (but not necessarily uniform) plaintexts but it is based on trapdoor permutations that are one-way for high min-entropy distributions of inputs. Both the proposals have a PRIV security proof in the standard model but, unfortunately, they are too generic and the relative instantiations usually supply very poor efficiency. These solutions are not fully discussed in this work owing to their scarce practical interest.

Boldyreva et al. [12] have independently proposed several constructions for deterministic encryption schemes: they are secure in the standard model using the weak-PRIV privacy notion. Some of the available instantiations offer an acceptable level of efficiency. A brief introduction to their work is available in Section 2.4.4.

Encrypt-with-Hash This is a natural and generic way to convert a randomized encryption scheme to a deterministic version. The basic idea is to replace the random coins of an existent randomized encryption algorithm with the hash of the public-key concatenated with message to encrypt. Let (KG, Enc, Dec) be a randomized encryption scheme, we make explicit the use of randomness in Enc with the following syntax: Enc(pk, x; r) where x is the message and r are the used coins. We use an auxiliary hash function \( H: \{0, 1\}^* \rightarrow \{0, 1\}^* \) with the property that its outputs are suitable to be used as randomness for the Enc algorithm. The deterministic variant of the scheme will be a triplet (KG, DEnc, DDec) so defined:

- DEnc(pk, x): computes \( r \leftarrow H(pk||x) \) and \( y \leftarrow Enc(pk, x; r) \); returns the ciphertext y;
- DDec(pk, sk, y): decrypts the ciphertext using \( x \leftarrow Dec(pk, sk, y) \) then it checks if \( y \cong Enc(pk, x; r) \) where the randomness is recomputed as \( r \leftarrow H(pk||x) \); if it is true then the plaintext x is returned, otherwise the process fails.

This construction is proven to be PRIV-CCA secure in the RO model in [4] under the assumption that the underlying randomized scheme is IND-CPA and others minor but

\[\text{The RO model [44, 7] is a non-standard and ideal model where certain hash functions used inside the schemes are modeled as random functions available to the parties via queries to a random oracle. It is not widely accepted but it simplifies the proof-drafting task and usually brings to more efficient constructions.}\]
RSA-DOAEP  Bellare et al.’s work [4] introduces a deterministic variant of the well-known RSA-OAEP [8, 47] where in the encryption algorithm a part of the message is used in place of the randomness and an extra third round is applied. The proposed scheme encrypts messages of fixed length $n$ with $n > 2k_0$ and $n \geq k_1$ where $k_0$ and $k_1$ are structural constants.

The RSA encryption scheme is used as underlying trapdoor-permutation: an RSA key-generator outputs a public-key $pk = (e, N)$ and a secret-key $sk = (d, N)$ where $N$ is the public composite modulus that here we assume to be of length $k_1$; the trapdoor-permutation of a message $x$ is computed as $y \leftarrow RSA(pk, x) = x^e \mod N$ and the inversion is done using $x \leftarrow RSA^{-1}(sk, y) = y^d \mod N$. Some hash functions are required in the final scheme: $H_1, H_2 : \{0,1\}^* \rightarrow \{0,1\}^{k_0}$ and $R : \{0,1\}^* \rightarrow \{0,1\}^{n-k_0}$. A synthesis of the encryption and decryption operations is sketched in Figure 1. Practical and secure values for the constants could be: $k_1 = 1024$ and $k_0 = 160$.

**Figure 1: The encryption and decryption of RSA-DOAEP**
This variant is proven to be PRIV-CPA secure in the RO model assuming that the RSA trapdoor-permutation is one-way (without the knowledge of the secret key). Nevertheless, the plain scheme is not PRIV-CCA secure, indeed [4] reports a clear attack in the CCA setting. However, this solution becomes PRIV-CCA secure if combined with authentication in the “encrypt-then-sign” paradigm with a secure digital signature. In practical contexts of application, like the outsourced database management, the authentication is really often required.

**Encryption of Long Messages:** in real applications messages to encrypt are usually quite long: a secure and efficient technique to handle them in a public-key setting is “hybrid encryption”. Given a (long) message \( x \) to encrypt with a public-key \( pk \) the procedure draws out a new random session-key \( k \), then \( x \) is efficiently encrypted with a symmetric encryption algorithm using the fresh key \( k \); the resulting ciphertext is outputted jointly with the asymmetric encryption of \( k \) through the public-key \( pk \).

Unfortunately, applying this method to the RSA-DAOEP scheme implies the loss of its characterizing preserving-length feature. The authors in [4] claim that the scheme is still secure even if applied to long messages without the use of hybrid constructions (the length \( n \) of the input block is only lower-bounded by the constants \( k_0, k_1 \)): this assertion is supported by the presence of the underlying Feistel network that acts like an “all-or-nothing transform” [21].

### 2.4.3 Efficiently Searchable Encryption

The efficiently searchable encryption notion formalizes a way to achieve the desirable fast search support even when using a randomized encryption algorithm. This is accomplished via the use of an auxiliary deterministic “tag” computable both from the plaintext and from the ciphertext that encrypts it (without the knowledge of the secret key). In this way sensible data can be encrypted using a fully-secure randomized scheme (for example, a IND-CCA secure one).

With such tags it is still possible to use tree-based data structures enjoining the consequent fast (logarithmic) search: in an outsourced database application the server can compute the tag of a ciphertext to be stored in the database and use it to store the ciphertext appropriately in the data structure. To query a plaintext, the owner can compute and send its tag, which the server can look up in the data structure, returning any matches and associated data: the receiver can then decrypt these matches discharging eventual false-positives.

The handling of the tags is formalized through a pair of functions \( F, G \) that, respectively, compute the tag from the plaintext or from the corresponding ciphertext. An encryption scheme \((KG, Enc, Dec)\) is said to be a \( \delta \)-efficiently searchable encryption (ESE) scheme for \( \delta < 1 \) if there exist deterministic algorithms \( F, G \) such that the following properties hold:

1. **Perfect Consistency:** for any possible public-key \( pk \) and for any possible plaintext \( x \), the tag directly computed from the plaintext \( x \), that is \( t = F(x) \), is always equal to the one computed from any possible ciphertext containing \( x \), that is to any \( t = G(Enc(pk, x)) \);
2. Computation Soundness: for any pair of messages $x_0, x_1$ and for any possible public-key $pk$, the probability that the tag directly computed from $x_0$ ($t_0 = F(x_0)$) coincides to the one computed from an encryption of $x_1$ ($t_1 = G(\text{Enc}(pk,x_1))$) is at most $\delta$.

The above soundness property limits the number of false-positives located as well: on a database with $n$ elements we expect, in the average case, to have at most $\delta \cdot n$ false-positives on a query. Note as any deterministic encryption scheme is also a 0-ESE scheme.

**Hash-and-Encrypt** In the database literature an example of ESE construction is present: it is formally analyzed in [4] under the name of “hash-and-encrypt”. The idea is to append a (deterministic) hash of the plaintext to a ciphertext to enable efficient search. Let $(\text{KG}, \text{Enc}, \text{Dec})$ be a (randomized) public-key encryption scheme and $H : \{0,1\}^* \rightarrow \{0,1\}^l$ for some $l > 0$ be an hash function. The “hash-and-encrypt” scheme $(\text{KG}, \text{HEnc}, \text{HDec})$ is defined as:

1. $\text{HEnc}(pk,x)$: computes $h \leftarrow H(pk||x)$ and $c \leftarrow \text{Enc}(pk,x)$; returns $c||h$;
2. $\text{HDec}(pk,sk,c||h)$: retrieves the plaintext as $x \leftarrow \text{Dec}(pk,sk,c)$ and checks if $h = H(pk||x)$; if it is true then returns $x$ otherwise the procedure fails.

In this case the tag associated to a message $x$ with an encryption scheme using a public-key $pk$ is defined as $H(pk||x)$ which can be easily computed from both the plaintext and the ciphertext. If we consider the hash function $H$ acting like a randomized function (this is the case in the analysis carried on in [4] where $H$ is a RO) it is straightforward to verify that $(\text{KG}, \text{HEnc}, \text{HDec})$ is a $2^{-l}$-ESE scheme. The construction brings to a PRIV-CCA (respectively PRIV-CPA) secure scheme if the underlying encryption is IND-CCA (respectively IND-CPA) secure and some minor conditions\(^\text{16}\) hold. This is formally proven in the RO model in [4] under the condition of high min-entropy. If this is not the case it is still possible to achieve an acceptable level of privacy using the bucketization technique decreasing the length of the hash $l$: this augments the quantity of false-positives in the retrieving phase. Even though this rises the amount of data transferred toward the owner as result of his query, on the other side, in this way the privacy can be preserved even in presence of small message spaces.

### 2.4.4 Deterministic Encryption with Hidden Hash Mode

In [12] several novel weak-PRIV secure constructions are proposed in the standard model. These schemes offer a great level of security with the drawback of a more complex and, in this specific case, less intuitive design. This brings to less efficient implementations if compared with the schemes already seen.

Their descriptions require the introduction of several minor notions about hash functions. A family of hash functions is a pair $(K,H)$ where $K$ is the key-generation algorithm that draws out a specific function from the family: it is represented by the outputted key $k$. The algorithm $H$ applies the selected function to a message $x$ as $H(k,x)$.

\(^{16}\)It requires a negligible max public-key probability, i.e. that no public-key occurs with too high probability.
Several notions of security are involved: universal, pairwise-independent, collision-resistant and target-collision resistant\(^\text{17}\) hash functions. Furthermore, an hash function is said to be “lossy” if the size of the range is quite smaller than the one of the domain, i.e. a loss of information is implied and the function becomes un-invertible.

All the constructions exploit the features of some “lossy” trapdoor-functions which have got two operational modes (switchable through the way in which the key is generated): a “normal” mode where the function operates as a trapdoor-function with the possibility to be inverted using the proper secret information, and a “lossy” mode where the function becomes un-invertible due to loss of information. It is also required that these modes are indistinguishable looking at the key that describes the function itself.

Such kind of functions are employed as a deterministic encryption scheme making use of its “normal” mode\(^\text{18}\): they are named as encryption scheme with hidden hash mode (HHM). Boldyreva et al. [12] prove that the resulting encryption is weak-PRIV secure in the CPA setting if the functions generated in the “lossy” mode are also universal (hidden universal hash mode – HUHM).

To gain the CCA security level we need a more complex scheme: the notion of encryption with HHM is generalized to the one of all-but-one (ABO) deterministic encryption. It is a family of encryption schemes \((\text{KG}, \text{Enc}, \text{Dec})\) where the specific member is denoted by the \(b\) used as an extra parameter of all the three algorithms: on such family several properties are required where the branches act like the modes in the “lossy” TDF with the presence of a “default branch” that selects an un-invertible “lossy” function (see the original paper for further details). The last necessary building block is a TCR hash function: its output is used to select a branch in the family of ABO encryption functions\(^\text{19}\). The final weak-PRIV-CCA secure scheme is a triplet \((\text{KG}_{\text{cca}}, \text{Enc}_{\text{cca}}, \text{Dec}_{\text{cca}})\):

1. \(\text{KG}_{\text{cca}}\): invokes the key-generation algorithms for the TCR hash function, for the encryption with HHM and for the ABO encryption; the public-key is \(pk = (k_{\text{ter}}, pk_{\text{hhm}}, pk_{\text{abo}})\) and the secret key\(^\text{20}\) is \(sk = (sk_{\text{hhm}})\);

2. \(\text{Enc}_{\text{cca}}(pk, m)\): computes \(h ← H_{\text{ter}}(k_{\text{ter}}, m)\), \(c_{1} ← \text{Enc}_{\text{hhm}}(pk_{\text{hhm}}, m)\) and, using \(h\) as branch for the ABO family, \(c_{2} ← \text{Enc}_{\text{abo}}(pk_{\text{abo}}, h, m)\); the final encryption is \(h||c_{1}||c_{2}\);

3. \(\text{Dec}_{\text{cca}}(pk, sk, h||c_{1}||c_{2})\): the message is retrieved as \(m' ← \text{Dec}_{\text{hhm}}(sk_{\text{hhm}}, c_{1})\)

and it is checked if \(\text{Enc}_{\text{cca}}(pk, m') = h||c_{1}||c_{2}\): if is true the message \(m'\) is returned, otherwise the procedure fails.

---

\(^{17}\)An hash function is said to be universal if for any two distinct messages, the probability of collision is the same as if it was a random function. It is said to be pairwise-independent if for any pair of messages \(x_1 \neq x_2\) and any \(y_1, y_2 \in R\) (where \(R\) is the range of the function), the probability that \(H(k, x_1) = y_1 \land H(k, x_2) = y_2\), for any \(k\), is less or equal to \(1/|R|^2\). It is said to be collision-resistant if, given the function, any adversary has negligible probability to find a pair of different messages that collide through it. Finally, it is said to be target-collision resistant if any adversary is unsuccessful in outputting a pair of different colliding messages where at least one of the two has to be chosen independently from the function.

\(^{18}\)The “lossy” mode is only exploited in the proof of security.

\(^{19}\)The default branch has to be excluded from the range of the TCR function.

\(^{20}\)Note that the secret key \(sk_{\text{abo}}\) for the ABO encryption is discharged.
The scheme is proven to be secure if all the employed functions are universal.

In [12] secure instantiations for each building blocks (TCR, HHM, ABO) are suggested: their security is shown to hold under the standard Decisional Diffie-Hellman (DDH) assumption on the working group. The final result is an important proof of concept but it is rather inefficient.

To realize more performing instantiations, a slightly different construction is available: the basic idea is to apply a “pre-processing” step with an invertible pairwise-independent permutation to the three function invocations in the body of the previous encryption procedure. This trick permits to spare the extra requirement of universality that allows the use of more common and efficient instantiations. The authors propose novel Paillier-based lossy and ABO TDFs that are essentially length preserving and quite efficient; as “lossy” TCR primitive, one can use a cryptographic hash function like SHA256 (see [12] for further details).

2.5 Zero-Knowledge Sets

Imagine some party $P$ wants to commit to a set $S$, in a way such that any other party $V$ can “access” $S$ in a limited but reliable manner. By limited here we mean that $V$ is given indirect access to $S$, in the sense that she is allowed to ask only questions of the form “is $x$ in $S$?”. $P$ answers such questions by providing publicly verifiable proofs for the statements $x \in S$ (or $x \notin S$). Such proofs should be reliable in the sense that a cheating $P$ should not be able to convince $V$ that some $x$ is in the set while is not (or vice versa). At the same time, they should be “discreet” enough not to reveal anything beyond their validity.

The notion of Zero Knowledge Sets (ZKS) was recently introduced by Micali, Rabin and Kilian [67] to address exactly this problem. Informally, ZKS allow a prover $P$ to commit to an arbitrary (but finite) set $S$ in a way such that $P$ can later prove statements of the form $x \in S$ or $x \notin S$ without revealing any significant information about $S$ (not even its size!). As already pointed out in [67], the notion of zero knowledge sets can be easily extended to encompass the more general notion of elementary databases (EDB). In a nutshell, an elementary database is a set $S$ with the additional property that each $x \in S$ comes with an associated value $S(x)$. In the following we will refer to ZKS to include zero knowledge EDB as well.

2.5.1 Zero-Knowledge Sets from mercurial commitments

The solution by Micali et al. [67] is non interactive and works in the so called shared random string model (i.e. where a random string, built by some trusted third party, is made available to all participants) building upon a very clever utilization of a simple commitment scheme, originally proposed by Pedersen [78].

Commitment schemes: commitment schemes play a central role in cryptography. Informally, they can be seen as the digital equivalent of an opaque envelope. Whatever is put inside the envelope remains secret until the latter is opened (hiding property) and whoever creates the commitment should not be able to open it with a message that is

---

21The DDH assumption says that $(g^a, g^b, g^{ab})$ is indistinguishable from $(g^a, g^b, g^c)$ where $a, b, c$ are random and $g$ is a public group generator.
not the one originally inserted (binding property). Typically, a commitment scheme is a two phase procedure. During the first phase, the sender creates a commitment $C$, to some message $m$, using an appropriate commitment algorithm, and sends $C$ to the receiver $R$. In the opening phase the sender opens $C$ by giving $R$ all the necessary material to (efficiently) verify that $C$ was indeed a valid commitment to $m$.

**Construction** The ZK-Sets construction in [67] uses a variant of the Pedersen’s commitment together with a collision resistant hash function. Since Pedersen’s commitment relies on the intractability of the discrete logarithm, so does the construction in [67]. Later, Chase et al. [30] abstracted away Micali et al.’s solution and described the exact properties a commitment scheme should possess in order to allow a similar construction. This led to an elegant new variant of commitments, that they called mercurial commitment.

Informally, a mercurial commitment is a commitment scheme where the binding requirement is somewhat relaxed in order to allow for two decommitment procedures: an hard and a soft one. At committing time, the sender can decide as whether to create an hard commitment or a soft one, from the message $m$ he has in mind. Hard commitments are like standard ones, in the sense that they can be (hard or soft) opened only with respect to the message that was originally used to construct the commitment. Soft commitments, on the other hand, allow for more freedom, as they cannot be hard opened in any way, but they can be soft opened to any arbitrary message. An important requirement of mercurial commitments is that, hard and soft commitments should look alike to any polynomially bounded observer.

Using this new primitive, Chase et al. proved that it is possible to construct ZKS from a variety of assumptions (number theoretic or general)\textsuperscript{22}. Their most general implementation, shows that (non interactive) ZKS can be constructed, in the shared random string model, assuming non interactive zero knowledge proofs (NIZK) [11] and collision resistant hash functions [37]\textsuperscript{23}. Moreover, they showed that collision resistant hash function are necessary to construct ZKS, as they are implied by the latter.

In order to complete the picture, Catalano, Dodis and Visconti [28] gave a construction of (trapdoor) mercurial commitments from one-way functions in the shared random string model. This result showed that collision resistant hash functions are necessary and sufficient for non interactive ZKS in the shared random string model.

**More efficient constructions** All the constructions above are based upon the common idea to use an authenticated Merkle tree of depth $k$ where each internal node is a mercurial commitment (rather than the hash) of its two children. Very informally, to prove that a given $x \in \{0, 1\}^k$ belongs to the committed set $S$, the prover simply opens all the commitments in the path from the root to the leaf labeled by $x$ (more details about this methodology will be given later on). Thus the length of the resulting proof

\textsuperscript{22}More precisely, they require the mercurial commitment to be trapdoor as well. Very informally, this means that the scheme comes with a trapdoor information $tk$ (normally not available to anyone) that allows to completely violate the binding property of the commitment.

\textsuperscript{23}It is known that one can construct NIZK under the assumption that trapdoor permutations exist or under the assumption that verifiable random functions (VRF) exist [53, 68]. These two assumptions, however, are, as far as we currently know, incomparable.
is $k \cdot d$, where $d$ denotes the length of the opening of the mercurial commitment, and $k$ has to be chosen so that $2^k$ is larger than the size of any “reasonably” large set $S$. Assuming $k = 128$ and $d = O(k)$, as it is the case for all known implementations, this can lead to quite long proofs.

Catalano, Fiore and Messina in [29] addressed this efficiency problem with a new construction of ZKS that allows for much shorter proofs. Their solution relies on a new primitive that they call trapdoor $q$-Mercurial Commitment ($qTMC$, for short). Informally, $qTMC$s allow the sender to commit to a sequence of exactly $q$ messages $(m_1, \ldots, m_q)$, rather than to a single one, as with standard mercurial commitments. The sender can later open the commitment with respect to any message $m_i$ but, in order to do so successfully, he has to correctly specify the exact position held by the message in the sequence. In other words, trapdoor $q$-Mercurial commitments allow to commit to ordered sequences of $q$ messages.

Using the same paradigm of [67, 30], Catalano et al. [29] showed how to construct ZKS from $qTMC$s and collision resistant hash functions. The key point of this construction is that $qTMC$s allow to use a “flat” Merkle tree (i.e. with branching factor $q$ rather than two). This does not lead, by itself, to shorter proofs, because the opening of the $qTMC$ could trivially be linear in $q$. However Catalano et al. [29] proposed an implementation of trapdoor $q$-mercurial commitments that overcomes the above limitation and achieves much shorter proofs. More precisely the size of each hard opening still depends linearly on $q$, but the size of each soft opening becomes constant and completely independent of $q$.

Related work After Micali et al. [67] introduced the notion of zero knowledge sets, other works related to this topic appeared.

Ostrovsky, Rackoff and Smith [76] described a construction that allows a prover to commit to a database and to provide answers that are consistent with the commitment. Their solution can handle more elaborate queries than just membership ones. Moreover they also consider the issue of adding privacy to their protocol. However their construction requires interaction (at least if one wants to avoid the use of random oracles) and requires the prover to keep a counter for the questions asked so far.

Gennaro and Micali [49] introduced the notion of independent zero knowledge sets. Informally, independent ZKS protocols prevent an adversary from successfully correlate her set to the one of a honest prover. Their notion of independence also implies that the resulting ZKS protocol is non-malleable and requires a new commitment scheme that is both independent and mercurial.

Liskov [65] considered the problem of updating zero-knowledge databases. In [65] definitions for updatable zero knowledge databases are given, together with a construction based on verifiable random functions [68] and mercurial commitments. The construction, however, is in the random oracle model [7].

Very recently Prabhakaran and Xue [79] introduced the notion of statistically hiding sets (SHS) that is related but different than ZKS. Informally, SHS require the hiding property to hold with respect to unbounded verifiers. At the same time, however, they

---

24This is because $2^k$ is also an upper bound for the size of the set. Thus, to meet the requirements of ZKS it should not reveal anything about the cardinality of the set itself.
relax the zero knowledge requirement to allow for unbounded simulators.

Currently known constructions of zero knowledge sets are still far from being practical. We think that further research work has to be done in order to obtain more efficient implementations. In particular, following the work of Catalano et al., it would be very interesting to construct a trapdoor \( q \)-mercurial commitment that allows for openings whose length is independent of \( q \). Such a primitive would automatically lead to a ZKS extremely efficient.

2.6 Biometrics as Cryptographics Keys

A cryptographic application usually relies on uniformly distributed random strings for its secrets: they are usually stored in reliable and safe devices like smart-cards or portable hard-drives. These devices can be easily lost or stolen. The problem of keeping this hardware stuff always with us, in a safe way, might be overcome by exploiting specific biometric measures of our body as secrets. They naturally provide some desirable characteristics: always available, not easily duplicable and almost unpredictable. Possible examples are: fingerprints, voice or iris scans. Unfortunately the extracted random strings can not be directly used in our cryptographic applications: their distribution is not usually uniform and they do not get precisely reproduced each time they are measured.

Here we explore the required secure techniques to make possible the use of such nonuniform and unreliable secrets in cryptography. The right way to adapt a protocol originally based on the use of strong stored secrets to the use of biometric-derived random strings is briefly described.

2.6.1 Fuzzy Extractors

Dodis et al. [40] face this task with a formally secure approach that generalizes several prior works. Their definitions are built on a generic metric space with a proper metric function. The proposed instantiations instead rely on specific metrics: Hamming distance (i.e., the number of symbol positions that differ between two strings), set distance (size of the symmetric difference of two input sets of features) and editing distance (the number of insertions and deletions needed to convert one string into the other).

The main tool that they introduce is the fuzzy extractor: it extracts a uniformly random string \( R \) from a non-uniform input \( w \) in a noise-tolerant way. Noise-tolerance means that if the input changes to some \( w' \) but remains close, then the string \( R \) can be exactly reproduced. To assist in reproducing \( R \) from \( w' \), the fuzzy extractor outputs a non-secret string \( P \). It is important to note that \( R \) remains uniformly random even given \( P^{25} \). Such tool permits the natural use of the biometric measure as a cryptographic secret given the public access to \( P \). Like in the protocols of Section 2.6.3, \( P \) can be sent by the authentication server to the user during the exchange of messages: in this way the user does not need to bring any storage device with him. As shown in Section 2.6.2 this approach can fail in a more complex adversarial model.

---

25Strictly speaking, \( R \) will be \( \epsilon \)-close to uniform rather than uniform; \( \epsilon \) can be made exponentially small, which makes \( R \) as good as uniform for the usual applications.
Dodis et al. [40] introduce also the notion of *secure sketch*. It allows precise reconstruction of a noisy input, as follows: on input $w$, a procedure outputs a sketch $s$. Then, given $s$ and a value $w'$ close to $w$, it is possible to recover $w$. The sketch is secure in the sense that it does not reveal much about $w$: $w$ retains much of its entropy even if $s$ is known. Thus, instead of storing $w$ for fear that later readings will be noisy, it is possible to store $s$ instead, without compromising the privacy of $w$. A secure sketch, unlike a fuzzy extractor, allows for the precise reproduction of the original input, but does not address nonuniformity.

Secure sketches and extractors can be viewed as providing fuzzy key storage: they allow recovery of the secret key ($w$ or $R$) from a faulty reading $w'$ of the biometric measure $w$ by using some public information ($s$ or $P$).

Dodis et al. [40] show how fuzzy extractors can be naturally built out of secure sketches by using *strong randomness extractors*\(^2\) [73], such as, for example, universal hash functions. A general technique for constructing secure sketches from transitive families of isometries is present too: it is instantiated in several concrete constructions for the considered metrics (Hamming distance, set distance and editing distance). There are instantiations suitable to specific settings like: large or small universe of features for the set distance, long or short sampled strings, and so on. Many of them are quite practical.

### 2.6.2 Reusable Fuzzy Extractors

Boyen [19] has pointed out some security flaws about the use of the previous tools in real contexts. Fuzzy extractors as well as secure sketches define two phases: an *extraction procedure* that produces the public information related to the biometric input $w$ and, only for the fuzzy extractors, a uniformly distributed value $R$; a second *regeneration procedure* that, given a noisy input $w'$, permits the reconstruction of, respectively, the previously extracted value $R$ or the original input $w$. Although the repeated use of the regeneration function on many inputs is typically allowed, all the security analysis in [40] implicitly assume that no more than a single extraction is ever performed from any secret: this constraint raises several practical limitations. On the other hand Boyen et al. in [19] show two schemes, fully compliant to the security definitions from [40], that are completely insecure if the adversary is allowed to repeatedly invoke the extraction (under plausible conditions).

The main contribution of [19] is a more robust security model based on the stringent notion of *adaptive chosen perturbation* attacks, wherein the adversary may query an oracle to perform extractions and regenerations based on chosen perturbations\(^2\) of the attacked secret. Two flavors of attacks are allowed: if the adversary is only given an extraction oracle, we have an *outsider attack*; in the general case, but only for the fuzzy extractors, there is also a more powerful *insider attack* where the adversary has an

\(^{26}\) Randomness extractors are families of hash which “convert” a high entropy input into a shorter, uniformly distributed output.

\(^{27}\) A perturbation is a function specified by the adversary and applied by the extraction oracle to the secret prior to processing a query. The opportunity to formalize weaker random perturbation is also considered in [19].
additional access to a regeneration oracle\textsuperscript{28}. Generic constructions for these “reusable” fuzzy extractors and secure sketches are provided and they are proven to be secure against outsider attacks in the information theoretic sense (not specific assumptions are used). Furthermore he proves that some existing constructions already satisfy the same security level. The generic framework can be extended to obtain reusable fuzzy extractors that withstand the powerful insider attacks: unfortunately in this case the information theoretic security is no longer feasible and the use of the random oracle in the proof of security is necessary.

2.6.3 Authentication using Biometrics

The authors present in [19] a simple authentication protocol that exploits the notion of reusable fuzzy extractor to permit a user (with some specific biometric characteristic $w$) to remotely authenticate himself with multiple servers. The scheme requires the use of a Certification Authority (CA) but the user is not asked to (securely or insecurely) store anything — except for the biometric data. The user would be able to remotely authenticate using biometrics without revealing any of them to the server (even if it is malicious and/or colludes with other servers). The CA is trusted by all and it issues user certificates using the following procedure: the user uses an insider secure fuzzy extractor to extract from $w$ a uniformly distributed value $R$ with a public value $P$ and then he uses $R$ as a private key in some existentially unforgeable (UF-CMA) signature message deriving a corresponding public-key $pk_R$\textsuperscript{29}; as last step the CA issues a signature on the pair (“User ID”, $pk_R$) to certify it. At this point the user can distribute (i.e., send to the servers) the following data: $pk_R$, $P$ and the certificate (note that the public value $P$ is not signed by the CA). During the remote authentication the server sends a fresh random challenge $c$ and a copy of the user’s public value $P$ to the user; using the claimed public value $P$ and an approximation of the fuzzy secret $w'$, the original value $R$ is regenerated by the user and it is used as secret signing key to sign the challenge $c$. The resulting signature of $c$ is sent back to the server that verifies it using the CA-issued public-key $pk_R$. It is worth to note that the protocol does not require the user to “remember” anything than his fuzzy secret (and in particular he does not need to obtain the CA’s authentic public-key to verify a certificate). The protocol is proven secure against malicious manipulation of the sent public data $P$ exploiting the underlying reusable fuzzy extractor that is secure against insider attacks. The same idea is still applicable to other identification schemes.

In another work Boyen et al. [20] study the same problem of the authentication through biometrics offering some more challenging features. First the adversary is able to manipulate the queries to the oracles in a more general way (this means a stronger attack). Second, at the end of a successful execution of the protocol, both the parts are mutually authenticated (where in the previous one only the user is identified by the server and not vice versa) and, finally, the proof of security uses a fully standard model.

\textsuperscript{29}In this case there are some constraints on the possible queries: the adversary can adaptively ask regeneration on public outputs received through previous extraction queries, and receives the extracted uniform value. Furthermore these regeneration queries cannot be invoked on the public output related to the challenge value.

\textsuperscript{29}If $R$ is not a well formed private key, a good one is deterministically derived from $R$ first.
Their constructions are still secure against an adversary that is able to manipulate the network and to impersonate one of the parts (the server or the user); the presence of multiple servers is not considered in [20].

Boyen et al. propose two different constructions. The first one is a generic solution which protects against modification of the public data $P$ in any context in which secure sketches or fuzzy extractors are used; in other words, they are semantically equivalent to the one in [40] remaining secure even if the adversary is able to manipulate the public data before an extraction step. This permits to transform any protocol which is secure when the public data is assumed to be transmitted reliably on the network into one which is secure even when it might be tampered with. The idea is to use “robust” variants of the previous tools that insert enough redundancy in the public data allowing the user to detect any manipulation of them in the extraction phase. The privacy of the biometric secret remains untouched even under the disclosure of this “stuffed” public data.

The second construction is specific for the settings of remote (mutual) authentication and key exchange, where it offers some improvements to the generic solution.

3 Signal Processing Basics

Before describing the main tools that are usually adopted to process signals it is worth to pause briefly to define what we mean by signal.

Classical textbooks on analog signal theory use to define a signal as the values assumed by a physical quantity as a function of time\textsuperscript{30}. In the theory of analog signals, the time variable is assumed to be real, however here we are interested in discrete signals, hence we will consider only the values assumed by the physical quantity on a discrete set $T = \{t_i\}_{i \in \mathbb{N}}$.

Though apparently trivial and a bit fuzzy, the above definition highlights perfectly the two main characteristics of signals: i) signals are related to physical quantities; ii) the way the physical quantity varies with time is often more important than the assumed values themselves.

In the sequel we will give a list of basic signal processing operators, and provide a brief description of each of them. Due to obvious space limitation, the list is not an exhaustive one, and the description will be very brief (especially for more complex operators, whose description would need much more space than available here). Despite its brevity, the list may be useful to get a panoramic idea of which operators we would need to extend great part of the currently available signal processing tools into their encrypted domain counterparts. Whenever possible, we will also highlight the main difficulties that we expect to encounter in the implementation of the various tools in the encrypted domain.

\textsuperscript{30}Though this definition focuses on the 1-dimensional case, the extension to multidimensional signals like images or video is straightforward.
3.1 Signal Representation

As it is evident from the brief discussion of the previous section, signal values should be generally represented by real numbers. However, in many applications it is customary to represent signals as functions taking values in the set of complex numbers $\mathbb{C}$. Complex numbers provide a very elegant and powerful way of interpreting the properties of signals and hence in the following we will allow signals to be either functions going from $\mathcal{T}$ to $\mathbb{R}$ or functions going from $\mathcal{T}$ to $\mathbb{C}$. Note, however, that the physical counterpart of the signal will always be represented by a real number, be it the magnitude of the signal, its real or imaginary part.

Real numbers have dynamic structures, such that the length of their representation strings generally increases during the computations. On the other hand, traditional encryption schemes work with strings of limited lengths, where these lengths are governed by security parameters. This means that if the traditional encryption schemes, which are based on ring of integers or polynomials, are used to perform exact homomorphic operations on encryption of real numbers, the security parameter should be infinitely large. The problem of representing real numbers by strings of limited length is not specific to cryptography. Traditional computing systems use truncations of real numbers for storing and representation purposes. The results of computations on encrypted values must also be truncated, in order that further computation on these numbers be possible. In the following we describe several approaches to represent real values in computing systems and analyze their possible adaptation to homomorphic encryption schemes.

3.1.1 Floating Point Representation

A wide range of real numbers can be represented by few bytes, when the floating point representation is used. In the basis $\beta$, the floating point representation of the number $x$ is the tuple $(s_x, e_x)$, where $s_x$ and $e_x$ are called significand and exponent, respectively, such that:

$$x \approx s_x \beta^{e_x}$$  \hspace{1cm} (1)

The significand and exponent are selected in such a way that the approximation error in (1) is as small as possible. Floating point representations use bit-strings of lengths $l_s$ and $l_e$ to store the significands and exponents, respectively. We denote such a floating point representation by $\mathcal{FLEFT}(l_s, l_e)$. These values are then interpreted as the signed integer representation for $e_x$, whereas that of $s_x$ is a decimal value such that, per agreement, the decimal point is at the leftmost position. For example if $\beta = 10$ the value 0.0125 is shown by $s_{0.0125} = 125$ and $e_{0.0125} = -1$ since we have:

$$0.0125 = 0.125 \times 10^{-1}$$

It can be shown that when the value $x$ is represented by $(s_x, e_x)$ in $\mathcal{FLEFT}(l_s, l_e)$, then the absolute value of the error of approximation in (1) can be bounded by

$$|\beta^{e_x} - (l_e - 1)|$$  \hspace{1cm} (2)
and hence to decrease the error, it is desirable to make $e_x$ as small as possible. This results in the normalized representation of floating point values in which the leftmost digit of the significand is not zero.

Multiplication of floating point numbers is straightforward: multiply the significand, add the exponents, and normalize the result. Addition is, on the other hand, more complicated. To do addition we have to find the maximum of the two exponents, adjust the significands in such a way that both of them have this common exponent, and add the significands together. Finally normalize the result.

To perform these operations using homomorphic encryptions, we can use a multiplicative homomorphic encryption for the significands and an additive one for the exponents. In this way the multiplication will be easier, but still requires truncation of significands during the normalization and checking if the leftmost digit is zero. Addition of two values will need comparison of two encrypted values and, generally, truncating the results to bring them into the larger exponent and finally normalization.

### 3.1.2 Fixed Point Representation

A less space-efficient (but simpler) approach to represent real numbers is to use fixed point representation. This method, in its original form, is characterized by the scaling factor

$$Q = \beta^p$$

(3)

which is multiplied by the rational number $x$ to compute its fixed point representation, denoted by $X$, i.e.,

$$x \approx X/Q$$

(4)

Here $\beta$ is the basis used to represent numbers and $p$ is the number of digits on the right side of the floating point. The integer $X$ is computed from the rational number $xQ$, either by taking the integer part, or by truncation according to the system used. It is easy to show that the approximation error in (4) can be bounded by $Q - 1$. In contrast to (3) we let $Q$ to be any integer and denote the fixed point representation by the scaling factor by $FIP(Q)$. Again here, the values are represented by a limited number of digits. Usually, we assume that the magnitude of $x$ is smaller than one (this can be achieved through a proper normalization of the signal values), so that the values of the integer representation of numbers in $FIP(Q)$ are smaller than $Q$, which we denote by representation length.

Computation using fixed point representation is easier than in floating point. Addition of values is done by adding corresponding integers. Multiplication is also multiplication of integers followed by truncation. The truncation stage can be explained as below. Let $X$ and $Y$ be the representations of $x$ and $y$ in $FIP(Q)$, i.e., $X \approx xQ$ and $Y \approx yQ$ and hence,

$$XY \approx xyQ^2$$

(5)

whereas the valid representation of $xy$ is an integer $Z$ such that $Z = xyQ$. It is not difficult to see that a suitable candidate for $Z$ can be computed by dividing $XY$ by $Q$ and taking the quotient. Although even this simple task does not seem to be easily possible in encrypted domain, but according to the above statements, it seems that fixed point representation is more suitable for computations in encrypted domain.
Indeed, one of the strongest arguments against fixed point representations is the long representations when the same accuracy as floating point is going to be achieved. We argue here, that this argumentation cannot be used in encrypted domain, since here, we already need long representations to achieve high security. As one of the most famous examples for additive homomorphic encryptions we mention the Paillier encryption scheme. One of the most important parameters in this system is the integer \( N \), where the arithmetic is done in the ring \( \mathbb{Z}_{N^2} \) and the plaintext \( x \) is assumed to be smaller than \( N \). \( N \) is generally called the modulo number in Paillier encryption scheme. On the other hand, according to (5) we see that the length of \( N \) must be larger than two times representation length, i.e., \( Q \). These and the fact that fixed point arithmetic consists of arithmetic on integers show that \( \text{FIP}(Q) \) can be used together with the Paillier encryption with the modulo number \( N \) as long as \( Q^2 < N \).

### 3.1.3 Algebraic Integers

An alternative representation for signals can be obtained by approximating a real or complex value over a ring of algebraic integers [48]. Algebraic integers offer some advantages with respect to either integer or rational representations: for example, they are dense over \( \mathbb{C} \), meaning that the quantization does not require any scaling factor, and the dynamic range of the quantized values is dramatically reduced for a given error tolerance. In the literature, the computation of a linear transform over a ring of algebraic integers was first proposed in [34], aiming at reducing the dynamic range of the computations over a residue number system.

Let \( \omega = e^{2\pi j/R} \), \( R = 2^\mu \), \( \mu \geq 2 \) be a primitive \( R \)th root of unity. The subring of the field of complex numbers \( \mathbb{C} \) generated by \( \omega \) over the integers \( \mathbb{Z} \) is defined as [48, 34]

\[
\mathbb{Z}[\omega] = \left\{ \sum_{i=0}^{R/2-1} \alpha_i \omega^i \bigg| \alpha_i \in \mathbb{Z} \right\}.
\]

The set \( \mathbb{Z}[\omega] \) is usually referred to as the ring of algebraic integers. Both addition and multiplication can be defined over \( \mathbb{Z}[\omega] \) as combinations of the integer coefficients \( \alpha_i \) [34].

If \( \mu = 2 \), then \( \mathbb{Z}[\omega] \equiv \mathbb{Z}[j] \) is the ring of Gaussian integers, i.e., it contains numbers in the form \( a + jb \), \( a, b \in \mathbb{Z} \). Hence, approximating over \( \mathbb{Z}[j] \) is equivalent to rounding both the real and imaginary part to the nearest integer.

If \( \mu \geq 3 \), then \( \mathbb{Z}[\omega] \) is dense in \( \mathbb{C} \) [48, 23], meaning that any complex number can be arbitrarily approximated by a number in \( \mathbb{Z}[\omega] \) without requiring the multiplication by a scaling factor. In the following, we will refer to \( \mathbb{Z}[\omega] \) assuming that \( \mu \geq 3 \), which is sometimes referred to as the ring of cyclotomic integers [34]. Note that a real value can be approximated using only \( R/4 \) integer coefficients, since \( \Re\{\omega^i\} = -\Re\{\omega^{R/2-i}\} \).

In practical applications, we are interested only in those elements of \( \mathbb{Z}[\omega] \) which can be represented by bounded integer coefficients. These numbers will be indicated by the set

\[
\mathbb{Z}[\omega]_K = \left\{ \sum_{i=0}^{R/2-1} \alpha_i \omega^i \bigg| \alpha_i \in \mathbb{Z}, \ |\alpha_i| \leq K \right\}.
\]
Note that each number in $\mathbb{Z}[\omega]_K$ can be correctly represented by coefficients modulo $N$ if $N \geq 2K + 1$.

Let $x \in \mathbb{C}$ be a complex signal value. In order to process $x$ over $\mathbb{Z}[\omega]$, we need an approximation function $\psi_Q : \mathbb{C} \to \mathbb{Z}[\omega]_Q$ such that

$$s = \psi_Q(x), \quad ||s - x|| \leq \delta$$

where $\delta$ is the given tolerance. In [34], it has been shown that if $\delta = 2^{-b}$ (that is, $b$ bit precision is required), then

$$Q \approx \frac{1}{2}(C_R 2^{b-2} \pi^{1/2}) \approx 2^{4b/R}$$

(9)

where $C_R = R \sin(2\pi/R) / (\pi(1 - \cos(2\pi/R)))$. Hence, with respect to a fixed point representation requiring $Q_I \approx 2^b$, the dynamic range is reduced as $Q \approx Q_I^{1/R}$.

By using the above approximation function, each element of $x$ can be represented as a vector of $R/2$ bounded integer coefficients, that is, it is suitable for computations in the encrypted domain. Let us consider the algebraic integer representation of two complex values, defined as $s_1 = \psi_{Q_1}(x_1)$ and $s_2 = \psi_{Q_2}(x_2)$. The product of the two values in the algebraic integer domain is simply

$$u = s_1 s_2$$

(10)

where all operations are carried out in $\mathbb{Z}[\omega]$. It can be demonstrated that $u \in \mathbb{Z}[\omega]_K$, where $K \leq Q_1(RQ_2/4)$ [34]. Hence, the above implementation is feasible in the encrypted domain if $N \geq 2Q_1(RQ_2/4) + 1$.

The above results can be compared with the requirements of fixed point system, showing that the algebraic integer representation allows us to perform a greater number of operations in the encrypted domain. As a drawback, such a representation requires to encrypt $R/2$ values for each complex samples. Since the number of bits of the encrypted representation is fixed due to security requirements, this means that the algebraic integer representation would require $R/4$ times the bits of a fixed point representation when translated into the encrypted domain.

### 3.2 Point Operations

This class of operations collects two kinds of possible processing: **linear and non-linear point-wise processing** and **pointwise operations between two or more signals**: they will be described in the following, more details can be found in [60, 18].

**Linear and non-linear point-wise processing** A point operation is a function $H$ of a single variable, applied identically and separately to each sample, based on the value of the sample itself. At each coordinate $n$, the signal $f(n)$ is modified by the function $H$ leading to a new signal $g(n)$:

$$g(n) = H[f(n)].$$

(11)
Such operations are also called zero-memory operations. By contrast, local operations
calculate a new sample value by considering the values in a (typically small) local
neighborhood.

Among all the possible functions that can be applied following relation (11), the
simplest are the linear point operations, where $H$ is a linear function of the value as-
sumed by the signal $f$ at coordinate $n$:

$$g(n) = H[f(n)] = Sf(n) + O.$$  \hspace{1cm} (12)

Basically, the linear point operation modifies the original value $f(n)$ by means of a
multiplicative scaling (using a scalar factor $S$) and an additive offset (using a scalar
factor $O$). $S$ and $O$ has to be chosen appropriately, considering the range of values
taken by the original signal. In general, each sample of a digital signal is an inte-
ger value described by a fixed number of bits ($n_b$) that defines its dynamic, namely
$(0, d_{\text{max}} = 2^{n_b} - 1)$: for example, the samples of an audio signal are usually recorded
with 12 bits, thus defining the dynamic of each sample to be $(0, 4095)$. Since each sample
takes value in a positive range, $S$ is usually a positive value; in particular for $S < 1$
($S > 1$) all the values of the signal are reduced (increased) thus leading to a narrower
(broader) range for the values of $g(n)$. For example, applying $S > 1$ to an audio signal,
means to increase the volume of the signal. Regarding the additive offset, the factor $O$
can be both negative and positive: in the former case $g(n)$ will be a dimmed version
of the original $f(n)$, while in the latter case $g(n)$ will be a grown version of $f(n)$. As
opposed to multiplicative scaling, in this case the difference between adjacent sample
values remains unchanged, thus making the processing almost imperceptible. Let us
note that the output of the linear point operation must be an integer value included in
the origin dynamic $(0, d_{\text{max}})$. Hence, when $g(n)$ is not an integer, a round operation
is applied to take it to the nearest integer value; further more, when $g(n)$ falls outside
the allowable range (i.e. $g(n) < 0$ or $g(n) > d_{\text{max}}$), such saturated values are clipped
to the endpoint values (i.e. 0 or $d_{\text{max}}$). Note that the clipping function is once more a
pointwise operation, but it is non-linear and it will be briefly described in the following.

A particular case of operation described by equation (12) is the signal negative, that
uses both scaling and offset, with the particular factor: $S = -1$ and $O = d_{\text{max}}$:

$$g(n) = H[f(n)] = -f(n) + d_{\text{max}}.$$  \hspace{1cm} (13)

A clear use of such a processing regards digital images, e.g. when a negative is digitized
and the positive has to be viewed.

One of the most common linear point operations, is the so called linear contrast
enhancement or contrast stretch. The addressed problem is the following: the signal $f$
takes values only in a small sub-interval $(v_{\text{min}}, v_{\text{max}}) \in (0, d_{\text{max}})$, that is:
$v_{\text{min}} = \min_n[f(n)]$ and $v_{\text{max}} = \max_n[f(n)]$. Clearly, the smaller the interval
$v_{\text{max}} - v_{\text{min}}$, the worse the perceptibility of the differences between signal samples. It
is the aim of the contrast stretch operation to expand the dynamic $(v_{\text{min}}, v_{\text{max}})$ versus
$(0, d_{\text{max}})$ in order to exalt signal variations (in other words to enhance the contrast). In
particular, according to equation (12), the scaling factor $S$ and the offset $O$ are achieved.
by imposing that $H[v_{\text{min}}] = 0$ and $H[v_{\text{max}}] = d_{\text{max}}$:

$$S = \frac{d_{\text{max}}}{v_{\text{max}} - v_{\text{min}}},$$
$$O = -v_{\text{min}} \frac{d_{\text{max}}}{v_{\text{max}} - v_{\text{min}}}.$$  \hspace{1cm} (14)

When $H$ is a non-linear function, equation (11) represents a non-linear point operation. Some simple examples of non-linear functions are the absolute value, square, square-root functions, that are applied pointwise to each sample of the signal. A very commonly used non-linear function is the **logarithmic point operation**:

$$g(n) = H[f(n)] = \log[1 + f(n)],$$  \hspace{1cm} (15)

where the logarithm is applied to each sample value $f(n)$ and the unity is added to avoid the logarithm of zero. Such a processing has the effect to non-linearly compress the signal value range: the maximum dynamic $(0, d_{\text{max}})$ of $f$ becomes $(0, \log[d_{\text{max}}])$ for $g$. Since larger values are compressed much more severely than smaller values, this kind of operation is useful when important features of the signal are poorly perceptible due to their low values. Usually, the logarithm function is followed by a **linear contrast enhancement**, thus bringing back the dynamic $(0, \log[d_{\text{max}}])$ to the whole admissible range.

The previously described operation of contrast enhancement can be also achieved by means of non-linear point operations; an example of non-linear contrast enhancement is the **gamma correction** or often simply gamma; it is a nonlinear operation defined, in the simplest cases, by the following expression:

$$g(n) = H[f(n)] = [f(n)]^{\gamma},$$  \hspace{1cm} (16)

where with $\gamma < 1$ we usually refer to **gamma compression**, and with $\gamma > 1$ to **gamma expansion**. It is usually applied to luminance values of video or image signals when they are to be displayed on a monitor. In fact, the nonlinearity of the electron-gun current-voltage curve in CRT monitor systems produces a gamma expansion ($\gamma = 2.2$ is a typical value for PC monitor) on the signals before they are shown; hence, a gamma compression ($\gamma = 1/2.2$) is applied as a pre-distortion to input signals, in order to compensate the display device effect and thus letting the viewer to see the correct brightness.

In digital signal processing there is a fundamental operator, the **quantization**, that is used for approximating a range of continuous values to a set of discrete symbols. Considering an **uniform quantization** and a possible continuous value $v \in (0, V_{\text{max}})$ assumed by the signal, and a quantization step $s_q$, the quantized value $v_q$ is achieved as:

$$v_q = \left\lfloor \frac{v}{s_q} \right\rfloor \cdot s_q.$$  \hspace{1cm} (17)

The total number of quantization levels depend on the maximum value assumed by the signal $V_{\text{max}}$ and the chosen quantization step $s_q$: $[V_{\text{max}}/s_q]$. Any number of quantization levels is possible, though we often consider powers of 2, thus immediately referring to the number of bits $n_b$ used for describing a signal sample.
Sometimes it is useful to have a binary representation of the signal, through a binarization process, that, in the simplest case, can be achieved by describing each sample of the signal with the corresponding most-significant bit. Really, such an operation is performed by means of a different approach based on signal thresholding. Each value \( f(n) \) is compared to a threshold \( T \), and basing on this comparison the value of the processed \( g(n) \) is fixed to one of two possible values:

\[
g(n) = \begin{cases} 0, & \text{if } f(n) < T \\ 1, & \text{if } f(n) \geq T. \end{cases}
\]  

(18)

It is clear that the choice of the threshold \( T \) deeply influences the resulting signal \( g \).

In some of the operation described so far (e.g. multiplicative scaling), there is the possibility that some values (saturated values) of the processed signal \( g \) fall outside the admissible dynamic. In such cases, a clipping operation is applied, that is values greater than the maximum are clipped to the maximum (i.e. \( d_{\text{max}} \)), while values below the minimum are clipped to the minimum (i.e. 0):

\[
g(n) = \begin{cases} 0, & \text{if } f(n) \leq 0 \\ d_{\text{max}}, & \text{if } f(n) \geq d_{\text{max}} \\ f(n), & \text{otherwise.} \end{cases}
\]  

(19)

**Simple pointwise operations between two or more signals** So far we have considered operations working pointwise on a single signal. Now we consider pointwise operations working on two or more signals. Of course, in order to process corresponding samples of different signals, such signals must be characterized by the same number of samples. Arithmetic operations (such as sum and difference) can be simply applied to corresponding samples of two or more signals. Considering \( N \) signals \( f_1, f_2, ..., f_N \), the averaging operation is defined as:

\[
g(n) = \frac{1}{N}[f_1(n) + f_2(n) + ... + f_N(n)] = \frac{1}{N} \sum_{i=1}^{N} f_i(n), \text{ for all } n \text{ samples.} \]  

(20)

A typical use of such a processing is the reduction of noise. In fact, if we suppose to achieve \( N \) signals reproducing the same content and disturbed by an additive zero-mean noise, averaging the \( N \) signals allows to reduce the zero-mean noise component.

In order to evidence differences between two signals, the absolute difference operation can be applied to corresponding samples of the two signals:

\[
g(n) = |f_1(n) - f_2(n)|, \text{ for all } n \text{ samples.} \]  

(21)

Similarly, it is possible to consider the squared differences, that is:

\[
g(n) = [f_1(n) - f_2(n)]^2, \text{ for all } n \text{ samples.} \]  

(22)

Of course it is also possible to apply a pointwise product and quotient:

\[
g(n) = f_1(n) \cdot f_2(n), \text{ for all } n \text{ samples} \\
g(n) = \frac{f_1(n)}{f_2(n)}, \text{ for all } n \text{ samples.} \]  

(23)
A special case of the pointwise product is represented by the multiplication between two signals, one of which is a *masking signal*, that is a particular signal assuming only binary values 0 and 1:

\[ g(n) = f(n) \cdot m(n), \text{ for all } n \text{ samples} \]

\[ m(n) = \begin{cases} 
0, & \text{if the } n \text{ sample has not to be selected} \\
1, & \text{if the } n \text{ sample has to be selected} 
\end{cases}, \text{ for all } n \text{ samples.} \tag{24} \]

Such operation permits to extract from the original \( f \) only the selected samples indicated by the masking function; the signal \( g \) will be characterized by several null values and contain only the selected samples.

Another operation involving two or more signals considers a comparison among their values, for determining the *minimum* or *maximum* value or for ordering such values:

\[ g(n) = \max_i f_i(n), \text{ for all } n \text{ samples} \]

\[ g(n) = \min_i f_i(n), \text{ for all } n \text{ samples.} \tag{25} \]

Two interesting applications of pointwise operation between more signals regard the transformations between different color systems in the image/video framework and the conversion from stereo to mono signals in the audio framework. In both the cases the signals reproducing the color and the audio components respectively, are combined together, sample by sample. Referring to color images (or videos), all perceived colors can be reproduced by mixing an appropriate set of 3 independent values referring to 3 primary colors of a given color system (e.g. \( RGB \), \( XYZ \), \( HVS \)). For describing a color image is thus necessary to provide three samples (color channels) for each pixel, which are interpreted as coordinates in some color space. It is possible to calculate transformations between different color systems in order to describe the same color image by means of different color space. To give an example of such point operation, let us consider the relations from the \( RGB \) color space to the \( XYZ \) color space:

\[
\begin{bmatrix}
X \\
Y \\
Z
\end{bmatrix} =
\begin{bmatrix}
0.490 & 0.310 & 0.200 \\
0.177 & 0.813 & 0.011 \\
0.000 & 0.010 & 0.990
\end{bmatrix}
\begin{bmatrix}
R \\
G \\
B
\end{bmatrix} \tag{26}
\]

where \( R, G, B \) and \( X, Y, Z \) are the color coordinates for each sample of the image. Regarding the audio case, from 2 stereo signals it is possible to achieved 1 mono audio signal, by combining (e.g. for simplicity by averaging them) sample by sample the two signals.

**Pointwise operations in the s.p.e.d framework** By considering all the pointwise operations previously described, it can be pointed out that if the interest is to perform such processing in the encrypted domain, several main operations should be computed in the encrypted domain:

- product and sum by scalar values;
• logarithm and power functions;
• comparisons;
• sum, difference and product between signals;

Indeed, at the moment the only possibility is to adopt some homomorphic cryptosystem that allow to compute in the encrypted domain only sum and difference between signals and product by scalar values. For such operators it will likely be necessary to introduce some form of interaction between the computation players.

3.3 Linear Filtering

This class refers to every linear operation affecting both the values of a signal and its temporal structure.

Linear filtering operations can be either time-invariant or time-varying. In the case of time-invariant systems, if the sequence \( f(n) \) produces the output \( g(n) \), then the sequence \( f(n - k) \) will produce the output \( g(n - k) \), for every \( k \). Examples of time-invariant linear filtering operations are convolution, FIR and IIR filters. Examples of time-varying linear filtering operations are decimation, interpolation, and filterbanks.

A linear filter is said to be stable if a bounded sequence \( f(n) \) (i.e., \( |f(n)| < A, \forall n \)) always produces a bounded output \( g(n) \).

A linear filter is said to be causal (anticausal) if the value of the output \( g(n) \) depends only on samples \( f(k) \) so that \( k \leq n \) (\( k > n \)).

More details about linear signal processing operators can be found in [75, 82].

Convolution

The convolution between two sequences is defined as

\[
g(n) = \sum_{k=-\infty}^{\infty} h(k) f(n - k). \tag{27}
\]

The convolution is the fundamental block of every linear filtering operation. In particular, any linear time-invariant filtering of the sequence \( f(n) \) can be expressed by (27), where \( h(k) \) is the impulse response of the filter. Note that if one of the sequences is finite the convolution is equivalent to a sequence of scalar products.

Finite Impulse Response (FIR) Filters

A FIR filter is given by

\[
g(n) = \sum_{k=0}^{N-1} h(k) f(n - k). \tag{28}
\]

The filter is defined by the finite sequence \( h(k) \). The \( N \) values composing \( h(k) \) are also referred to as filter coefficients. A FIR filter is always stable and can be always made causal through a time shift.
Infinite Impulse Response (IIR) Filters  
An IIR filter is given by the following difference equation
\[ g(n) = \sum_{k=0}^{M-1} b(k)f(n - k) - \sum_{k=1}^{N} a(k)g(n - k). \]  \hspace{1cm} (29)

It can be shown that the above equation is equivalent to a convolution for an infinite duration \( h(k) \). Usually the number of coefficients \( b(k) \) and \( a(k) \) of an IIR filter is much less than that of a FIR filter. The stability and causality of an IIR filter depend on the properties of \( b(k) \) and \( a(k) \). In particular, an IIR filter is stable and causal if the roots of the polynomial \( P(z) = 1 + \sum_{k=1}^{N} a(k)z^{-k} \) lie inside the unit circle. For example, if \( N = 1 \) an IIR filter is stable and causal if and only if \( |a(1)| < 1 \).

Decimation and Interpolation  
These operations are used to change the time resolution of a digital sequence. A decimator is defined as
\[ g(n) = f(nM) \]  \hspace{1cm} (30)

where \( M \) is the decimation factor. An interpolator is defined as
\[ g(n) = \begin{cases} f(n/L) & \text{if } n \mod L = 0 \\ 0 & \text{elsewhere} \end{cases} \]  \hspace{1cm} (31)

where \( L \) is the interpolation factor. Both decimation and interpolation are basic operations which alter only the structure of a signal. Smooth resampling of digital sequences is usually obtained by combining the above operators with FIR/IIR filters. The interpolator and the decimator are also used to implement multiplexers (multidimensional to monodimensional signal conversion) and demultiplexers (monodimensional to multidimensional signal conversion), respectively.

Filterbanks  
A filterbank is a signal processing structure composed by an analysis part and a synthesis part. The analysis part is a demultiplexer followed by a bank of FIR/IIR filters, where each filter processes a different output of the demultiplexer. The synthesis part is a bank of FIR/IIR filters followed by a multiplexer. By imposing suitable conditions on the filters, it is possible to design analysis/synthesis pairs yielding perfect reconstruction, i.e., the analysis part is perfectly inverted by the synthesis part [86].

Filterbanks have been used in many audio coding applications. Moreover, linear transforms, which are described more in detail in the next section, can be viewed as particular cases of a filterbank.

A filterbank can be also modeled as a multidimensional linear filter, that is, an operation of the following kind
\[ g(n) = \sum_{k} H(k)f(n - k) \]  \hspace{1cm} (32)

where \( f(n) \) models a multidimensional signal and the matrices \( H(k) \) are the multidimensional filter coefficients.
3.4 Linear Transforms

Strictly speaking, linear transforms are particular cases of a linear filter operation. However, since they are fundamental building blocks in many signal processing applications, they will be treated as a separate class of operations.

A one dimensional linear transform is defined as

\[ F(k) = \sum_{n=0}^{N-1} C_k(n) f(n) \]

(33)

where \( F(k) \) are called transform coefficients and \( C_k(n) \) define the transform basis. Signal processing applications are mainly interested in orthogonal transforms, i.e., if we define the matrix \( [C]_{nk} = C_k(n) \), then \( C^H C = K I \), where \( [C^H]_{kn} = C_k^*(n) \). The inverse of an orthogonal transform is

\[ f(n) = \frac{1}{K} \sum_{k=0}^{N-1} C_k^*(n) F(k) \]

(34)

Moreover, if \( C^H C = I \) the transform is also unitary. Unitary transforms are important because they preserve the energy of a signal, i.e., \( \sum_n |f(n)|^2 = \sum_k |F(k)|^2 \).

A list of the most widely used transforms is given in the following [60]:

- **DFT**: \( C_k(n) = \frac{1}{\sqrt{N}} e^{-j2\pi nk/N} \);

- **DCT**: \( C_k(n) = \begin{cases} \frac{1}{\sqrt{N}} & k = 0 \\ \frac{1}{\sqrt{N}} \cos \frac{(2n+1)k}{2N} & k \neq 0 \end{cases} \);

- **Hadamard**: is defined recursively as \( C_1 = \frac{1}{\sqrt{2}} \begin{bmatrix} 1 & 1 \\ 1 & -1 \end{bmatrix} \), \( C_N = \frac{1}{\sqrt{2}} \begin{bmatrix} C_{N-1} & C_{N-1} \\ C_{N-1} & -C_{N-1} \end{bmatrix} \);

- **Karhunen-Loeve (KL)**: this transform depends on the autocorrelation matrix of \( f(n) \), defined as \( [R_f]_{nk} = E\{f(n)f^*(k)\} \), where \( E\{\} \) denotes the expected value. In practical applications, the expected value is usually estimated by using the statistical analysis tools described in Section 3.6. Given the eigendecomposition of the autocorrelation matrix \( \Phi^H R_f \Phi = \Lambda \), the KL transform is defined by \( C = \Phi^H \).

DFT, DCT, and Hadamard transforms admit a fast implementation with complexity \( O(N \log N) \). The KL is the transform achieving optimal decorrelation property. However, for some classes of signals the decorrelating performance of the KL is well approximated by one signal-independent transform (for example, images and DCT).

For what concerns multidimensional transforms, applications often consider separable transforms, defined by \( C_{k_1,\ldots,k_p}(n_1,\ldots,n_p) = C_{k_1}(n_1) \cdots C_{k_p}(n_p) \). Separable transforms are implemented by applying in sequence a monodimensional transform to each dimension of the signal.
Wavelets  Although wavelets do not fit into the linear transform model considered above, they are usually considered as an evolution of the classical linear transforms in many signal processing applications. The wavelet transform finds its mathematical justification in the multiresolution representation theory [66, 39], in which bases for the space of square summable functions satisfying certain multiresolution properties are constructed based on dilations and translations of a lowpass scaling function \( \phi(t) \) and a bandpass wavelet function \( \psi(t) \). The link between wavelets and signal processing is given by the following dilation equations

\[
\phi(x) = \sqrt{2} \sum_k h_0(k) \phi(2x - k)
\]

\[
\psi(x) = \sqrt{2} \sum_k h_1(k) \phi(2x - k)
\]

which allows us to express functions at coarser scales as a linear combination of functions at a finer scale. Based on the above equations, a multiresolution analysis of a signal \( f \) can be performed with a filterbank composed of a lowpass analysis filter \( h_0(k) \) and a highpass analysis filter \( h_1(k) \)

\[
a_{j+1}(n) = \sum_k h_0(k) a_j(2n - k)
\]

\[
w_{j+1}(n) = \sum_k h_1(k) a_j(2n - k)
\]

where \( a_j(k) \) is the approximation of the signal \( f \) at the \( j \)th step, whereas \( a_{j+1}(k) \) and \( w_{j+1}(k) \) are the approximation and wavelet coefficients at the \( (j + 1) \)th step. If the approximation of \( f \) at the first step is given by its samples \( f(n) \), then the wavelet transform of \( f(n) \) (which is often referred to as wavelet decomposition) is implemented by recursively applying to \( f(n) \) the above two-channel filterbank. The outputs of the wavelet transform are the wavelet coefficients obtained at each step plus the approximation coefficients of the last step. If the wavelet and scaling functions realize an orthogonal decomposition of the signal, the wavelet transform is also orthogonal and can be perfectly inverted. Perfect reconstruction can be also achieved if the orthogonality constraint is relaxed, leading to biorthogonal wavelets [32].

The wavelet transform usually provides a very compact representation of a signal, that is, a signal can be approximated very well by few of its wavelet coefficients. Thanks to the multiresolution property, the compactness of the representation is not affected by the presence of features having different scales (for examples, edges and textures in images), which is not true in the case of DCT and DFT. For such reasons, the wavelet transform is widely used in signal compression, denoising, and restoration.

3.5 Non Linear Filters

This class consists of all the operations affecting both the structure and the values of a signal which can not be expressed as linear operators. Non linear filters are usually building blocks of higher level processing operations (e.g. segmentation, features extraction). Some of the most common non linear operations used in signal processing are described in the following. Details can be found in [60, 18].
Median Filters

The median filter is defined as

$$g(n) = \text{median}\{f(n - k), k \in W\}$$ (39)

where $W$ is a suitably chosen window. The algorithm for median filtering requires arranging the signal samples in increasing or decreasing order and picking the middle value. Usually the window $W$ contains an odd number of samples. The median filter is useful in removing isolated noisy samples while preserving the overall signal structure.

Dilation, Erosion, and Majority Filters

Such operators are usually defined on binary signals. The dilate filter is

$$g(n) = \text{OR}\{f(n - k), k \in W\}$$ (40)

the erode filter is

$$g(n) = \text{AND}\{f(n - k), k \in W\}$$ (41)

and the majority filter is given as

$$g(n) = \text{MAJ}\{f(n - k), k \in W\}.$$ (42)

where MAJ return the value of the majority of the samples in the set.

The above operators are used to remove isolated values from the signal structure. Often, a combination of erode and dilate is used. In particular, the open filter is erode followed by dilate, whereas the close filter is dilate followed by erode. The majority filter is the binary counterpart of the median filter.

Both dilate and erode filters can be respectively extended to non binary signals as follows

$$g(n) = \text{dilate}[f(n), W] = \max_{k \in W} \{f(n - k)\}$$ (43)

$$g(n) = \text{erode}[f(n), W] = \min_{k \in W} \{f(n - k)\}.$$ (44)

Morphological Filtering

Morphological filtering is usually implemented by applying dilate, erode, majority, and their combinations to either binary or non binary multidimensional signals. One of the key points in morphological filtering is the shape of $W$, which allows the filter to capture different properties of the geometrical structure of a signal. The set of points contained in $W$ is also called structuring element of the morphological filter.

Morphological filtering is widely used in image processing and computer vision applications. In image processing, two commonly used examples for the structuring element $W$ are the disk shaped window and rotated version of a line segment at different angles, which respectively smooth the contours and enhance directional features [18]. Some common morphological filters used in image processing are:

- **Hit-Miss**: usually defined on binary images. Given two disjoint structuring elements $W_{fg}$ and $W_{bg}$, the hit-miss filter returns 1 if $f(n - k) = 1$, $\forall k \in W_{fg}$ and $f(n - k) = 0$, $\forall k \in W_{bg}$, otherwise it returns 0. It can be expressed as $g(n) = \text{erode}[f(n), W_{fg}] - \text{dilate}[f(n), W_{bg}]$. Detects particular shapes in binary images;
• **Open**: $g(n) = \text{dilate}[\text{erode}[f(n), W]]$. Smooths contours and remove isolated pixels;

• **Close**: $g(n) = \text{erode}[\text{dilate}[f(n), W]]$. Blocks up narrow channels and fills holes;

• **Boundary**: $g(n) = f(n) - \text{dilate}[f(n), W]$. Gives the boundary pixels of an object.

### 3.6 Statistical Analysis

Statistical analysis is a very useful operation to be carried out on signals. As a matter of fact, the statistics of a signal provide many useful information about it and are a fundamental brick in many signal processing systems. As we will further observe at the end of this section, statistical operators are not intrinsically different than the operators described so far, all of them can be obtained by a proper concatenation of the basic operators described previously. However, due to their importance and to the their special role in many applications, we preferred to dedicate to them a specific section.

In the sequel we list the most commonly used statistical operators. For sake of simplicity we will refer to 1D signals, however the extension to multidimensional signals like images, videos or volumes is straightforward.

**Mean value** This is the simplest statistical operator. It computes the temporal (or spatial in the case of still images) average of the signal itself and is defined as:

$$\text{Mean}(f) = \overline{f} = \frac{1}{N} \sum_{n=1}^{N} f(n), \tag{45}$$

where $N$ indicates the number of samples contained in the signal.

**Mean absolute value** Whenever it is necessary to evaluate the average magnitude of signal samples, the mean must be substituted by an operator which does not compensate negative and positive values. The simplest operator to do so is the mean absolute value, defined as:

$$\text{MAE}[f] = \frac{1}{N} \sum_{n=1}^{N} |f(n)|. \tag{46}$$

If $f(n)$ is a complex signal, then the absolute value should be replaced by the norm.

**Second order moments** Second order moments play a crucial role in signal processing since they are closely related to fundamental physical quantities such as power and energy. The quantities that are more commonly used are the following.
Energy. The energy of a signal\textsuperscript{31} is defined as the squared Euclidean norm of the vector representing the signal.

\[ E_f = \sum_{n=1}^{N} f(n)^2. \] (47)

If the energy is divided by the number of samples the signal consists of, we obtain the (average) power of the signal:

\[ P_f = \frac{1}{N} \sum_{n=1}^{N} f(n)^2. \] (48)

Another useful quantity is the central second order moment of the signal. It gives the average squared deviation of the signal from its mean value.

\[ s_f = \frac{1}{N} \sum_{n=1}^{N} [f(n) - \overline{f}]^2. \] (49)

The quantity \( s_f \) (as the other temporal averages defined previously) should not be confused with its probabilistic counterpart. More specifically, if we model the signal \( f(n) \) as a stochastic process, then we can compute the expected value of \( f(n) \) or its variance \( \text{Var}[f(n)] \). In this framework, the temporal moments defined above can be seen as estimators of the true probabilistic moments. If the stochastic process \( f(n) \) is ergodic, then the temporal and probabilistic averages can be exchanged, however they are conceptually different quantities. With regard to equation (49), the quantity \( s_f \) can be seen as an estimate of the signal variance, however given that instead of the probabilistic mean of the signal it uses the temporal mean \( \overline{f} \), the estimate is biased. For an unbiased estimate equation (49) should be replaced by

\[ s_f' = \frac{1}{N - 1} \sum_{n=1}^{N} [f(n) - \overline{f}]^2. \] (50)

or

\[ s_f'' = \frac{1}{N} \sum_{n=1}^{N} [f(n) - \mu_f]^2. \] (51)

if the expected value (\( \mu_f = E[f] \)) of the signal is known.

**Error statistics** In many cases the statistical analysis is used to measure the difference between two signals, say \( e(n) = f(n) - g(n) \). This is a very useful operation when one of the two signals can be seen as an estimate or a distorted version of the other. In this case, the difference statistics assume the meaning of error statistics and are very useful to estimate how large the error is. To do so, the statistical operators are

\textsuperscript{31}Here we focus on deterministic signals; if we consider the signal as a random process, the following definitions are valid only for a particular realization of the process (i.e. a particular signal).
directly applied to the difference signal. Among the error statistics the most frequently
used are the MSE (mean square error) and the MAE (mean absolute error):

$$\text{MSE} = \frac{1}{N} \sum_{n=1}^{N} [f(n) - g(n)]^2. \quad (52)$$

$$\text{MAE} = \frac{1}{N} \sum_{n=1}^{N} |f(n) - g(n)|. \quad (53)$$

The energy of the error signal can be conveniently compared with the energy of the
original signal, to give an idea of how important the error is with respect to the magni-
tude of the signal under analysis. This is the role of two of the most commonly used
quantity in signal processing, the SNR (signal to noise ration) and PSNR (peak signal
to noise ratio). The SNR is defined as:

$$\text{SNR} = \frac{\sum_{n=1}^{N} f(n)^2}{\sum_{n=1}^{N} e(n)^2}. \quad (54)$$

When a reference signal against which the reference error can be compared does not
exist, it is customarily to compare the error against the maximum possible power that
the signal may have. By assuming that the signals are represented by $n_b$ bits, such a
maximum power is $(2^{n_b} - 1)^2$. If, for instance, $n_b = 8$, as it is usually the case for still
images, the maximum signal power will be $255^2$, yielding the popular PSNR formula:

$$\text{PSNR} = \frac{\sum_{n=1}^{N} 255^2}{\sum_{n=1}^{N} e(n)^2} = N \cdot \frac{255^2}{\sum_{n=1}^{N} e(n)^2}. \quad (55)$$

**Correlation** The autocorrelation function of a signal is another signal estimated as
the expected value of the product of two samples of the signal itself, defined as:

$$R_{ff}(k) = \frac{1}{N} \sum_{n} f(n)f(n + k). \quad (56)$$

The autocorrelation of a signal is a very important quantity, that is closely related to the
frequency analysis of the signal. In fact, the Fourier transform of $R_{ff}$ gives the energy
spectrum of the signal. Moreover $R_{ff}(0)$ is nothing but the energy (power) of the
signal. When two signals are considered, the cross correlation among the signals must
be considered:

$$R_{fg}(k) = \frac{1}{N} \sum_{n} f(n)g(n + k). \quad (57)$$

In this case $R_{fg}(0)$ (often called itself correlation between the signals) has a very particular
meaning, since it gives a precise measure of the similarity between the signals

\[\text{For signals with infinite energy the autocorrelation is defined as } \lim_{N \to \infty} \frac{\sum_{n=-N/2}^{N/2} f(n)f(n+k)}{N} \text{ and its transform gives the power spectrum of the signal.}\]
A normalized measure of the similarity is given by the correlation coefficient defined as:

$$\rho_{fg} = \frac{\sum_n f(n)g(n)}{\sqrt{E_f E_g}}$$

(58)

The correlation coefficient is always lower than 1 and reaches 1 if and only if \(f(n) = g(n)\).

**Relative frequencies** Let us assume that the signal \(f(n)\) takes values in a finite alphabet \(F\) (e.g. all the integer numbers in the 0-255 interval). It is very instructive to compute the relative frequencies of the various elements of \(F\) within the signal \(f(n)\). Such an information is usually stored in a function \(h(f) : F \rightarrow N\) defined as follows

$$h(f) = \sum_{n=1}^{N} 1\{f(n) = f\}$$

(59)

where \(1\{\}\) denotes the indicator function which is equal to 1 if the condition contained in the brackets verifies and 0 otherwise. The function \(h(f)\) is usually called the histogram of the signal and is an empirical approximation of the probability density function according to which the signal \(f(n)\) is distributed (in this framework the signal \(f(n)\) is seen as a stochastic process). The histogram plays a fundamental role in image processing application, since it gives a precise idea of the distribution of the gray levels within the image. For instance, a very compact histogram is often an indication of a low contrast, whereas a bimodal histogram (i.e. an histogram with two distinct and well defined peaks) indicates the possible presence within the image of two different regions (objects) characterized by different brightness.

**Entropy** The entropy of a source (or a stochastic process) is the funding concept of information theory. Given a stationary, memoryless, process \(f(n)\), whose samples are distributed according to a probability mass function \(p(f)\), its entropy is defined as the expected value (through \(p(f)\)) of \(\log p(f)\), that is:

$$H(f) = \sum_{f \in F} p(f) \log_2 \frac{1}{p(f)}$$

(60)

where we have assumed that the signal takes values in the finite alphabet \(F\). The estimation of the above quantity on a single realization of the process, i.e. on a single signal, is called the empirical entropy of the signal (sometimes and somewhat erroneously called the entropy of the signal). The computation of the empirical entropy passes through the computation of the signal histogram as follows. First the histogram is normalized so that its entries can be interpreted as probabilities

$$h'(f) = \sum_{n=1}^{N} \frac{1}{N} 1\{f(n) = f\}$$

(61)

then equation (60) is applied by replacing probabilities with relative frequencies:

$$\hat{H}(f) = \sum_{f \in F} h'(f) \log_2 \frac{1}{h'(f)}.$$
**Order statistics** The class of order statistics are a particular class of operators that permit to analyze signals without considering (or reducing the effect of) samples with outlying values (the so called outliers). This is not the case for standard statistics, e.g. the mean, whose value may be altered by the presence of even one single sample assuming an extremely high value. Order statistics permit to avoid the above problem. The first step in the definition of order statistics consists in ordering the samples of the to be processed signal in ascending order, let us called the signal with re-ordered samples $f([n])$, where the square brackets indicate that the sequence has been re-indexed. An order statistics is a weighted linear combination of the values of $f([n])$. The most famous order statistic is with no doubt the median of a signal defined as:

$$\text{Med}[f(n)] = f([(N + 1)/2]) = \sum_n a(n)f([n])$$

(63)

where $a(n) = 0$ for every $n$ except $n = (N + 1)/2$ for which $a(n) = 1$ and where we have assumed that $N$ is an odd number. In practice the median is the central point of the re-ordered signal $f([n])$. Other order statistics include the maximum and the minimum of a signal:

$$\text{Max}[f(n)] = f([N])$$

(64)

$$\text{Min}[f(n)] = f([0])$$

(65)

and the alpha trimmed mean

$$\alpha_{\text{mean}}[f(n)] = \frac{\sum_{n=\lfloor \alpha N \rfloor}^{N-\lfloor \alpha N \rfloor} a(n)f([n])}{N - 2\lfloor \alpha N \rfloor}$$

(66)

whose effect is that of computing the mean of the signal after that the largest and smallest $\lfloor \alpha N \rfloor$ values of the signal have been discarded.

**Computing statistics in the encrypted domain** With regard to the possibility of computing the statistics described in the previous paragraphs directly in the encrypted domain, we can notice that most of the statistics we described can be seen as special cases of the operators described in the previous sections. For instance, from a mathematical point of view, computing the mean of a signal is nothing but a linear filter, where all the samples of the signal contribute to calculate a single value (the mean) at the output of the filter (whose coefficients, then, are constantly equal to $1/N$. The same applies to the other statistics that can be seen as the mean value of the signal obtained by applying a non-linear sample wise operator to the original signal. For instance, the average power of a signal $f(n)$ is nothing but the mean value of $f^2(n)$, and can be obtained by cascading the point-wise square operator and a linear filter. In the same way, the median value of a signal can be obtained by first ordering the sample values and then applying a linear filter whose coefficients are all zero except the central one which is equal to 1. For this reason, we can conclude that applying the statistical operators in the encrypted domain, raises the same difficulties raised by the operators described in Section 3.2
3.7 Signal Compression

Signal compression is a fundamental operation in many applications. As a matter of fact some kinds of signals like videos (and to a lesser extent audio) are only stored and exchanged in compressed format. Generally speaking any signal compression system follows the general scheme reported in Figure 2.

Decorrelation

The first operation performed by virtually all the compression algorithms is signal decorrelation. As a matter of fact, all the signals of practical interest, be them audio signals, images or videos, exhibit a very strong correlation between adjacent samples. From an information theoretic perspective this means that signals are information sources with memory. Henceforth, we know from rate distortion theory that they should be compressed by adopting a vector quantizer (possibly followed by an entropy source encoder). However vector quantizers for sources whose memory lasts for many samples are very difficult to design and very complex to apply, hence it is customary to first decorrelate the signal, transforming it in an approximately memoryless source, and then to apply a much simpler scalar quantizer. The two most common approaches to signal decorrelation are signal prediction and signal transformation. In the first case, the value of each sample is predicted by relying on the previous samples, then the error between the true signal and the prediction is computed. If the prediction is a good one the samples of the prediction error will be uncorrelated and hence they can be quantized scalarly. In the simplest case a linear predictor is adopted, yielding

$$f(n) = \sum_{k=1}^{L} a(k) f(n - k),$$

where the prediction is performed by relying on the reconstructed signal values $f(n - k)$ given that these are the only quantities available at the decoder. According to the above equation prediction is nothing but a causal linear filter. In some cases, though, the coefficients of the filter $a(k)$ are not known in advance and hence must be computed during the compression itself and stored together with the signal.

The second approach to signal decorrelation is to apply a linear transform with good decorrelation properties. The optimal transformation in this sense leads to the Karhunen-Loeve Transform (KLT), whose form, however, depends on the to-be-compressed signal and hence can not be used, since its coefficients would have to be stored together with the compressed signal, hence compromising the compression rate. For a wide class of signals, including audio signals and images, the DCT transform (or the DFT transform) provides a good approximation of the KLT and hence is

\[\text{Figure 2: General form of a compression algorithm.}\]

\[\text{In some cases, one of the blocks appearing in the figure may be missing or substituted by an alternative operation, however the great majority of the compression schemes operate according to this scheme.}\]
used very frequently in many compression schemes. The most popular example in this sense is surely the JPEG compression standard where pixel decorrelation is obtained by applying an $8 \times 8$ DCT to $8 \times 8$ image blocks.

**Quantization** The quantization step, be it applied directly to signal samples, to the prediction error or in a transformed domain, is the step where some information is discarded thus making compression possible. As such it is a non-invertible step, whose effects must be carefully analyzed to ensure that the quality of the compressed signal is still acceptable. As discussed in Section 3.2, quantization is a non-linear transformation, hence its implementation in the encrypted domain is likely to pose some problems.

Another key point of the quantization step is that quantization is usually performed by taking into account the impact that the loss of information involved by it has on the perceived quality of the compressed (and reconstructed) signal. For instance, in the JPEG compression standard adopts a different quantization step for different frequency coefficients due to the different sensibility of the human eye to noise added in different frequency bands. In some cases, doing this kind of consideration in the encrypted domain may raise some difficulties.

**Entropy coding** The actual compression of the signal, i.e. its representation by a string of symbols (usually bits) as short as possible is the last operation to be performed and it usually consists of a scalar entropy coder applied at the output of the quantization step. In some cases, the scalar encoder is replaced by a vector encoder or by more sophisticated operations, like in the JPEG coding standard where the quantized DCT coefficients are first run-length encoded prior to entropy coding (performed by using a classical Huffman coding algorithm). In principle both lossless and lossy entropy coding can be used, however in most cases a lossless approach is used by leaving the lossy part to the quantization step.

As a matter of fact, the entropy coding step is not a pure signal processing operation, since it takes as input a signal and produces at the output a string of symbols, nevertheless it is customarily to treat it as signal processing operation. Among the most commonly used entropy coding algorithms we mention\(^{34}\): Huffman coding, arithmetic coding, context-based coding, Lempel-Ziv coding.

**Coding encrypted signals** Compressing an encrypted signal poses several problems, both from a practical and a theoretical point of view. Practical problems regard the difficulty of performing the first two steps depicted in Figure 2 in the encrypted domain. Whereas some possibilities exist for sample decorrelation, given that it may be performed by means of linear operators, whose application to encrypted signals is in principle feasible by relying on the homomorphic properties of some asymmetric cryptosystems, quantization is more cumbersome since it is based on thresholding that is a notoriously difficult operation to be carried in the encrypted domain (at least if the recourse to interactive protocols is not allowed).

\(^{34}\)A detailed description of even the most popular entropy coding algorithms is beyond the scope of this brief section, given that it would require the introduction of several information theoretic concepts. Interested readers may refer to any good textbook in Information Theory.
The problems with entropy coding are of a more theoretical nature. Given that the output of a good encryption scheme should resemble as much as possible a random sequence, we could argue that the lossless encoding of an encrypted signal is a desperate task. As a matter of fact, this may not be the case. By relying on Distributed Source Coding (DSC) principles, it may be possible to losslessly encode an encrypted signal at the same rate of a coder working on the plain data (this is possible, at least in principle, if the decoder knows the encryption key). The feasibility of DSC-based compression of encrypted data has been demonstrated in [61], where such principles are applied to the compression of binary strings encrypted by XOR-ing them with a secret binary key of the same size of the to-be-protected data. An extension to lossy compression of Gaussian sequences is also proposed. However the applicability of the approach described in [61] to realistic scenarios where more practical encryption systems are used (e.g. DES or AES), and to the lossy compression of multimedia data is still to be investigated.

3.8 High Level Signal Processing

The signal processing operations that we described so far are very simple ones. Usually they take a signal as an input and provide a signal at the output (with the only exception of signal compression where the output is a string of symbols). In many applications, however, more complex operations are needed, for instance, we may want to extract features from signals, or we may want to take decisions based on the signals we have. Examples of such, higher level, operations are:

- **Segmentation.** The signal is segmented into meaningful chunks having homogeneous properties. For instance an audio signal could be split into segments corresponding to speech and music, or an image may be segmented into regions representing different objects.

- **Discontinuity extraction.** In this case we want to identify the points in the signal where a given characteristic of the signal changes abruptly. In the case of images, this may correspond to edge extraction, whereas for video sequences it may be used for the detection of scene changes.

- **Classification.** Different parts of the signal (or even single samples - or pixels in the case of images) are assigned to one of a predefined set of classes. For instance, once a signal or an image has been segmented, we may want to label each segment as belonging to a class among a predefined set of classes. For music signals we may want to classify each segment according to the type of music being played, whereas for remote sensing images, each pixel (or region) may be assigned to a particular class of terrain, e.g. water, urban areas, agricultural land etc ... 

- **Decision making.** By relying on the observation of a signal a decision must be taken, e.g. a medical diagnosis.

---

35 According to certain authors the operations listed below can not even be categorized as truly signal processing operators, since they are often seen as pattern recognition operators.
The algorithms involved by all the above operations, require that many of the basic tools described previously are applied several times, according to a certain algorithm flow. Then, the application of these higher level tools on encrypted data does not only require that the basic signal processing described before are possible but that the right algorithm flow can be followed without understanding the data the algorithm works on. For instance, we may have to perform an if or a for command whose ruling conditions have to be checked without decrypting the data they are based on. It can be argued that this is a difficult goal that can be achieved only by resorting to some amount of interaction between the players involved in the computation.

4 Pattern recognition in the encrypted domain

As it is clear from the previous sections, homomorphic encryption allows some very simple signal processing operations to be carried out in the encrypted domain, thus effectively protecting the signals involved in the computation. Specifically, the availability of an additively homomorphic cryptosystem permits the application of linear operators, like linear transforms or linear filters, to encrypted signals. Unfortunately linear operators alone are not enough for many practical applications, where highly non-linear tasks, like taking decisions or selecting a proper entry in a database, must be performed. This is the case with the most common pattern recognition tasks, including classification, matching, database search . . . . It is the goal of this brief section to analyze the most common pattern recognition tools involved in biometrics applications, to discuss the security requirements associated to them and identify a restricted number of primitives that are needed to apply these tools to encrypted data and signals.

4.1 Two very common tasks

The number of tools and tasks usually encompassed under the Pattern Recognition umbrella is virtually endless. An exhaustive discussion of how such tools could be applied in the encrypted domain is then impractical. On the other side, by restricting the analysis to biometrics applications, we can easily identify two tasks that are common to virtually all applications and whose solution would permit to solve the security problems of a great number of scenarios of practical interest. These two central tasks can be broadly identified as: i) Pattern Matching and ii) Classification. Pattern matching can be further divided into two cases: i) Verification, as a one to one matching problem, and ii) Identification, as a typical one to many matching problem.

More specifically, the verification problem can be defined as follows: given two patterns \(V_1\) and \(V_2\) decide whether they represent the same object or not. On the other side, the identification problem answers the following question: given a pattern \(V\) and a set of patterns \(V = \{V_1, V_2 \ldots V_n\}\), is there a pattern in \(V\) that corresponds to \(V\)? If yes, which is the index of such a pattern?

As to classification, we are given a pattern \(V\) and a number of classes \(C_1, C_2 \ldots C_p\) defined in a proper mathematical way, and we must decide the class \(V\) belongs to.

Note that in this description we purposely used rather vague terms, avoiding to give a precise definition of the involved entities like patterns, objects and classes. We will
be more precise later on.

A two-step process. From a very general point of view, both pattern matching and classification can be seen as two-step processes. The first step is the so called feature extraction, in which the pattern to be classified is transformed into a \( p \) long vector whose components, called features, describe some particular characteristics of the to-be-classified pattern. As an example, we may consider the classification of image regions. The region to be classified is the pattern, while the feature vector may contain the average grey level and the standard deviation of the pixels belonging to the region, the area of the region, its inertia moments etc.... Feature extraction is a crucial and necessary step since on one side it permits to simplify the pattern description, by reducing it to a vector in \( \mathbb{R}^p \), and on the other side the extracted features are supposed to describe some meaningful characteristics of the pattern to be classified.

The second step is the actual matching or classification step. The feature extraction step is highly application dependent and no general theory exists for it. For this reason, it is not possible to define the set of primitives that need to be developed to extract the features in the encrypted domain. Moreover, the involved operations are usually highly non-linear and can not be carried out by resorting to homomorphic encryption only. On the contrary, very complicated MPC protocols would be needed thus making a s.p.e.d.implementation\(^{36}\) of the feature extraction block very difficult. Two examples where the features could be extracted in the encrypted domain are when they correspond to a set of frequency coefficients (in particular their energy, thus calling for non linear operations), and when they are calculated by projecting a longer vector (e.g. signal samples or image pixels) on known directions (see for instance face classification by means of eigenfaces). In the majority of the cases we can assume that features extraction is carried out in the plain domain\(^{37}\), this is the case for instance of shape features, features relying on AR model fitting, identification of minutiae in fingerprint processing ....

4.2 Pattern matching: the verification problem

As opposed to feature extraction, processing the feature vectors for either pattern matching or classification is a rather standard (though not easy) task always following a few number of fixed steps. It is then extremely important that these steps are defined and their security constraints identified, since doing so will allow to build a rather general theory encompassing the application of some the most common pattern recognition tasks on encrypted data. In this section we start by considering the easiest among our problems, namely the verification problem.

A general verification problems can be summarized as follows:

- One party, say \( P_1 \) knows a feature vector \( V_1 \).
- Another party, say \( P_2 \) knows another feature vector \( V_2 \).

\(^{36}\)By s.p.e.d.(standing for Signal Processing in the Encrypted Domain) implementation we mean an implementation that is carried out in the encrypted domain.

\(^{37}\)This necessary assumption has some consequences, though, since it make impossible, for instance to hide the features a particular pattern recognition algorithm is based on.
We want to answer the question: is $V_1$ close enough to $V_2$?

As it can be seen, the verification problem boils down to only two operations: i) distance calculation and ii) comparison against a threshold. As soon as an efficient protocol is made available to perform these two tasks, a secure protocol for pattern verification can be built. In order to do so, it is necessary that the security requirements are defined. Though many different scenarios are possible, in most of the cases the following requirements must be verified. At the end of the protocol:

- $P_1$ gets yes/no.
- $P_2$ gets nothing or yes/no.
- $V_1$ and $V_2$ are kept secret.
- The distance function and the threshold may be assumed to be public parameters.

In a first variant it may be required that the threshold is known to one party only. In a second variant the actual distance between $V_1$ and $V_2$ may be revealed to one or both parties.

### 4.3 Pattern matching: the identification problem

While the verification problem involves a one to one matching, the identification problem corresponds to a one to many match. Specifically, pattern identification can be summarized as follows:

- One party, say $P_1$ knows a feature vector $V_{\text{test}}$.
- Another party, say $P_2$ knows a set of feature vectors $\mathcal{V} = \{V_1, V_2 \ldots V_n\}$.
- The possible questions to be answered are:
  1. Is $V_{\text{test}}$ close to at least one $V_i \in \mathcal{V}$? Boiling down to: is the minimum distance between $V_{\text{test}}$ and the elements in $\mathcal{V}$ smaller than a threshold?
  2. Which is the index of the feature vector in $\mathcal{V}$ closest to $V_{\text{test}}$?
  3. How many elements in $\mathcal{V}$ are close enough to $V_{\text{test}}$?

As it can be easily seen, in most of the cases identification boils down to calculation of several distances, thresholding and/or computation of a minimum. Hence there are two main differences with respect to verification. The first one is quantitative, in that several distances and thresholds must be computed instead of one. The second is qualitative, since a new operator, namely minimum computation is now needed.

With regard to the security requirements, the situation is slightly more complicated than in the verification case, however it is still possible to consider a standard set of requirement as follows. At the end of the protocol:

- $P_1$ gets: i) yes/no or ii) the index of the minimum distance feature vector
- $P_2$ gets nothing
• \( V_{test} \) and \( V \) are kept secret.
• The distance function and the threshold may be assumed to be public parameters.

A common alternative in which the verification is used to decide whether \( P_1 \) belongs to a set of users allowed to access a given system is the following:

• \( P_1 \) gets nothing
• \( P_2 \) gets yes/no
• \( V_{test} \) and \( V \) are kept secret.
• The distance function and the threshold may be assumed to be public parameters.

4.4 The classification problem

We now turn the attention to the last (and more complex) pattern recognition problem addressed in this report, i.e. classification. The definition of the classification problem is rather simple:

• One party, say \( P_1 \) owns a feature vector \( V \).
• One party, say \( P_2 \) has a model of several classes \( C_1, C_2 \ldots C_p \).
• Which class does \( V \) belong to?

Note that we are implicitly assuming that \( V \) always belongs to one of the classes considered by \( P_2 \). If this is not the case we could always add a fake class including all the feature vectors that do not belong to the original classes.

Despite its simplicity, a general identification of the tools involved in a classification problem is not an easy task. The main difficulty relies in the way the classes are modeled. This is a crucial point that determines the kind of processing needed for the classification. For instance, if a prototype is given for each class, classification of \( V \) may coincide with an identification problem. On the other hand, if the classes are implicitly described by means of a neural network, classification is equivalent to running the neural network. Among the possible ways to represent the classes of a classification problem we have: i) description by prototypes, ii) description by examples, iii) neural networks, iv) support vector machines, v) branching programs, vi) classification trees and many others.

By abstracting from the model used to describe the classes, we may try to fix the security requirements for the classification problem. There are two possible scenarios. According to the first one a party wants to classify a pattern, he knows how to do it, but he lacks the computational power to do it himself. In this case the security requirements may be set as follows:

• \( P_1 \) owns \( V \).
• The classes \( C \) are publicly available.
At the end of the protocol, $P_1$ gets the class $V$ belongs to, $P_2$ gets nothing.

There is at least a second fundamental scenario where $P_1$ may ask $P_2$ to classify his data. In this scenario, a remote classification service is provided by a non-trusted party ($P_2$). In order to preserve the data owned by $P_1$, $P_2$ carries out its task without getting any knowledge about the data provided by $P_1$. At the same time, the service provider may no be willing to disclose the algorithms it is using for the classification, since they represent the basis for the service it is providing. Keeping the algorithm secret may be interpreted in two different ways. According to the former the mathematical model used to describe the classes is known but the model parameters are not, while according to the latter the mathematical model is also to be kept secret. The second case usually requires that the features used to represent the to-be-classified pattern are computed in the encrypted domain by the server, a situation that is rather difficult to handle, hence in the following we will refer only to the first scenario. In this case, the security requirements can be set as follows:

- $P_1$ wants to keep the vector $V$ secret.
- $P_2$ wants to keep the parameters of the models defining the various classes secret.
- At the end of the protocol, $P_1$ gets the class $V$ belongs to, $P_2$ gets nothing.

The crucial point in the classification problem is the choice of the most suitable model to describe the classes, since the primitives needed for a s.p.e.d.implementation of a classifier strongly depend on such a model. While several solutions are appearing in the literature addressing classification through neural networks, branching programs or linear classifiers, a rigorous comparison of the drawbacks and advantages offered by the various implementations from a s.p.e.d.point of view is still missing.

## 5 Conclusion

In this deliverable we discussed some of the main cryptographic and signal processing tools that are required to perform signal processing in the encrypted domain. As an application scenario we considered the problem of pattern recognition over encrypted data and we identified a restricted number of primitives that are needed to apply such techniques to encrypted data and signals.

## References


