R. (Inald) L. Lagendijk, Zekeriya Erkin, and Mauro Barni

Encrypted Signal Processing for Privacy Protection

Conveying the utility of homomorphic encryption and multiparty computation

In recent years, signal processing applications that deal with user-related data have aroused privacy concerns. For instance, face recognition and personalized recommendations rely on privacy-sensitive information that can be abused if the signal processing is executed on remote servers or in the cloud. In this tutorial article, we introduce the fusion of signal processing and cryptography as an emerging paradigm to protect the privacy of users. While service providers cannot access directly the content of the encrypted signals, the data can still be processed in encrypted form to perform the required signal processing task. The solutions for processing encrypted data are designed using cryptographic primitives like homomorphic cryptosystems and secure multiparty computation (MPC).

NEED FOR PRIVACY PROTECTION IN SIGNAL PROCESSING

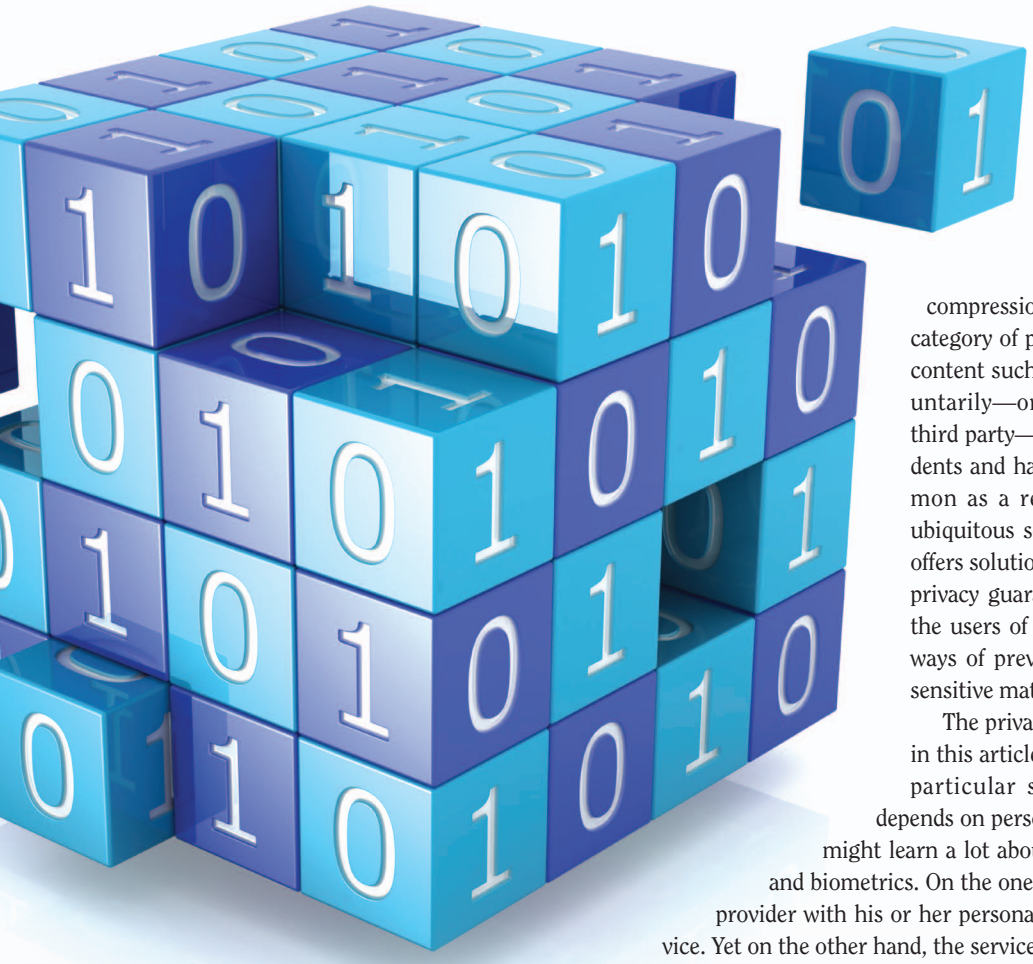
Research by the signal processing community has given birth to a rich variety of signal recording, storage, processing, analysis, retrieval, and display techniques. Signal processing applications are found in many fields of economic and societal relevance, including medical diagnosis, multimedia information services, public safety, and the entertainment industry. The design of a particular signal processing solution is commonly driven by objective or perceptual quality requirements on the processing result and by the tolerable computational complexity of the solution. In the past decade, however, rapid technological

Digital Object Identifier 10.1109/MSP.2012.2219653
Date of publication: 5 December 2012

1053-5888/13/$31.00©2013IEEE

developments in areas such as social networking, online applications, cloud computing, and distributed processing in general have raised important concerns regarding the security (and in particular, the privacy) of user-related content. We believe that the time has come to bring privacy-sensitive design to the signal processing community.

Privacy concerns about personal information have always existed. For instance, privacy-sensitive information might be exchanged during a video conferencing session. In such cases, the classic model of security is applicable, specifically two parties that trust each other communicate while protecting their communication from third, possibly malicious, parties. It is generally sufficient that some cryptographic primitives are applied on top of transmission, compression, and processing modules. Another category of privacy concerns exists when personal content such as images and videos are shared voluntarily—or unknowingly made available by a third party—on social media. Privacy-related incidents and harms are becoming increasingly common as a result of the growing popularity of ubiquitous social media. Here, technology hardly offers solutions; proper legislative underpinning of privacy guarantees and simultaneously educating the users of social media seem far more effective ways of preventing the abuse of shared privacy-sensitive material.



The privacy threats that we are concerned with in this article are associated with the provider of a particular service. In offering a service that depends on personal information, the service provider might learn a lot about a user's preferences, past behavior, and biometrics. On the one hand, the user must trust the service provider with his or her personal information to make use of the service. Yet on the other hand, the service provider is not a trusted party per se. In this article, when we say service provider, we generally refer to parties that operate on some user-related content. We give three signal processing examples to illustrate service provider-related privacy issues.

- Biometric techniques, such as face recognition, are increasingly deployed as a means to unobtrusively verify the identity of a person. Digitized photos allow the automatizing of identity checks at border crossings using face recognition [1]. Surveillance cameras in public places have led to interest in the use of face recognition technologies to automatically match the faces of people shown on surveillance images against a database of known suspects [2], [3]. The widespread use of biometrics raises important privacy concerns if the face recognition process is performed at a central or untrusted server. People might be tracked against their will, and submitting query faces to a database of suspects might unjustly implicate innocent people in criminal behavior.
- People use social networks to get in touch with other people, and they create and share content that includes personal information, images, and videos. A common service provided in social networks is the generation of recommendations for finding new friends, groups, and events using collaborative filtering techniques [4]. The data required for the collaborative filtering algorithm is collected from sources such as the user's profile, friendships, click logs, and other actions. The service providers often have the additional right to distribute processed data to third parties for completely unrelated commercial or other usage [5].

- Many homes have a set-top box with high storage capacity and processing power. The television service providers use smart applications to monitor the viewers' actions to gather statistical information on their viewing habits and their likes and dislikes. Based on the information collected, the service provider recommends personalized digital content like television programs, films and products. At the same time, an individual's way of living can be inferred from the information collected by the service provider.

While users in the above examples clearly experience the advantages of providing personal information, they also put their privacy at risk as the service provider might or might not preserve the confidentiality of personal information, or the information might be leaked or stolen.

The need for privacy protection has triggered research on the use of cryptographic techniques within signal processing algorithms. Privacy-sensitive information is encrypted before it is made available to the service provider—an action that might seem to impede further processing operations like the ones described above. However, this is not the case. The use of modern cryptographic techniques makes it possible to process encrypted signals, and users and service providers taking part in the processing have the opportunity to keep secret the information, or part thereof, that they provide for the computation. The application of cryptographic techniques to enable privacy-protected signal processing is of invaluable importance in many situations.

The aim of this tutorial article is to expose signal processing theorists and practitioners to privacy threats due to service providers executing signal processing algorithms. Certain cryptographic primitives that can be elegantly used to protect the privacy of the users are described. The article does not assume particular cryptographic background knowledge, and we explicitly take a signal processing perspective on the privacy problem. Alternative signal processing-based approaches to privacy protection exist that use, for instance, blinding and randomization techniques [6]–[8]. We focus specifically the use of encryption techniques because this approach is far less well known to the signal processing community.

In "Private Key and Public Key Cryptography," "Additively Homomorphic Public Key Encryption," "Arithmetic Comparison Protocol," and "MPC Using Garbled Circuits," a concise, high-level introduction is provided and background knowledge of cryptographic primitives used in the article. To match the tutorial level to the signal processing community, some of the cryptographic protocols' descriptions have been simplified to the point that they might look awkward to cryptographic experts. Our intention is not to give a comprehensive overview of all cryptographic techniques that are useful in signal processing, but to convey the basic ideas, utility, challenges, and limitations of cryptography applied to signal processing.

This article explains the privacy-protected counterparts of three well-known signal processing problems: face recognition, user clustering, and content recommendation. These three algorithms have been deliberately chosen to build up the complexity of the resulting encrypted signal processing. The algorithms have also been selected such that key concepts in signal processing are well covered, specifically, linear operations, inner products, distance calculation, dimension reduction, and thresholding.

A SIMPLE PRIVACY-PROTECTED SIGNAL PROCESSING ALGORITHM

ALGORITHM

Let us start by introducing a simple yet representative signal processing algorithm to expound how cryptographic techniques are used to achieve privacy protection. The algorithm we use exemplifies two operations commonly encountered in signal processing, specifically 1) the weighted linear combination of input samples, as in linear filtering and signal transformations, and 2) the comparison of some processed result to a threshold, an operation reminiscent of signal quantization and classification. Two parties are involved in the signal processing operation: the party denoted by *A(lice)*, which owns the privacy-sensitive signal $x(i)$, say some recorded biometrics or medical signals; and the party *B(ob)*, which has the signal processing algorithm $f(.)$, say some access control algorithm or diagnosis algorithm.

Alice is interested in the result $f(x(i))$, but does not wish to reveal $x(i)$ to Bob because of privacy concerns. Bob, on the other hand, cannot or does not wish to reveal (essential parameters of) his algorithm $f(.)$ to Alice for computational or commercial reasons. An example would be the intricate details of a commercially successful service such as search engines. It is roughly known which data is used in producing search results, but the exact function involved is not publicly known. This setup is typical of the examples mentioned in the section "Need for Privacy Protection in Signal Processing," and it will also play an important role when we discuss more elaborate privacy-protected signal processing operations in later sections.

The toy example that we will use to convey the main ideas in privacy-protected signal processing is the following. Bob owns an algorithm that processes two signal samples $x(1)$ and $x(2)$ to obtain a binary classification result C (see also Figure 1):

$$C = \begin{cases} 0 & \text{if} \quad h_1 x(1) + h_2 x(2) < T, \\ 1 & \text{otherwise.} \end{cases} \tag{1}$$

SECURITY MODEL

In this simple example, the value of the signal samples $x(1)$ and $x(2)$ are private to Alice and hence held secret from Bob. The linear weights h_1, h_2, and threshold T are private to Bob. The classification result $C \in \{0,1\}$ should be private to Alice. In other words, Bob must be unaware not only of the input signal $x(i)$, but also of the output result C.

At this point, we need to make two security model assumptions. The first is that Bob plays his role correctly and always executes $f(x(i))$ in a correct manner, without attempting to disrupt C. Bob's possible attacks concentrate on obtaining the input signal $x(i)$ or the intermediate results, such as the value of $h_1 x(1) + h_2 x(2)$, or C. Under this assumption, Bob is called an honest-but-curious or a semihonest party [9]. The second

assumption is that Alice is not able to submit unlimited processing requests to Bob, because this would enable her to learn the values of h_1, h_2, and T by trial and error, which is commonly known as a sensitivity attack [10]. Sensitivity attacks are usually not treated in cryptography, and they are considered to be outside of the attacker model.

LINEAR COMBINATION OF ENCRYPTED SAMPLES

We now consider (1) in the situation that Alice sends Bob her input signal $x(i)$ in encrypted form. Alice encrypts $x(i)$ sample by sample, using a public key cryptosystem (see "Private Key and Public Key Cryptography") with the additively homomorphic property [11], [12]. The key properties of additive homomorphic encryption are

$$\begin{aligned} \mathcal{D}_{SK}(\mathcal{E}_{PK}(m_1)\cdot\mathcal{E}_{PK}(m_2)) &= m_1+m_2,\\ \mathcal{D}_{SK}(\mathcal{E}_{PK}(m)^w) &= w\cdot m. \end{aligned} \tag{2}$$

We refer to "Additively Homomorphic Public Key Encryption" for more background information. Alice sends only her public key PK and ciphertext to Bob. Figure 2 illustrates the subsequent processing steps. For the purpose of simplicity, we use the following shorthand ciphertext notation for encrypted signal samples using public encryption key PK:

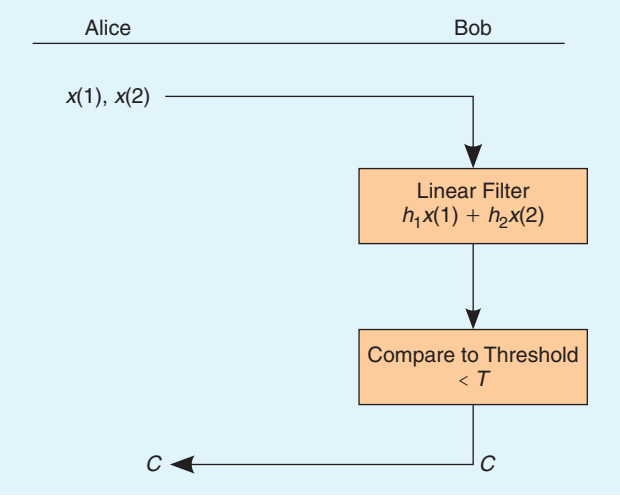
$$x(i) \overset{\text{encrypt with key } PK}{\rightarrow} \mathcal{E}_{PK}(x(i)) = [\![x(i)]\!]. \tag{3}$$

Thanks to the additive homomorphic property of the cryptographic system used, the linear part of the signal processing algorithm (1) can be rewritten to directly work on the encrypted signal values $[\![x(1)]\!]$ and $[\![x(2)]\!]$. Specifically, the counterpart of $h_1x(1)+h_2x(2)$ in the encrypted domain is (see "Additively Homomorphic Public Key Encryption")

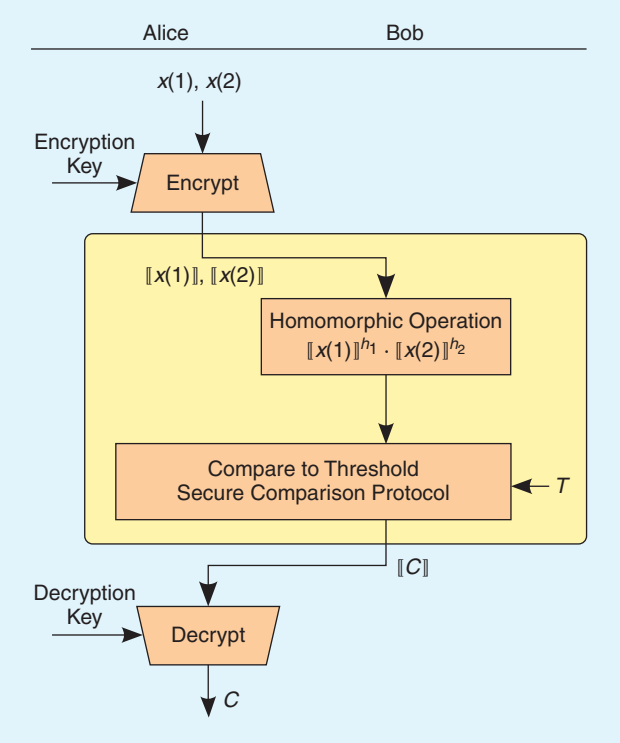
$$\begin{aligned} \mathcal{E}_{PK}(h_1x(1)+h_2x(2)) &= [\![h_1x(1)+h_2x(2)]\!]\\ &= [\![h_1x(1)]\!]\cdot[\![h_2x(2)]\!] \mod n\\ &= [\![x(1)]\!]^{h_1}\cdot[\![x(2)]\!]^{h_2} \mod n. \end{aligned} \tag{4}$$

This result shows that Bob can directly compute the encrypted result of the linear combination $h_1x(1)+h_2x(2)$ from the ciphertext values $[\![x(1)]\!]$ and $[\![x(2)]\!]$ without having access to Alice's secret decryption key SK. Bob also does not need to involve Alice in computing $[\![h_1x(1)+h_2x(2)]\!]$; the computation protocol does not require interaction between the parties. The result that Bob obtains is still encrypted, and it can only be decrypted (if needed) by Alice using SK.

Even though the equivalence of the plaintext-based operation in (1) and the ciphertext-based operation (4) is elegant, (4) comes with an important inherent limitation. In (1), $h_1x(1)+h_2x(2)$ can, at least in principle, be evaluated on any real values $(x(1), x(2))$ and (h_1, h_2). The operation $[\![x(1)]\!]^{h_1}\cdot[\![x(2)]\!]^{h_2}$, however, assumes that (h_1, h_2) are arbitrary integers, and that $[\![x(1)]\!]$ and $[\![x(2)]\!]$ and all intermediate results are integers in the interval $[0, n-1]$. This is because public key cryptographic operations are carried out in finite fields, involving arithmetic operations on integers such as the multiplication (modulo n) in (4). A common workaround is to multiply real-valued numbers by a sufficiently large constant and quantize the scaled value. The scaling factor becomes part of the public key information as, for instance, the value of T must be scaled too. The loss of accuracy due to quantization is directly controlled by the scaling factor. Note that indiscriminate scaling is not possible due to the range limitations on the processed signal values. If negative numbers are required, these are typically mapped on the upper half of the interval $[0, n-1]$.



[FIG1] Block diagram of the toy signal processing algorithm in (1).



[FIG2] Privacy-protected version of the signal processing algorithm in (1). The yellow area indicates the operations that are carried out on encrypted (signal) values.

PRIVATE KEY AND PUBLIC KEY CRYPTOGRAPHY

The objective of encryption is to hide a number m, commonly called plaintext message, in a ciphertext c that is unintelligible to anyone not having access to the proper decryption key. In signal processing, the message m can be an audio sample $x(i)$, a pixel $x(i, j)$, or a feature derived from the signal such as the signal's mean value or a DFT coefficient.

When encrypting m, one can use private key or public key encryption, which are also known as symmetric and asymmetric key encryption, respectively. Private key and public key encryption differ in which keys are used by the encrypting party A (commonly called Alice in the field of cryptography) and the decrypting party B (Bob).

In private key encryption, Alice encrypts m with key K, yielding the ciphertext $c = \mathcal{E}_K(m)$. Bob, the recipient of the message, decrypts c using the same key K. In other words, if $\mathcal{D}_K(c)$ indicates the decryption of the ciphertext c using key K, then $m = \mathcal{D}_K(\mathcal{E}_K(m))$. Since the same key K is used for encryption and decryption, this key must be kept secret from everyone except Alice and Bob. The difficulty in private key encryption is the sharing of the key before encryption starts. Key distribution protocols exist that elegantly solve this difficulty by using public key encryption approaches [98]. Private key encryption algorithms are typically based on repeatedly applying rounds of highly efficient but complex operations on input bits or bytes. The ciphertext is secure because inverting these concatenated rounds of operations is prohibitively expensive. Examples of commercially used private key encryption are Data Encryption Standard (DES) [99] and Advanced Encryption Standard (AES) [100].

The distinguishing technique used in public key cryptography is the use of two different keys. The key used to encrypt a message is not the same as the key used to decrypt it. The encryption key PK is made publicly available not only to Alice but, in principle, globally distributed. Bob uses the private key SK to decrypt the ciphertext $\mathcal{E}_{PK}(m)$, yielding $m = \mathcal{D}_{SK}(\mathcal{E}_{PK}(m))$. The two keys are mathematically related, but it is computationally infeasible to compute the secret decryption key SK from the public encryption key PK. The security of public key encryption is based on the presumed hardness of mathematical problems like factoring the product of two large primes [101] or computing discrete logarithms in a finite field with a large number of elements [102]. The advantage of public key encryption is the simpler key management. Disadvantages are that the keys are much larger (more bits) and they yield substantially more computational overhead. This is due to the mathematical operations involved, such as exponentiation with large numbers.

As the multiplications and exponentiation on encrypted signal values $[\![x(1)]\!]$ and $[\![x(2)]\!]$ are carried out in modular arithmetic, the second and third lines of (4) require the operator "mod n" or "mod n^2," depending on the ciphertext space used by the cryptosystem. As is commonly done in cryptography, we will drop the modulo operator, whenever possible for the sake of notational simplicity. Nevertheless, it is important to be aware that computations on encrypted data are always performed in the algebra of the cipher text space.

COMPARISON TO THRESHOLD

The next step in (1) is that Bob compares the encrypted result $[\![h_1 x(1) + h_2 x(2)]\!]$ to the plaintext threshold T. Bob cannot calculate the result all by himself. In fact, solutions that would output the decision in clear would be insecure since Bob would then be able to efficiently decrypt every ciphertext by binary search. Rather, Bob has to obtain assistance from Alice: the comparison of an encrypted number and nonencrypted number requires an interactive protocol. Such a solution is called a secure two-party computation protocol, or just secure function evaluation. In "Arithmetic Comparison Protocol" and "MPC Using Garbled Circuits," we illustrate the essentials of two-party computation approaches to the comparison problem. The solution described in "Arithmetic Comparison Protocol" is representative of the class of arithmetic protocols, and exploits homomorphic encryption. "MPC Using Garbled Circuits" describes an example of the class of Boolean protocols which uses garbled circuits.

After the completion of the interactive protocol, Bob holds the encrypted result $[\![C]\!]$. Bob submits $[\![C]\!]$ to Alice, who obtains the result of the algorithm (1) after decryption with her secret key SK.

COMPLEXITY ANALYSIS

Processing signal samples that have been encrypted using a public key cryptosystem is computationally more demanding than the original plaintext version of the algorithm. The first reason for the increase in complexity is data expansion. Whereas signal samples $x(i)$ typically take 8–16 b, their encrypted counterparts are 1,024 or more bits long. For signal processing applications, such an enormous amount of data expansion is practically unacceptable, from both a storage and a communication point of view. Approaches have been developed for some signal processing problems that pack multiple signal samples into a single encrypted number [13]. In these cases the data expansion remains manageable, often at the cost of some computation overhead. Alternatively, the communication costs can be reduced by considering cryptosystems that have a smaller ciphertext space with the same security level, for instance the Okamoto–Uchiyama cryptosystem [14].

The second reason for the increased complexity is the nature of the operations involved. In particular, exponentiations such as $[\![x(i)]\!]^{h_i}$ in (4) and r^n in the Paillier cryptosystem (see "Additively Homomorphic Public Key Encryption"), are computationally most demanding. It is therefore necessary to seek an efficient cryptographic solution that causes as little increase in complexity as possible.

A common way to quantify and compare the complexity of the plaintext and ciphertext implementations is to count the number of multiplications, and (for the ciphertext version) the encryptions, decryptions, and exponentiations. The amount of data communicated between parties is also an indicator of the complexity of the ciphertext implementation. The complexity is usually expressed in terms of the order of magnitude of some algorithm

ADDITIVELY HOMOMORPHIC PUBLIC KEY ENCRYPTION

Central to most privacy-preserving signal processing algorithms is that certain public key cryptosystems are additively homomorphic. This means that there exists an operation on ciphertext $\mathcal{E}_{PK}(m_1)$ and $\mathcal{E}_{PK}(m_2)$ such that the result of that operation corresponds to a new ciphertext whose decryption yields the sum of the plaintext messages $m_1 + m_2$. In the case that the operation on the ciphertext is multiplication, we have

$$\mathcal{D}_{SK}(\mathcal{E}_{PK}(m_1) \cdot \mathcal{E}_{PK}(m_2)) = m_1 + m_2.$$

Note that m_1 and m_2 must both be encrypted with the same public key PK. As a consequence of additive homomorphism, any ciphertext $\mathcal{E}_{PK}(m)$ raised to the power of w results in an encryption of $w \cdot m$. This is easily seen from

$$\begin{aligned}\mathcal{D}_{SK}(\mathcal{E}_{PK}(m)^w) &= \mathcal{D}_{SK}(\underbrace{\mathcal{E}_{PK}(m) \cdot \mathcal{E}_{PK}(m) \ldots \mathcal{E}_{PK}(m)}_{w \text{ terms}}) \\ &= \underbrace{m + m + \cdots + m}_{w \text{ terms}} = w \cdot m.\end{aligned}$$

The subtraction of plain text messages can also be realized directly on the ciphertext, specifically

$$\mathcal{D}_{SK}(\mathcal{E}_{PK}(m_1) \cdot (\mathcal{E}_{PK}(m_2))^{-1}) = m_1 - m_2,$$

where a^{-1} denotes the multiplicative inverse of a. The Paillier public key cryptosystem [103] is an additively homomorphic cryptosystem that is quite popular in privacy-protected signal processing [11], [12]. The secret key consists of two large primes $SK = \{p, q\}$. "Large" in this context means that the primes contain 1,024 b or more. If $n = p \cdot q$, then the messages to be encrypted need to be in the range $[0, n-1]$, or in mathematical proper terms: $m \in \mathbb{Z}_n$. The Paillier encryption operation on a message m is then given by

$$\mathcal{E}_{PK}(m, r) = g^m \cdot r^n \mod n^2,$$

where $\mathcal{E}_{PK}(m, r) \in \mathbb{Z}^*_{n^2}$. The first thing to note is that, as in many cryptosystems, the operations are in modular arithmetic in the algebra of the ciphertext space. There is thus a limit, albeit an extremely large one, on the number of different messages that can be encrypted. The second thing to note is that the encryption equation takes two more parameters, specifically g and r. The number g is a generator of a subset (or formally, a subfield) of n values embedded in the range $[0, n^2 - 1]$. Together with the value of n, the public key of the Paillier cryptosystems is $PK = \{n, g\}$. The number r is randomly picked to ensure that when repeatedly encrypting the *same* message m, each ciphertext $\mathcal{E}_{PK}(m, r)$ is different. Interestingly enough, the random value r is *not* needed for decryption of ciphertext. We therefore often drop r from the notation, that is, $\mathcal{E}_{PK}(m, r) = \mathcal{E}_{PK}(m)$. Pallier decryption is somewhat more elaborate than encrypting; we refer readers to [103] for the details.

The homomorphic property of the Paillier cryptosystem can easily be verified. If we consider two Paillier encrypted messages $\mathcal{E}_{PK}(m_1, r_1)$ and $\mathcal{E}_{PK}(m_2, r_2)$ – note that we use two random values r_1 and r_2 – we find

$$\begin{aligned}\mathcal{E}_{PK}(m_1, r_1) \cdot \mathcal{E}_{PK}(m_2, r_2) &= (g^{m_1} \cdot r_1^n) \cdot (g^{m_2} \cdot r_2^n) \mod n^2 \\ &= g^{m_1 + m_2} \cdot (r_1 \cdot r_2)^n \mod n^2 \\ &= \mathcal{E}_{PK}(m_1 + m_2, r_1 \cdot r_2).\end{aligned}$$

Indeed, the product of $\mathcal{E}_{PK}(m_1, r_1)$ and $\mathcal{E}_{PK}(m_2, r_2)$ is an encryption of $m_1 + m_2$. Note that it is tempting to say that the product of the encryptions of m_1 and m_2 is equal to *the* encryption of $m_1 + m_2$. This is, however, generally incorrect as different randomly generated values r will be used for m_1, m_2, and $m_1 + m_2$. Other examples of public key encryption with homomorphic properties are Rivest, Shamir, and Adelman (RSA) [101], ElGamal [102], Damgård, Geisler, and Krøigård (DGK) [95], [94] Goldwasser–Micali [104], and Okamoto–Uchiyama [14] cryptosystems.

parameters. In Table 1, we show the result of counting these operations for the given example. Perhaps surprisingly, in Table 1 the complexity is caused not by the homomorphic operations in (4), but by the interactive protocol comparing $[\![h_1 x(1) + h_2 x(2)]\!]$ to T. The important parameter is β—the number of bits needed to represent $h_1 x(1) + h_2 x(2)$ (see "Arithmetic Comparison Protocol"). The main complexity is the $O(\beta^2)$ number of exponentiations in the arithmetic comparison protocol. In addition to the exponentiation, we should also realize that a multiplication of ciphertexts requires modular arithmetic, which by itself is also more expensive than the multiplication of plaintext values.

SECURITY MODELS

PRIVACY REQUIREMENTS

When we follow the steps in the above privacy-protected signal processing algorithm, it seems obvious enough that Bob does not learn anything about Alice's privacy-sensitive information. After all, the inputs and the intermediate and output results are encrypted. Alice does not learn anything about the processing algorithm except for the processing result C. Such informal inspection of the cryptographic operations is generally not sufficient to claim that the solution is indeed privacy-preserving. All cryptographic protocols, including those that concern privacy-preserving signal processing, must be accompanied by a formal security proof, or at least by a sketch reducing the proof at hand to a simpler, well-studied protocol. Another proof technique is to formulate how the protocol should work in an "ideal" world, and then show that the ideal and real world behave in a "similar" way.

[TABLE 1] COMPUTATION AND COMMUNICATION COMPLEXITY OF PLAINTEXT AND CIPHERTEXT VERSIONS OF (1).

	PLAINTEXT	CIPHERTEXT
MULTIPLICATION	$O(1)$	$O(\beta^2)$
ENCRYPTION	-	$O(\beta)$
DECRYPTION	-	$O(\beta)$
EXPONENTIATION	-	$O(\beta^2)$
COMMUNICATION	$O(1)$	$O(\beta)$

All proofs start by explicitly stating which information is to be kept secret by which party, and which capabilities and intentions potential adversaries have, be they participating parties or outsiders.

As an illustration, let us consider a security model assumption we made in the above signal processing algorithm. We said that "Bob cannot or does not wish to reveal his algorithm $f(.)$ to Alice." On closer inspection, we might wonder if we meant 1) that the parameters (h_1, h_2) and T are secret, or 2) that even the fact that Bob calculates a linear combination and compares a result to a threshold is secret, that is, the structure of the algorithm is secret. The first interpretation can be shown to be privacy-preserving, whereas the second interpretation is problematic. Bob leaks to Alice information about the algorithm, for instance, that the algorithm includes a comparison simply because Alice participates in the comparison protocol. Under this security model, the presented solution is formally not privacy-preserving from Bob's perspective.

Security proofs are important, but they are often lengthy and detailed at the same time. Conforming to the tutorial nature of this article, we will abstain from giving security proofs of the privacy-protected signal processing algorithms. Where relevant, we will refer to literature for further reading.

ATTACKER MODEL

Proofs of security critically rely on assumptions about the capabilities and intentions of adversaries. First, in public key cryptography it is always assumed that the attacker has restricted computational power. An attacker is therefore not able to break the hard mathematical problem on which the used cryptosystem relies (see "Private Key and Public Key Cryptography"). We also assume that, when needed, the keys are generated and certified by a third trusted party (a certification authority) prior to execution of the protocols, and the public keys are available to all users in the system.

A second assumption concerns the intentions of adversaries. An important assumption in the section "A Simple Privacy-Protected Signal Processing Algorithm" was that Bob is a curious-but-honest adversary participating in the computation. This attacker model describes Bob as a party that will follow all protocol steps correctly, but who is curious and collects all input, intermediate, and output data in an attempt to learn some information about Alice. A much more aggressive attacker model for Bob is the malicious adversary participating in the computation. In this case, Bob's intentions are to influence the computations such that Alice obtains a possibly incorrect answer. In the given example, Bob can easily influence the outcome by ignoring Alice's values $[\![x(1)]\!]$ and $[\![x(2)]\!]$, and using some fictive input values $\tilde{x}(1)$ and $\tilde{x}(2)$. Bob encrypts these values using Alice's public key PK, and the processing proceeds as explained earlier. In fact, if the communication between Alice and Bob is not secured using traditional (private key) cryptographic techniques, even a malicious outsider adversary, not participating in the computation, might influence the result by actively capturing Alice's encrypted input and replacing it with some encrypted bogus signal values. Even though outsider adversaries are important in real-world applications, the focus in encrypted signal processing is on the protection of privacy toward adversaries that participate in the computing process.

Achieving security against malicious adversaries is a hard problem that has not yet been studied widely in the context of privacy-protected signal processing. There are two likely reasons for this. First, it can be shown that any protocol that is secure against a curious-but-honest adversary can be transformed into one that is secure against malicious adversaries. The transformation requires proving the correctness of all intermediate computation steps using cryptographic techniques known as commitment schemes [15] and zero-knowledge proofs [16]. For instance, Alice could prove that she knows a certain value of an encrypted number without revealing that value itself. The drawback is that commitment schemes and zero-knowledge proofs are known to be notoriously computationally demanding and they increase significantly the number of interactive protocols between Alice and Bob. Loosely speaking, the protocol slowdown is in the order of a factor of ten [17], [18]. Second, the objective of privacy-protected signal processing is not to enforce correct computations on the service provider's side, as they already do that in nonprivacy-protected settings. Therefore, the malicious adversarial model might simply be an unrealistically aggressive scenario for many signal processing applications.

A third aspect of the attacker model describes whether and, if so, how parties involved in the computation might collude with each other. They could, for instance, exchange pieces of information that they collected to infer privacy-sensitive information. To illustrate collusion attacks, consider the slightly modified (and in the eyes of cryptographic experts, ridiculously simple) toy example where Alice has the private signal value $x(1)$ and another party, called Charles, has the private signal value $x(2)$. If Alice and Charles make use of the same public-private key pair, then collusion between Bob and Charles will leak $x(1)$ to Charles as he can decrypt the value $[\![x(1)]\!]$. A seemingly obvious solution to make such collusion impossible is to have Alice and Bob use different public-private key pairs. Unfortunately, we can then no longer exploit the additive homomorphic property and (4) no longer holds. We will see a similar issue arise in a more realistic situation in the section "Privacy-Protected K-Means Clustering."

The fourth and final aspect we address is the secrecy of the algorithm that Bob uses. As we mentioned in the section "A Simple Privacy-Protected Signal Processing Algorithm," Alice should be prohibited from repeatedly sending arbitrary input signals because she can infer critical parameters from the outputs of the algorithm using a sensitivity attack [10]. For instance, by sending the input $([\![x(1)]\!], [\![x(2)]\!]) = ([\![1]\!], [\![0]\!])$ Alice learns whether $h_1 \lesseqgtr T$. Furthermore, the secrecy of Bob's algorithm can be guaranteed only if certain algorithmic parameters cannot be inferred directly from the input-output relation. Let us take the example where Bob carries out a convolution with a filter whose impulse response he wishes to keep secret. If Bob provides the filter output directly to Alice, he completely reveals

his algorithm. After all, Alice simply sends Bob a signal with a delta impulse, and obtains the impulse response of the (supposedly secret) filter as output. Hence, although in some cases there might be a need for the secure evaluation of a secret algorithm, the algorithm's inherent properties might make secrecy a meaningless concept. Since sensitivity attacks are primarily related to algorithm properties, they are typically not considered part of the attacker model in cryptography.

We end this section by pointing out that privacy-protected solutions such as the one in the section "A Simple Privacy-Protected Signal Processing Algorithm" do not automatically render Alice anonymous. This is because Bob will be able to identify Alice not only on the basis of her IP address in the case of an online service, but (more relevant to this tutorial) also because of her unique public-private key pair. If a third user, say Charles, makes use of Bob's signal analysis service, then Charles will have a different public and private key. Users are unlikely to change their keys over time as this requires the re-encryption of all their data. Therefore, Bob will be able to identify Alice and Charles when they revisit Bob's service with another signal to process.

PROCESSING OF ENCRYPTED SIGNALS

We have familiarized ourselves with various aspects of processing encrypted signals through the toy example in (1). In this section, we provide more details on the applicability of the two cryptographic primitives used in the sections "A Simple Privacy-Protected Signal Processing Algorithm" and "Security Models," specifically homomorphic encryption and secure multiparty computation. Whereas "Additively Homomorphic Public Key Encryption," "Arithmetic Comparison Protocol," and "MPC Using Garbled Circuits" focus on the cryptographic aspects of homomorphic encryption and secure multiparty computation, this section takes the signal processing perspective to get a feeling for what could be "the right" cryptographic approach for a given privacy-sensitive signal processing problem. In later

ARITHMETIC COMPARISON PROTOCOL

We illustrate the principle of arithmetic two-party computation (secure function evaluation) by describing the main steps of a well-known protocol for comparing an encrypted number $[\![a]\!]$ and an unencrypted number b [95], [30]. Here, $[\![a]\!]$ has been encrypted by Alice using an additively homomorphic cryptosystem with public key PK. Note that in the toy example in the section "A Simple Privacy-Protected Signal Processing Algorithm," we have $[\![a]\!] = [\![h_1 x(1) + h_2 x(2)]\!]$ and $b = T$.

The idea of the comparison protocol is to consider the difference of a and b, and determine the sign, or the most significant bit, of this difference. Since a is only available as the encryption $[\![a]\!]$ it is impossible to get direct access to the sign bit. Instead, the sign bit is obtained by modulo reduction of encrypted differences. We next describe the protocol in more detail.

Initially, Bob has access to $[\![a]\!]$ and b, and he also knows that $0 \le a, b < 2^\beta$, where $\beta \in \mathbb{N}$. As a first protocol step, Bob computes the encrypted number $[\![z]\!]$ as

$$[\![z]\!] = [\![2^\beta + a - b]\!] = [\![2^\beta]\!] \cdot [\![a]\!] \cdot [\![b]\!]^{-1}.$$

Bob uses Alice's public key PK to compute the encryptions of 2^β and b. Note that we exploit the homomorphic property to add numbers under encryption. The value of z is a positive $(\beta+1)$-bit number. Moreover, z_β, the most significant bit of z, is exactly the comparison result we are looking for:

$$z_\beta = 0 \Leftrightarrow a < b.$$

If Bob had an encryption of $z \bmod 2^\beta$, the result would be immediate, because in the second protocol step z_β could be computed as

$$z_\beta = 2^{-\beta} \cdot (z - (z \bmod 2^\beta)).$$

The subtraction sets the least significant bits of z to zero. Bob can compute the difference of z and $z \bmod 2^\beta$ on encrypted values thanks to the homomorphic property. The multiplication by $2^{-\beta}$ effectively divides $z - (z \bmod 2^\beta)$ by 2^β, in this way shifting down the interesting bit. Multiplying by the plaintext constant $2^{-\beta}$ – that is the multiplicative inverse of 2^β – can be done again thanks to the homomorphic property.

Unfortunately, the value z is available to Bob only in encrypted form, so the modulo 2^β reduction cannot easily be performed. The solution is to engage with Alice in the third and interactive protocol step. Essentially Bob will ask Alice to first decrypt $[\![z]\!]$, then compute $z \bmod 2^\beta$, and finally re-encrypt the result before sending it to Bob. Obviously, such approach would reveal z to Alice, which leaks information about a and b. Therefore, Bob first blinds the value of z by adding a randomly generated value r only known to Bob:

$$[\![d]\!] = [\![z + r]\!] = [\![z]\!] \cdot [\![r]\!].$$

If r has the right random properties, Bob can safely send $[\![d]\!]$ to Alice, who will learn no useful information after decryption. Alice then computes $d \bmod 2^\beta$, and sends Bob the encrypted result. Bob finally removes the initially added random value r as follows

$$\begin{aligned}[\![z \bmod 2^\beta]\!] &= [\![(d \bmod 2^\beta) - (r \bmod 2^\beta) + \lambda\, 2^\beta]\!] \\ &= [\![d \bmod 2^\beta]\!] \cdot [\![r \bmod 2^\beta]\!]^{-1} \cdot [\![\lambda]\!]^{2^\beta}.\end{aligned}$$

Here, $\lambda\, 2^\beta$ is a correction term with $\lambda \in \{0, 1\}$ indicating whether $(d \bmod 2^\beta)$ is larger or smaller than $(r \bmod 2^\beta)$. The encrypted value $[\![\lambda]\!]$ is obtained using a subprotocol, known as Yao's millionaire problem [24], which compares Alice's plaintext value $(d \bmod 2^\beta)$ to Bob's plaintext value $(r \bmod 2^\beta)$. We refer to [105]–[108] for details on this subprotocol, which operates on the β individual bits of the values to be compared. That is why the parameter β shows up in various computation complexity tables in this article.

sections, we elaborate on concrete applications of these cryptographic primitives in recent publications.

USING HOMOMORPHIC CRYPTOSYSTEMS

At the heart of many signal processing operations, such as linear filters, correlation evaluations, and signal transformations, is the calculation of the inner product of two discrete-time signals or arrays of values $x(i)$ and $y(i)$. If both signals contain M samples, then their inner product $\mathcal{I}$ is defined as

$$\mathcal{I} = \langle x(.), y(.) \rangle = [x(1), x(2), \ldots, x(M)] \cdot \begin{bmatrix} y(1) \\ y(2) \\ \vdots \\ y(M) \end{bmatrix} = \sum_{i=1}^{M} x(i) y(i). \quad (5)$$

We can directly carry out this calculation on encrypted signals provided that in one signal, say $x(i)$, the samples are individually encrypted, and the other signal, say $y(i)$, is in plaintext. The encryption system used must also have the additive homomorphic property (see "Additively Homomorphic Public Key Encryption"). Using the notation in (3) and applying the additive homomorphic property of, for instance, the Paillier public key cryptosystem, we can rewrite (5) in a form that directly operates on the encrypted signal samples $[\![x(i)]\!]$

$$\mathcal{E}_{PK}(\mathcal{I}) = \mathcal{E}_{PK}\left(\sum_{i=1}^{M} x(i) y(i)\right) = \prod_{i=1}^{M} \mathcal{E}_{PK}(x(i) y(i)) = \prod_{i=1}^{M} \mathcal{E}_{PK}(x(i))^{y(i)} = \prod_{i=1}^{M} [\![x(i)]\!]^{y(i)}. \quad (6)$$

This expression is a generalization of (4) for M, rather than just two, samples. Equation (6) is an important result, as it allows us to implement efficiently linear operations on entire encrypted signals, without having to resort to interactive protocols between parties. As we will see in later sections, although interactive protocols are unavoidable in privacy-protected signal processing, the more of the processing that can be done by exploiting homomorphic properties, the more efficient the ciphertext version of the algorithm will be.

The result $\mathcal{E}_{PK}(\mathcal{I})$ is encrypted and can be decrypted only by the party that has access to the secret key SK. Note that the homomorphic addition is applied in a modular fashion, which allows for a finite number of different amplitudes of the samples $x(i)$. Especially for larger values of M, it is therefore essential to choose a plaintext space that is large enough so that overflows due to modular arithmetic are avoided when operations are performed on encrypted data.

One particular example of (5) is to make $y(i)$ equal to the samples of the basic functions of the discrete Fourier transform (DFT). Equation (6) then implements the DFT of an encrypted signal, yielding encrypted DFT coefficients. The computational complexity and memory requirements are studied in [19] and [20] as a function of M and mod n for such encrypted DFT and the commonly used fast implementation FFT.

One might wonder whether a similar reformulation of (5) exists in the case that both $x(i)$ and $y(i)$ have been encrypted. Following (6), this boils down to the question whether the following identity holds:

$$\mathcal{E}_{PK}(x(i) \cdot y(i)) = \mathcal{E}_{PK}(x(i)) \odot \mathcal{E}_{PK}(y(i)). \quad (7)$$

Here, $\odot$ represents a "multiplication-like" operation that produces $[\![x(i) \cdot y(i)]\!]$ as a result. Note that $\odot$ is not the usual modular multiplication of two encrypted numbers, as this is equivalent to $[\![x(i) + y(i)]\!]$ in an additively homomorphic cryptosystem.

For (7) to be true, the cryptosystem must also possess the multiplicative homomorphic property. A cryptosystem that possesses both the additive and multiplicative homomorphic property is called a fully (or algebraically) homomorphic cryptosystem. The existence of a secure fully homomorphic cryptosystem has long been studied by cryptographers. The seminal paper by Gentry [21] constructs a particular encryption scheme $\mathcal{E}_{PK}(x)$ that has the algebraic homomorphism property. From a theoretical perspective, this solves any secure computation problem. Alice just publishes her encrypted private data, and Bob can compute (at least in theory) an arbitrary (computable) function. Only Alice can recover the result of the computation using her secret key. Despite some recent advances [22], [23] to date, fully homomorphic encryption schemes are mainly of theoretical interest and far too inefficient to be used in practice. Thus, a secure two-party multiplication protocol is required for multiplying two encrypted samples. We will describe the secure multiplication protocol in the section "Using Blinding."

Besides linear operations, a common operation in signal estimation, compression and filtering is to calculate the squared error distance between two signals. If the signals $x(i)$ and $y(i)$ contain M samples, then their squared error distance $\mathcal{D}$ is defined as

$$\mathcal{D} = \| x(.) - y(.) \|^2 = \sum_{i=1}^{M} (x(i) - y(i))^2 = \sum_{i=1}^{M} x(i)^2 - 2\sum_{i=1}^{M} x(i) y(i) + \sum_{i=1}^{M} y(i)^2. \quad (8)$$

Let us consider again the case that $x(i)$ is only available as ciphertext $[\![x(i)]\!]$. We can then compute the encrypted value of $\mathcal{D}$ as follows:

$$\mathcal{E}_{PK}(\mathcal{D}) = \mathcal{E}_{PK}\left(\sum_{i=1}^{M} (x(i) - y(i))^2\right) = \left[\!\!\left[\sum_{i=1}^{M} x(i)^2\right]\!\!\right] \cdot \left[\!\!\left[\sum_{i=1}^{M} x(i)(-2y(i))\right]\!\!\right] \cdot \left[\!\!\left[\sum_{i=1}^{M} y(i)^2\right]\!\!\right] = \prod_{i=1}^{M} [\![x(i)^2]\!] \cdot \prod_{i=1}^{M} [\![x(i)]\!]^{-2y(i)} \cdot \prod_{i=1}^{M} [\![y(i)^2]\!]. \quad (9)$$

The terms in this expression deserve further investigation. The last term requires the encryption of $y(i)^2$, which is easy to compute

using the public key PK. The second term is reminiscent of the earlier inner product calculation and can also be directly computed. Only the first term of (9) cannot be computed directly since it requires the encryption of $x(i)^2$ whereas only the encryption of $x(i)$ is given. In fact, obtaining $[\![x(i)^2]\!]$ from $[\![x(i)]\!]$ is a problem analogous to (7), and requires a simplified version of the secure multiplication protocol. We conclude that on the one hand computing squared error distances and derived versions such as perceptually weighted squared errors can be done directly on encrypted data. On the other hand, the computations are more involved than the inner product calculation, as they require an interactive protocol for squaring M encrypted samples.

USING SECURE TWO-PARTY COMPUTATION

The homomorphic property comes in handy for linear signal processing operations on entire signals and allows for efficient implementations of inner products and squared error distance calculations. There is, of course, a large class of signal processing operations for which homomorphic properties are not immediately helpful. We encountered the example of multiplying two numbers that have been encrypted, but also common operations such as division by a constant or by an encrypted number, exponentiations, logarithms, and trigonometric functions, require secure two-party, or in general, MPC [24]–[26]. Secure MPC is arguably the most important approach in cryptography to evaluate an arbitrary function $f(x_1, x_2, \ldots, x_m)$, where the input x_j is private to the jth party. In other words, the m parties need each other's input to be able to jointly evaluate the function $f(.)$, but each party wishes to keep its input secret. We point out that in traditional secure MPC, the function $f(x_1, x_2, \ldots, x_m)$ is known to all parties involved in the computation. In encrypted signal processing this is the case if just the parameters of the function are private to the service provider. However, as mentioned earlier, (the structure of) the function itself might also be private as it represents commercial value to the service provider. This distinction can lead to different solutions in server-oriented encrypted signal processing than in traditional secure MPC.

For many years, MPC has been considered to be of theoretical interest only. In the last few years great improvements and actual implementations have made MPC of practical interest, even if the protocols are still computationally costly. After initial applications in domains such as electronic voting, actioning, and data mining, secure MPC is now becoming part of solutions for privacy-protected signal processing [26].

MPC comes in two flavors. The first type are arithmetic protocols. These protocols are often based on additively homomorphic encryption, and involve operations such as additions, multiplications, and blinding of encrypted integers. We give the example of comparing an encrypted and unencrypted number using arithmetic two-party computation in "Arithmetic Comparison Protocol." Arithmetic two-party protocols are interactive in that they require both parties to take part in the computation. A characteristic of these protocols is that it is usually not possible to derive the protocol from first principles. Many protocols are derived from prototypical solutions, for instance from secure comparison or secure multiplication.

The second kind of MPC is based on formulating the joint function $f(.)$ as a circuit of Boolean operations, and next protecting each Boolean operation by garbling input and output using private key encryption. "MPC Using Garbled Circuits" illustrates the principles of garbled circuits on the problem of securely comparing two plaintext numbers. Garbled circuits also require interaction between the parties. Bob creates and garbles the circuit, after which Alice evaluates the circuit without knowledge of Bob's private keys. Garbled circuits rely on private-key encryption rather than public-key encryption (see "Private Key and Public Key Cryptography"). Since no complex operations are required such as exponentiations, garbled circuits can be evaluated efficiently. However, the formulation of $f(.)$ as a series of garbled Boolean operations is memory and consequently communication intensive. Furthermore, garbled circuits typically require a cryptographic protocol known as oblivious transfer (OT) to let Alice determine the right input keys corresponding to her private input to the garbled circuit. OT protocols are often computation and communication more demanding than the creation and evaluation of the garbled circuit itself.

Since MPC protocols can be used to evaluate an arbitrary function $f(.)$, an obvious alternative way to achieve the secure evaluation of (1), (5), and (8) is to use an arithmetic protocol or a garbled circuit. While such solutions can certainly be derived, it is a matter of relative efficiency. The homomorphic operations central to (4), (6), and (9) are computationally quite efficient. Whether exploiting homomorphic encryption is to be preferred over MPC is quite dependent on the signal processing task. It is clear, however, that few signal processing problems can be implemented using only additions on homomorphically encrypted data. For instance, the encrypted versions of (1) and (8) also require MPC for some operations, yielding hybrid overall solutions. As a rule of thumb, problems that predominantly require algebraic operations can usually be implemented efficiently using homomorphic encryption; problems that require many nonlinear operations or problems that rely on access to individual bits are better implemented using garbled circuits.

Finding practically efficient MPC protocols is of prime importance, not only for signal processing problems but for the field of applied cryptography at large. Moore's law obviously helps, as does continuously increasing communication data rates. But also the protocols themselves need to become more efficient. In the precomputing approach, parts of the protocols that are not dependent on the party's inputs are computed offline, for instance in idle time of a server. Precomputations might involve, for instance, key generation, as well as the generation, encryption and exponentiation of random values (see "Additively Homomorphic Public Key Encryption"). Especially in signal processing where we deal with large volumes of samples or pixels, the challenge is to design protocols that allow for as much precomputing as possible, thus significantly increasing MPC efficiency in the online phase of the protocol.

USING BLINDING

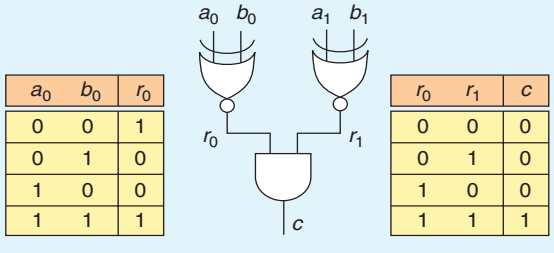
The final cryptographic primitive often used in signal processing is blinding. In the usual cryptographic context, blinding is a technique whereby Bob hides a value in such a way that Alice can decrypt the value, perform some operation(s), re-encrypt and send the result to Bob. For instance, in "Arithmetic Comparison Protocol," blinding is used in calculating $d \bmod 2^{\beta}$. Here we give another relevant example, specifically secure squaring and multiplication. Let us assume that Bob has available the value $[\![x]\!]$ encrypted with Alice's public key. Bob invokes

MPC USING GARBLED CIRCUITS

Garbled circuits provide a generic approach to secure function evaluation [109]. We illustrate the construction of a garbled circuit for verifying if two numbers are equal. Let us assume that Alice's and Bob's private values a and b, respectively, consist of 2 b ($a_1 a_0$) and ($b_1 b_0$). The output $c \in \{0, 1\}$ of the protocol is $a = b \Leftrightarrow c = 1$. First, Alice and Bob formulate their joint function as a series of Boolean operations on the bits of their private inputs. For the equality function we easily derive $c = \overline{(a_0 \oplus b_0)} \cdot \overline{(a_1 \oplus b_1)}$. The following Boolean circuit with two XOR-gates and one AND-gate in Figure S1 implements the equality function. As an illustration, Figure S1 shows two of the logic tables. Bob then constructs an encrypted version of the circuit in the following way.

- For each input bit a_0, a_1, b_0, b_1 he selects two private keys, one for each bit-value. For example, for a_0 he selects key $K_{a_0}^0$ for $a_0 = 0$, and $K_{a_0}^1$ for $a_0 = 1$. In total Bob selects eight uniformly distributed random keys: $K_{a_0}^0, K_{a_0}^1, K_{a_1}^0, K_{a_1}^1, K_{b_0}^0, K_{b_0}^1, K_{b_1}^0$, and $K_{b_1}^1$.
- Bob also selects private keys for all the intermediate connections of the circuit. In our example, he uses the following four uniformly distributed random keys for r_0 and r_1: $K_{r_0}^0, K_{r_0}^1, K_{r_1}^0$, and $K_{r_1}^1$.
- Bob replaces the entries in each gate's logic table by encrypting the output key with the input keys corresponding to the tables entry. For instance, for $a_0 = 0, b_0 = 0$ we have $r_0 = 1$. The output key corresponding to $r_0 = 1$ is $K_{r_0}^1$ that is encrypted with the input keys $K_{a_0}^0$ and $K_{b_0}^0$, i.e., $\mathcal{E}_{K_{b_0}^0}(\mathcal{E}_{K_{a_0}^0}(K_{r_0}^1))$ [1].
- The encrypted logic table for the first XOR-gate of the circuit thus becomes

Input a_0	Input b_0	Output
$K_{a_0}^0$	$K_{b_0}^0$	$\mathcal{E}_{K_{a_0}^0}(\mathcal{E}_{K_{b_0}^0}(K_{r_0}^1))$
$K_{a_0}^0$	$K_{b_0}^1$	$\mathcal{E}_{K_{a_0}^0}(\mathcal{E}_{K_{b_0}^1}(K_{r_0}^0))$
$K_{a_0}^1$	$K_{b_0}^0$	$\mathcal{E}_{K_{a_0}^1}(\mathcal{E}_{K_{b_0}^0}(K_{r_0}^0))$
$K_{a_0}^1$	$K_{b_0}^1$	$\mathcal{E}_{K_{a_0}^1}(\mathcal{E}_{K_{b_0}^1}(K_{r_0}^1))$.



[FIGS1] Boolean circuit implementing the equality function $a = b <=> c = 1$.

A similar table can be derived for the second XOR-gate. For the garbled table of the AND-gate implementing $c = r_0 \cdot r_1$, we have

Input r_0	Input r_1	Output c
$K_{r_0}^0$	$K_{r_1}^0$	$\mathcal{E}_{K_{r_0}^0}(\mathcal{E}_{K_{r_1}^0}(0))$
$K_{r_0}^0$	$K_{r_1}^1$	$\mathcal{E}_{K_{r_0}^0}(\mathcal{E}_{K_{r_1}^1}(0))$
$K_{r_0}^1$	$K_{r_1}^0$	$\mathcal{E}_{K_{r_0}^1}(\mathcal{E}_{K_{r_1}^0}(0))$
$K_{r_0}^1$	$K_{r_1}^1$	$\mathcal{E}_{K_{r_0}^1}(\mathcal{E}_{K_{r_1}^1}(1))$.

Since this is the final gate of the circuit, the output is the encryption of the output bit c rather than an encryption key.

- Bob randomizes the positions of the four entries in the garbled tables to break the association between entry number and input-output bit values.
- Bob sends the resulting garbled circuit to Alice while keeping all keys $K_{a_j}^{0/1}, K_{b_j}^{0/1}$, and $K_{r_j}^{0/1}$ secret. That is, Bob sends Alice only the last column of the garbled logic tables. Bob also specifies which gate outputs have to be used as inputs for subsequent gates.

It is important to notice that the encryption keys are constructed in such a way that Alice can correctly decrypt only one output key $K_*^{0/1}$ per gate, depending on the provided input keys. Alice can be informed which key has been decrypted correctly by, for instance, appending each key-to-be-encrypted with a number of trailing zeros, thus replacing $\mathcal{E}_{K_a}(\mathcal{E}_{K_b}(K))$ by $\mathcal{E}_{K_a}(\mathcal{E}_{K_b}(K \,|\, 00\cdots0))$.

To start the evaluation of the garbled circuit, Alice needs the keys associated with Bob's input and the keys associated with her own input bits $K_{a_j}^{0/1}$. Bob directly sends his keys to Alice since she cannot retrieve Bob's bits from the keys. However, Bob should not know which keys among $K_{a_j}^{0/1}$ he has to transfer to Alice as this might reveal Alice's bits to Bob. To solve this apparent deadlock, Bob and Alice run a oblivious transfer (OT) protocol, which allows Bob to transfer the proper keys to Alice without Bob learning which keys have been selected [9], [110]. Once Alice knows $K_{a_j}^{0/1}$ and $K_{b_j}^{0/1}$ she can decrypt the output of the first two gates of the circuit. The outputs contain the keys $K_{r_j}^{0/1}$ associated to the actual value of r_0 and r_1. Alice uses these keys to evaluate the output of the final gate of the circuit, which reveals to her whether $a = b$ or not.

The above highly structured procedure can easily be generalized, thus permitting the private computation of virtually any function, including the comparison function in "Arithmetic Comparison Protocol," which can be expressed by a nonrecursive Boolean circuit. We conclude by observing that garbled circuits are implemented using symmetric encryption only. This avoids the need for long keys and the necessity to perform computationally expensive operations. A drawback is, however, the need to describe the (potentially complicated) function at the level of logical gates, which might lead to very large circuits.

an interactive protocol with Alice to obtain $[\![x^2]\!]$ as follows. He first blinds the value x by homomorphically adding a random value r: $[\![z]\!] = [\![x + r]\!] = [\![x]\!] \cdot [\![r]\!]$. After sending the value $[\![z]\!]$ to Alice, she decrypts, squares z and re-encrypts to $[\![z^2]\!]$. Alice sends the value $[\![z^2]\!]$ to Bob, who can now compute $[\![x^2]\!]$ because he knows r and because of the homomorphic property

$$[\![x^2]\!] = [\![z^2 - (2xr + r^2)]\!] = [\![z^2]\!] \cdot [\![x]\!]^{-2r} \cdot [\![r^2]\!]^{-1}. \tag{10}$$

The random properties of r must be chosen such that $x + r$ does not leak information to Alice. If, for instance, $x \in \mathbb{Z}_n$, then r must be uniformly distributed over $\mathbb{Z}_n$. We can easily extend the above protocol to secure multiplication of two encrypted numbers $[\![x]\!]$ and $[\![y]\!]$. Bob homomorphically blinds $[\![x]\!]$ and $[\![y]\!]$ with random values r_x and r_y, respectively. Alice decrypts $[\![z_x]\!] = [\![x + r_x]\!]$ and $[\![z_y]\!] = [\![y + r_y]\!]$, multiplies, and re-encrypts the result $[\![z_x z_y]\!]$. Note that even though Alice decrypts z_x and z_y, she does not learn x and y since these have been blinded by the random values r_x and r_y, respectively. Bob then calculates the final result of the secure multiplication $\odot$ as

$$\begin{aligned}[\![x]\!] \odot [\![y]\!] &= [\![z_x z_y - (x r_y + y r_x + r_x r_y)]\!] \\ &= [\![z_x z_y]\!] \cdot [\![x]\!]^{-r_y} \cdot [\![y]\!]^{-r_x} \cdot [\![r_x r_y]\!]^{-1}.\end{aligned} \tag{11}$$

In signal processing, an alternative way of using blinding has been developed. In this approach, which is sometimes called data perturbation, random components $r(i)$ are added to signal values $x(i)$ in such a way that 1) the individual values $x(i)$ are "sufficiently" hidden, and 2) the random components cancel out in the targeted signal processing algorithm [27]. As a straightforward example, consider the following alternative to (6) to obtain a privacy-preserving implementation of (5), and let $y(i) = 1$ for the sake of simplicity. If Alice adds random values $r(i)$ to her input signal $x(i)$, then Bob computes the desired (now unencrypted) output $\mathcal{I}$ degraded by a random component $\sum_{i=1}^{M} r(i)$. The random properties of $r(i)$ are chosen such that 1) individual signal values $x(i) + r(i)$ are sufficiently random, and 2) the degrading term is small. The degrading term can even be made equal to zero by choosing $r(M) = -\sum_{i=1}^{M-1} r(i)$. Obviously, depending on the random properties of $r(i)$, this signal processing-inspired blinding approach might be significantly less secure, and even arguably insecure, than cryptographic blinding. Nevertheless, data perturbation can be an attractive alternative for some parts of encrypted signal processing since its computational requirements are much lower.

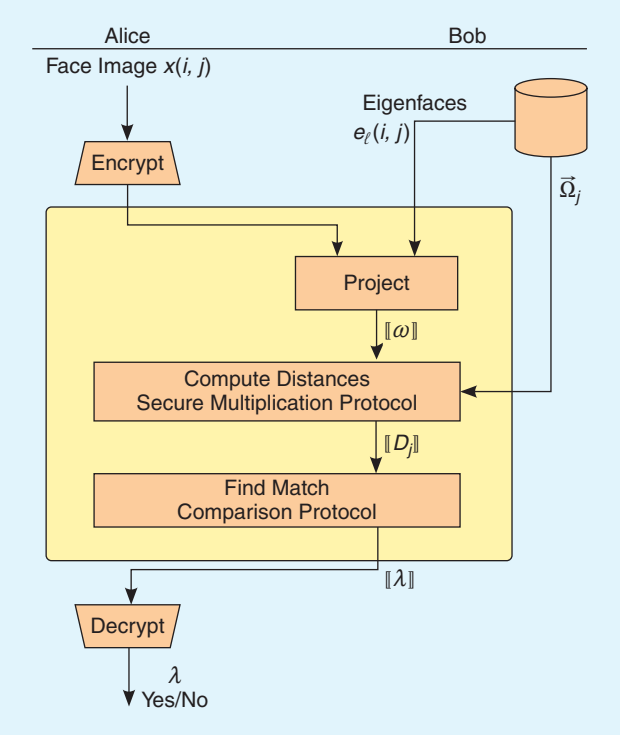
PRIVACY-PROTECTED FACE RECOGNITION

Next, we address a number of well-known privacy-sensitive signal processing problems and show how the ideas put forward in the previous sections can be used. We commence by describing a privacy-protected face recognition system that is based on eigenfaces [28], [29]. The solution allows both the biometric data and the authentication result to be hidden from the server that performs the matching [30], [31]. Related approaches for privacy-protected face recognition based on other face features have been published in [32], and for obscuring faces while analyzing suspected behavior in video surveillance in [33]–[37].

Alice owns a face image (the query image) and Bob owns a database containing a collection of face images (or corresponding feature vectors) of individuals. Alice and Bob wish to determine whether the picture owned by Alice shows a person whose data is in Bob's database. While Bob accepts that Alice might learn basic parameters of the face recognition system, he considers the content of his database private data that he is not willing to reveal. In contrast, Alice trusts Bob to execute the algorithm correctly, but is not willing to share with Bob either the query image or the recognition result. Finally, Alice will only learn if a match occurred. In a real-world scenario, Bob could be an honest-but-curious police organization, and Alice could be some private organization running an airport or a train station. It is of common interest to identify certain people, but it is generally considered too privacy intrusive to use Bob's central server directly for identification, as this would allow him, for instance, to create profiles of travelers.

The face recognition system we use is illustrated in Figure 3. In this figure, the yellow part includes the operations that need to be performed on encrypted data to protect the user's privacy. The system has five basic steps.

- Alice submits the query image $x(i,j)$ to Bob. The query image contains a total of M pixels.
- Bob transforms the query face image into a characteristic feature vector $\vec{\omega}$ of a low-dimensional vector space, whose basis is composed of L eigenfaces $e_\ell(i,j)$ ($\ell = 1, \ldots, L$). The eigenfaces are determined through a training phase that Bob has already carried out when building the faces database.



[FIG3] Block diagram of privacy-protected face recognition based on eigenfaces.

[TABLE 2] COMPUTATIONAL COMPLEXITY OF THE PRIVACY-PROTECTED FACE RECOGNITION SYSTEM (FROM [30]).

	ALICE	BOB
ENCRYPTION	$O(M+N_f\beta)$	$O(N_f(L+\beta))$
DECRYPTION	$O(N_f\beta)$	-
MULTIPLICATION	-	$O(ML+N_f(L+\beta^2))$
EXPONENTIATION	-	$O(ML+N_f(L+\beta^2))$

- Bob computes the Euclidean distances $\mathcal{D}_j$ between the feature vector $\vec{\omega}$ of the query image and the feature vector $\vec{\Omega}_j$ $(j=1,\ldots,N_f)$ of each face enrolled in the database. The total number of faces in the database is N_f.
- Bob finds the best matching face (smallest Euclidean distance $\mathcal{D}_{\min}$), and determines if the face is similar enough by comparing the distance $\mathcal{D}_{\min}$ to a threshold T.
- Bob sends the outcome $\lambda \in \{0,1\}$ of the recognition process to Alice, where $\lambda = 1 \Longleftrightarrow T < \mathcal{D}_{\min}$.

The knowledge provided in the sections "A Simple Privacy-Protected Signal Processing Algorithm" and "Processing of Encrypted Signals" allows us to see how a privacy-protected version of the face recognition system can be designed. Alice encrypts the individual pixels of the query image $x(i,j)$ using her public key encryption scheme, and sends $[\![x(i,j)]\!]$ to Bob. The transformation of the query face image into the feature vector $\vec{\omega}$ requires the calculation of L inner products of $x(i,j)$ and eigenfaces $e_\ell(i,j)$. Similar to the result in (6), we obtain for the L elements in $\vec{\omega}$

$$[\![\omega_\ell]\!] = \prod_{i,j} [\![x(i,j)]\!]^{e_\ell(i,j)} \quad \ell = 1,2,\ldots L. \tag{12}$$

Bob calculates the distances between the encrypted feature vector $[\![\vec{\omega}]\!] = ([\![\omega_1]\!],[\![\omega_2]\!]\ldots,[\![\omega_L]\!])$ and unencrypted feature vectors $\vec{\Omega}_j$. This problem was addressed in the section "Processing of Encrypted Signals," and the result in (9) directly applies. As discussed, the evaluation of (9) can be done by Bob on his own, except for the calculation of $[\![\omega_\ell^2]\!]$ from $[\![\omega_\ell]\!]$. For these terms, Bob and Alice invoke the secure squaring protocol described in the section "Using Blinding."

To find the minimum distance $[\![\mathcal{D}_{\min}]\!]$ among the N_f encrypted distances $[\![\mathcal{D}_j]\!]$, and to compare this minimum to the threshold T, Bob again runs a two-party protocol with Alice. To find the minimum, a straightforward recursive procedure can be used. In the first step, Bob compares distances $[\![\mathcal{D}_{2j+1}]\!]$ and $[\![\mathcal{D}_{2j+2}]\!]$ for $j=0,1,\ldots,N_f/2$. This requires a protocol for comparing two encrypted values, for instance, similar to the protocol in "Arithmetic Comparison Protocol" [30] or a garbled circuit solution [31]. To prevent Bob learning which of the two values is smallest, Alice rerandomizes the output of the comparison protocol. Rerandomization is the process of changing the random value r used in the (for instance, Paillier) encryption process (see "Additively Homomorphic Public Key Encryption"). The rerandomized encryption of the smaller distance is retained after the comparison. After this step we have $N_f/2$ values left, and we repeat the procedure for the remaining encryptions, and so forth. After $\log_2(N_f)$ iterations there is exactly one encryption left, the minimum $[\![\mathcal{D}_{\min}]\!]$. Bob uses a protocol such as the one in "Arithmetic Comparison Protocol" to determine whether $[\![\mathcal{D}_{\min}]\!]$ is smaller than the threshold T, and finally returns the encrypted result $[\![\lambda]\!]$ to Alice.

[TABLE 3] RUN TIME (IN SECONDS) OF THE PRIVACY-PROTECTED FACE RECOGNITION SYSTEM.

NUMBER OF FACES N_f	ARITHMETIC MPC [30]	ARITHMETIC MPC [30] WITH PRECOMPUTATIONS	GARBLED CIRCUITS [31]
10	24.0	8.5	13.2
50	26.0	10.0	13.5
100	29.0	11.5	13.8
200	34.2	14.5	15.1
300	39.6	17.5	16.4

The computational complexity of the above encrypted signal processing algorithms is given in Table 2 as a function of the various parameters. The parameter β is the number of bits required to represent the distances $\mathcal{D}_j$. The encryption of the query image pixel-by-pixel takes considerable computational power due to the sheer amount of data. Other computationally expensive parts of the algorithm are the $O(ML)$ exponentiations in (12) and the $O(N_f(L+\beta^2))$ exponentiations in the comparison protocol. The complexity of the privacy-protected algorithm grows linearly with the number of pixels M and the size of the database N_f.

A number of publications have studied the implementation of the above privacy-protected face recognition system [30], [31]. Table 3 summarizes the run-time results of two implementations for images of size 92×112 pixels. Both use homomorphic operations for the calculation of $[\![\omega_\ell]\!]$. The result from [30] is based on using arithmetic MPC protocols for the various comparison calculations and [31] uses garbled circuits. We refer to these papers for details on the choice of parameters and various implementation and optimization aspects. In both cases, however, the largest part of the CPU time is spent on the comparison protocols. The table also shows that it pays off to make a given protocol more efficient by employing precomputations that are carried out offline in idle time of the server.

PRIVACY-PROTECTED K-MEANS CLUSTERING

The second privacy-sensitive signal processing problem we consider in more detail is cluster analysis using the K-means algorithm. Clustering is a well-studied combinatorial problem in data mining [38]. It concerns finding a structure in a collection of unlabeled data. The K-means algorithm partitions a given data set into K clusters while minimizing an overall error measure. Privacy is important if the unlabeled data reflect user information, or when the assignment of data to a particular cluster is privacy sensitive, as in the following two examples.

- Users provide information about themselves to find kindred spirits. Based on the data provided, the K-means algorithm clusters users into groups that have a high level of similarity. Such social grouping might be based on tastes (delicious.com), dating, television viewing behavior, or chronic diseases (www.patientslikeme.com). The honest-but-curious server carrying

out the clustering should not be aware of either the users' unlabeled data or the group assignment of an individual user.

- Two companies, each owning a customer relationship management (CRM) database, wish to determine market segmentations or customer profiles based on their joint CRM data. They are not willing to share their data because of market competition, or because legislation does not allow this. The server carrying out the clustering should therefore not have access to the plaintext CRM data.

Privacy-protected clustering is a relatively well-studied problem in the pattern recognition community. Comprehensive overviews of approaches to privacy-preserving data mining are given in [39]–[41]. Cryptographic techniques toward privacy-preserving K-means clustering are based on MPC and secure permutations [42], random shares [43]–[45], or secure multiplications [46]. The approaches in [47] and [27] are examples of signal processing-based blinding techniques. We follow the cryptographic approach described in [48]. This approach is easy to explain because of its similarities to the secure face-recognition solution. There are, however, a number of interesting differences between the two solutions.

Figure 4 illustrates the K-means algorithm. Again, the yellow part includes the operations that will be performed on encrypted data to protect the user's privacy. The K-means algorithm consists of the following steps:

- The ith user Alice ($i \in 1, \ldots, N_u$) has an L dimensional data vector $\vec{p}_i$ with preferences on the basis of which the clustering will take place.
- We need to find C cluster centres or centroids $\vec{c}_j$ that best represent the N_u user data vectors. We use the Euclidean distance as error measure. The algorithm starts with C randomly selected centroids.
- Bob computes for each user data vector the distance to each centroid as

$$\mathcal{D}_{i,j} = \sum_{\ell=1}^{L} (p_i(\ell) - c_j(\ell))^2, \tag{13}$$

for $i \in 1, \ldots, N_u$, and $j \in 1, \ldots, C$.

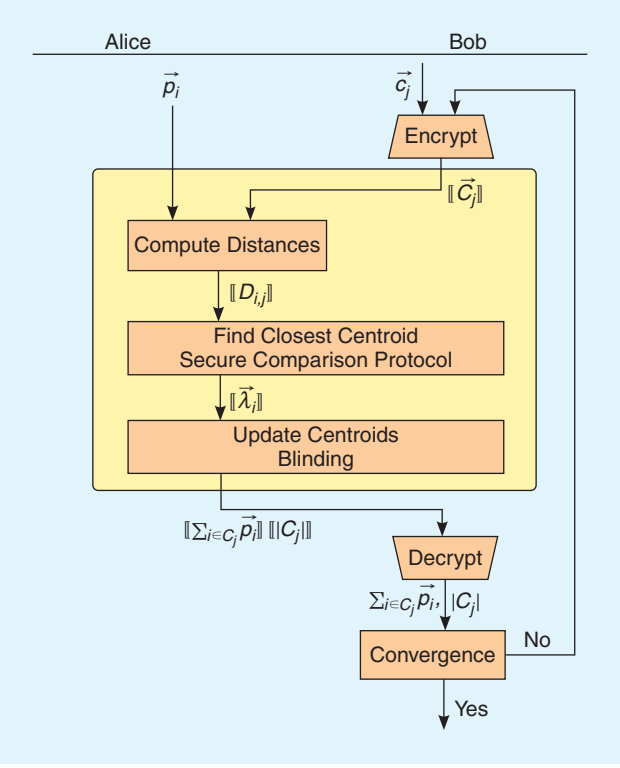
- Bob assigns Alice and her associated data vector $\vec{p}_i$ to the cluster C_m of the closest centroid $\vec{c}_m$, in this way partitioning the user data vectors into C subsets:

$$m = \arg\min_j \mathcal{D}_{i,j} \Longleftrightarrow i \leftarrow m, \vec{p}_i \leftarrow \vec{c}_m. \tag{14}$$

- As a final step, Bob computes the updated centroid $\vec{c}_j$ for each cluster C_j based on the subset of user data vectors $\vec{p}_i$ assigned to C_j:

$$\vec{c}_j = \frac{1}{|C_j|} \sum_{i \in C_j} \vec{p}_i. \tag{15}$$

- The algorithm terminates after a fixed number of iterations, or if the average distance between user data vectors and closest centroids has converged. If this is not the case, repeat the iterations starting from the third bullet.



[FIG4] Block diagram of privacy-protected K-means algorithm.

Different privacy-protected versions of the K-means algorithm have been developed depending on the privacy requirements. We consider the case where Alice only learns to which cluster index m she has been assigned. The data $\vec{p}_i$ of user Alice and the assignment of Alice to cluster C_m is privacy-sensitive information that the honest-but-curious server Bob and all other users should not learn. Bob is, however, allowed to learn the resulting centroids $\vec{c}_j$.

When we compare the calculation of the distances $\mathcal{D}_{i,j}$ in (13) to the distance calculation in the section "Processing of Encrypted Signals" and the matching process in the section "Privacy-Protected Face Recognition," we see strong similarities. Indeed, the expression for calculating the encrypted distances $\mathcal{E}_{PK}(\mathcal{D}_{i,j})$ is nearly identical to (9). However, an important difference is that we cannot let different users, say Alice and Charles, encrypt their private data $\vec{p}_i$ with their own public-private key pair since this will make it impossible to further combine results. As an illustration, consider (15), which requires the addition of user data vectors $\vec{p}_i$. The homomorphic addition of the vectors $\vec{p}_i$ assigned to cluster C_j is possible only if all $\vec{p}_i$ have been encrypted using a homomorphic public key cryptosystem with the same modulus mod n, specifically the same key. This of course will not be the case if we let users choose their own keys.

It is therefore Bob who selects a public-private key pair. He keeps the private decryption key to himself and provides the public encryption key to all users. We immediately realize that an encrypted data vector $\mathcal{E}_{PK}(p_i)$ is now secured toward other users, but not toward Bob. Hence, we cannot let Bob compute the distances $\mathcal{E}_{PK}(\mathcal{D}_{i,j})$ using encrypted user data $\mathcal{E}_{PK}(\vec{p}_i)$. Alternatively,

the users now compute the distances $\mathcal{E}_{PK}(\mathcal{D}_{i,j})$ using encrypted centroids $\mathcal{E}_{PK}(\vec{c}_j)$. The expression for encrypted distances can be derived analogously to (9). The result takes the following form

$$\mathcal{E}_{PK}(\mathcal{D}_{i,j}) = \left[\!\left[\sum_{\ell=1}^{L} p_i(\ell)^2\right]\!\right] \cdot \prod_{\ell=1}^{L} [\![c_j(\ell)]\!]^{-2p_i(\ell)} \cdot \left[\!\left[\sum_{\ell=1}^{L} c_j(\ell)^2\right]\!\right]. \quad (16)$$

Alice can easily compute $\mathcal{E}_{PK}(\mathcal{D}_{i,j})$ since she can encrypt $[\![\sum p_i(\ell)^2]\!]$ using Bob's public key, and Bob provides the encryptions $[\![c_j(\ell)]\!]$ and $[\![\sum c_j(\ell)^2]\!]$.

To find the closest centroid for user i, Alice needs to find the minimum among C encrypted distances $\mathcal{E}_{PK}(\mathcal{D}_{i,j})$ with $j \in 1,\ldots,C$. We have already seen a two-party protocol solution to this problem in the section "Privacy-Protected Face Recognition" and in "Arithmetic Comparison Protocol." The result is an encrypted vector $[\![\vec{\lambda}_i]\!]$ of dimension C, with $\lambda_i(m) = 1$ if user i has been assigned to cluster C_m as in (14), and $\lambda_i(j) = 0$ for all other values of j. Note that at this point Alice does not yet know to which cluster C_m she has been assigned as she cannot decrypt the elements of the vector $[\![\vec{\lambda}_i]\!]$.

If Alice sends Bob the encrypted $L \times C$ matrix

$$[\![p_i(\ell) \cdot \lambda_i(j)]\!] = [\![\lambda_i(j)]\!]^{p_i(\ell)}, \ \ell = 1,\ldots,L, \ j = 1,\ldots,C, \quad (17)$$

he can homomorphically compute the sum term in (15) needed for the updated cluster centroids as follows:

$$\mathcal{E}_{PK}(c_j(\ell)) = \prod_{i=1}^{N_u} [\![\lambda_i(j)]\!]^{p_i(\ell)} \ \ell = 1,\ldots,L. \quad (18)$$

Unfortunately, Alice cannot send Bob the matrix in (17) because Bob can decrypt this information using the private decryption key, thus directly obtaining Alice's private information $\vec{p}_i$. A possible solution is that users collaboratively blind their values $[\![\lambda_i(j)]\!]^{p_i(\ell)}$ in such a way that the blinding factors cancel out when computing the product in (18). Similarly, the dividing factor $\frac{1}{|C_j|}$ in (15) can be computed securely using the encrypted vector $[\![\vec{\lambda}_i]\!]$ with collaborative blinding. We refer to [48] for further details on the blinding process.

After the final iteration, Alice learns her cluster number by first computing the (encrypted) sum $\sum_{m=1}^{C} m \cdot \lambda_i(m)$ under homomorphism, and then running a secure decryption protocol with Bob. For the decryption protocol, Alice blinds the encrypted sum with a random number, sends the blinded value to Bob for decryption, and finally subtracts the random number.

The computational complexity is given in Table 4 as a function of the various parameters. The parameter β is the number of bits required to represent the distances $\mathcal{D}_{i,j}$ [see (13)]. The top half of the table gives the complexity of the privacy-protected algorithm described in this section. As we can see, the computationally most expensive part is the $O(C(L+\beta^2))$ exponentiations, where the term $O(CL)$ originates from (16) to (18) and $O(C\beta^2)$ originates from the comparison protocol that finds the closest centroid.

The bottom half of the table gives the computational complexity if we parallelize Alice's computation of (16) to (18) by packing C cluster centres into a single encrypted number. We refer to [49] and [13] for a discussion on data packing. The computational advantage is that the $O(CL)$ exponentiations reduce to $O(L)$. Nevertheless, the main complexity due to the comparison protocol, specifically $O(C\beta^2)$ exponentiations, remains and forms the computational bottleneck in actual implementations. Table 5 illustrates the run time of the algorithm without data packing, where about 80% of the run time is spent on repeated execution of the comparison protocol. As in the section "Privacy-Protected Face Recognition," precomputations reduce the run time by a factor of two to three.

An alternative cryptographic approach discussed in [49] uses, in addition to users and the clustering server, a curious-but-honest privacy service provider. Although this setup complicates the architecture of the solution (a third party is now needed to perform the K-means algorithm), the collaborative blinding is no longer needed. Further the computation burden of calculating distances $\mathcal{E}_{PK}(\mathcal{D}_{i,j})$ and finding the closest centroid $\vec{c}_m$ is removed from the user's shoulders. We expand on the idea of using a curious-but-honest privacy service provider in the following section, which deals with collaborative filtering.

[TABLE 4] COMPUTATIONAL COMPLEXITY OF THE PRIVACY-PROTECTED K-MEANS ALGORITHM (FROM [48]).

DESCRIBED ALGORITHM	ALICE	BOB
ENCRYPTION	$O(C(L+\beta))$	$O(C(L+\beta))$
DECRYPTION	-	$O(C(L+\beta))$
MULTIPLICATION	$O(C(L+\beta^2))$	$O(N_u CL)$
EXPONENTIATION	$O(C(L+\beta^2))$	-
MODIFIED WITH DATA PACKING	**ALICE**	**BOB**
ENCRYPTION	$O(C\beta)$	$O(L+C\beta)$
DECRYPTION	-	$O(C\beta)$
MULTIPLICATION	$O(C\beta^2)$	$O(N_u C)$
EXPONENTIATION	$O(L+C\beta^2)$	-

PRIVACY-PROTECTED CONTENT RECOMMENDATION

The third and final signal processing application that we discuss in greater detail is content recommendation. Here, our focus is on a popular technique used in recommender systems, specifically collaborative filtering [50], [51]. Rather than explicitly grouping users into clusters like the K-means algorithm does, collaborative filtering directly generates recommendations for users based on the profiles, preferences and ratings of similar

[TABLE 5] RUN TIME (IN MINUTES) OF THE PRIVACY-PROTECTED K-MEANS ALGORITHM (FROM [48]).

N_U		WITH PRECOMPUTATIONS
100	17.9	6.9
250	44.1	17.2
500	88.0	33.8
750	134.5	51.6
943	166.1	64.5

users. The item recommendations are specific to an individual user, but are based on information obtained from many users. As before, the objective of privacy protection is to hide from the service provider the profiles, preferences and ratings of users as well as the item recommendations to individual users.

Among the approaches to privacy-protected collaborative filtering, we again find blinding [52], [53] and cryptographic techniques such as differential privacy [54], MPC and secret sharing [55], and dimension reduction based on homomorphic encryption [56], [57]. We follow the approach in [58] and [59], in which the similarity between users and recommendations are computed on homomorphically encrypted user data. In addition to users and the recommendation service provider, we introduce a third party that allows us to exclude users from participation in cryptographic protocols.

We first summarize collaborative filtering for plaintext user data. The ith user Alice ($i \in 1,\ldots,N_u$) has a preference vector $\vec{p}_i$ of dimension L, which typically contains nonnegative ratings on content items. To generate recommendations for Alice, we follow a five-step procedure.

- Alice sends the recommendation service provider (RSP) her preference vector $\vec{p}_i$.
- The RSP computes the similarity between Alice and all other users based on a set of L_r items that all users have rated. For the purpose of simplicity, we assume that these rated items are the first L_r elements in preference vector $\vec{p}_i$, and that they have been normalized such that $\sum_{\ell=1}^{L_r} p_i(\ell) = 1$. A common measure to compute similarity between the ith user Alice and jth user Charles is the cosine similarity or inner product

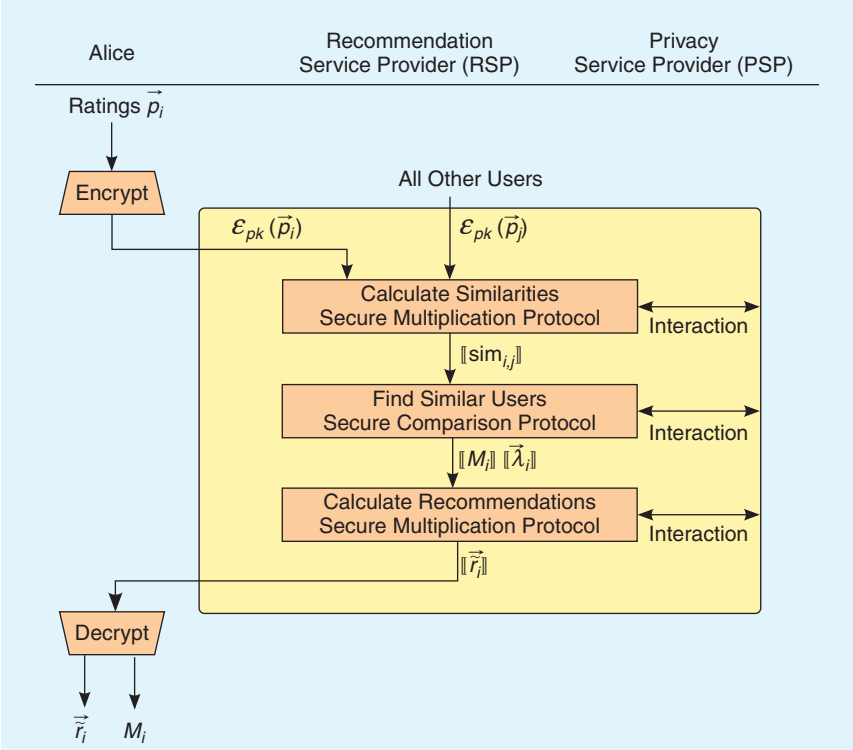
$$\text{sim}_{i,j} = \sum_{\ell=1}^{L_r} p_i(\ell) \cdot p_j(\ell). \tag{19}$$

- The RSP selects the top M_i most similar users by comparing their similarities to a threshold T.
- The recommendations $\vec{r}_i$ for user Alice on all $L - L_r$ remaining items are computed as the average ratings of the M_i most similar users

$$r_i(\ell) = \frac{1}{M_i} \sum_{m=1}^{M_i} p_m(\ell), \quad \ell = L_r + 1, \ldots, L. \tag{20}$$

- Alice obtains the recommendations $\vec{r}_i$ from the RSP.

We focus on the main signal processing operations of the above algorithm. The privacy requirements we consider are as follows. First, the preference vector $\vec{p}_i$, the recommendation vector $\vec{r}_i$ and the number of most similar users M_i are private to user i. Second, privacy-sensitive intermediate information, including the similarity values $\text{sim}_{i,j}$ and data of the M_i most similar users, are hidden from all users and the honest-but-curious recommendation service provider.



[FIG5] Block diagram of privacy-protected collaborative filtering.

Much like in the K-means algorithm, we need a public-private key pair common to all users to encrypt the ratings in the preference vectors $\vec{p}_i$ ($i \in 1,\ldots,N_u$). In principle, we can follow an approach similar to the one mentioned in the section "Privacy-Protected K-Means Clustering," whereby the recommendation service provider creates the public encryption and private decryption key, and users carry out a large proportion of the computations on encrypted data.

As an alternative that avoids the required user involvement, we introduce a privacy service provider (PSP). The honest-but-curious PSP is a third party to which part of the computation is outsourced [60]. The PSP creates the public-private key pair for the homomorphic public key cryptosystem. We can now store the database of all encrypted user data $\mathcal{E}_{PK}(\vec{p}_i)$ at the recommendation service provider as it does not have access to the decryption key of the PSP. Compared to the privacy-protected K-means algorithm, the users' computational complexity will become much lower. However, the users' privacy might be violated if the recommendation service provider and the privacy service provider collude by sharing encrypted data and/or keys. Hence, in this setup additional legal or organizational measures must be taken to prevent collusion between the RSP and PSP.

We achieve privacy-protected collaborative filtering as follows (also see Figure 5). Alice, like all other users, encrypts

her preference vector using the PSP's public key and sends $\mathcal{E}_{PK}(\vec{p}_i)$ to the RSP. The RSP aims to compute the encrypted value of the similarity (19) directly on users' encrypted data

$$[\![\mathrm{sim}_{i,j}]\!] = \left[\!\!\left[\sum_{\ell=1}^{L_r} p_i(\ell)\cdot p_j(\ell)\right]\!\!\right] = \prod_{\ell=1}^{L_r} [\![p_i(\ell)]\!] \odot [\![p_j(\ell)]\!]. \tag{21}$$

Note that the operation $\odot$ denotes the secure multiplication of two encrypted numbers, which is needed here since the RSP has access only to encrypted preference vectors. The RSP and PSP use the protocol explained in the section "Using Blinding" to carry out the secure multiplications. Also note that the privacy service provider does more than just key management: it also participates in some two-party computation protocols.

To find the M_i most similar users, the recommendation service provider compares the similarities $[\![\mathrm{sim}_{i,j}]\!]$ for user i to the threshold T. We have already seen such a comparison protocol in the section "Privacy-Protected Face Recognition"; the same two-party protocol described in "Arithmetic Comparison Protocol" can be used by the RSP and the PSP. The result is an encrypted N_u-dimensional vector $[\![\vec{\lambda}_i]\!]$. Here, $\lambda_i(j) \in \{0,1\}$, and $\lambda_i(m) = 1$ indicates that the similarity between the ith and mth user is larger than the threshold $\mathrm{sim}_{i,m} > T$. Computing the encrypted total number of similar users M_i is straightforward

$$\mathcal{E}_{PK}(M_i) = \left[\!\!\left[\sum_{j=1(j\neq i)}^{N_u} \lambda_i(j)\right]\!\!\right] = \prod_{j=1(j\neq i)}^{N_u} [\![\lambda_i(j)]\!]. \tag{22}$$

The sum term of the secure equivalent of (20) is obtained by first multiplying $\lambda_i(j)$ and $\vec{p}_j$, and then adding the products:

$$\begin{aligned}\mathcal{E}_{PK}(\tilde{r}_i(\ell)) &= \left[\!\!\left[\sum_{j=1(j\neq i)}^{N_u} \lambda_i(j)\cdot p_j(\ell)\right]\!\!\right] \\ &= \prod_{j=1(j\neq i)}^{N_u} [\![\lambda_i(j)]\!] \odot [\![p_j(\ell)]\!]. \end{aligned}\tag{23}$$

Note that in this expression the summation runs over all users (except user i itself), since the RSP does not know for which users $\lambda_i(m) = 1$, as these values are encrypted. The product $\lambda_i(j)\cdot p_j(\ell)$ selects the user preference vectors to be included in the summation. As in (21), the RSP and the PSP need to run a secure multiplication protocol for the product of the encrypted values $[\![\lambda_i(j)]\!]$ and $[\![p_j(\ell)]\!]$. The recommendation service provider sends encrypted results $\mathcal{E}_{PK}(M_i)$ and $\mathcal{E}_{PK}(\vec{\tilde{r}}_i)$ to Alice, who decrypts the results by running a simple (blinding-based) decryption protocol with the privacy service provider [59]. Alice obtains the final recommendations by dividing the elements in $\vec{\tilde{r}}_i$ by M_i.

The computational complexity is shown in Table 6. The parameter β is the number of bits to represent the similarity values $\mathrm{sim}_{i,j}$. It is immediately clear that thanks to the introduction of the PSP, users (like Alice) have a relatively low complexity of just $O(L)$ encryptions. The main complexity is the $O(N_uL)$ encryptions and exponentiations of the recommendation service provider, and the $O(N_u(L+\beta))$ encryptions of the privacy service provider. In both cases, this complexity is caused by the secure multiplication protocol needed in (21) and (23). This computational complexity can be reduced significantly by packing subsets of the user's ratings into a single encrypted number. We refer to [59] for details on the data packing operations and the resulting parallelization of (21) and (23). The effect of the data packing operation can best be illustrated using run time figures from an actual implementation. Table 7 shows the run times of the straightforward implementation of the collaborative filtering algorithm, as well as the version that packs 60 items of the users' rating data into a single encrypted number. The latter implementation, although still computationally expensive, is about 60 times faster.

OTHER EMERGING APPLICATIONS

Here, we briefly address other signal processing applications in which encrypted signal processing has been used or is emerging as a means for privacy protection and other security requirements.

NEURAL NETWORK CLASSIFICATION FOR e-HEALTH

The health-care industry is moving at a fast pace toward technologies offering personalized online self-service, medical error reduction, customer data collection, and more. These technologies have the potential of revolutionizing the way medical data are processed and made available to millions of users throughout the world. Respecting the privacy of customers is an important design constraint.

We consider a remote diagnosis service that classifies electrocardiograms (ECGs) provided by users into a number of categories, including healthy subjects and some disease categories. The server should carry out the classification without getting any knowledge about the private ECG data provided by the users. At the same time, the service provider is not willing to disclose algorithmic details since they represent the basis for

[TABLE 6] COMPUTATIONAL COMPLEXITY OF PRIVACY-PROTECTED COLLABORATIVE FILTERING (FROM [59]).

	RSP	PSP	ALICE
ENCRYPTION	$O(N_uL)$	$O(N_u(L+\beta))$	$O(L)$
DECRYPTION	-	$O(N_uL)$	-
MULTIPLICATION	$O(N_u(L+\beta))$	-	$O(L)$
EXPONENTIATION	$O(N_uL)$	-	-

[TABLE 7] RUN TIME (IN MINUTES) OF PRIVACY-PROTECTED COLLABORATIVE FILTERING (FROM [59]).

N_U	WITHOUT DATA PACKING	WITH DATA PACKING
1,000	260	5
5,000	1,500	26
10,000	3,100	51

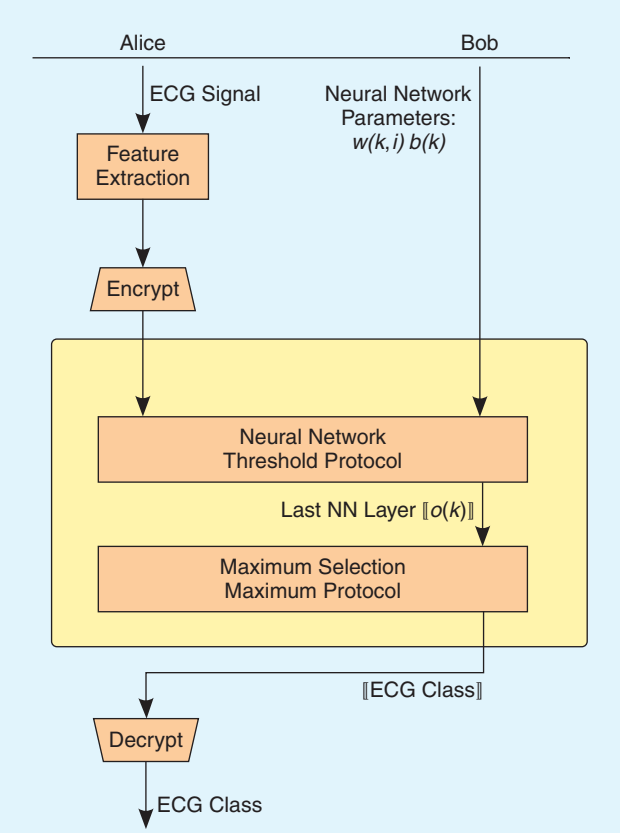
the diagnosis service it is providing. As illustrated in Figure 6, the classification is performed in two steps [61], [62]. First, a set of features are extracted by modeling the ECG signal as an autoregressive (AR) process. Six features are computed corresponding to the (four) coefficients of the AR model and two measures of the prediction error. Because this phase is quite complex yet fairly standardized, it is carried out on the plaintext ECG signal by Alice. The actual classification can be performed by Bob in several ways. In the following we describe a solution based on a neural network (NN) that has the advantage of being easily extendable to other scenarios. For an alternative solution based on branching programs, we refer to [61]. The NN computation protocol is fed with the features encrypted with Alice's public key and gives as an output the encrypted index of the class to which the ECG belongs. Only Alice can decrypt the classification index by using her secret key.

Let us consider the security requirements. On Alice's side, the situation is quite simple. She requires that Bob does not obtain any information about the ECG signal (features) and the result of the classification. As to Bob, he wants to keep secret the classification algorithm. As discussed in the section "Privacy Requirements," we see that it is impossible for Bob to keep all the details of the classification algorithm private. For instance, Alice knows the features the classifier relies on because she calculates these features. Similarly, we need to assume that Bob agrees to reveal to Alice that he uses an NN-based classifier. Even more, we also assume that Bob agrees to reveal to Alice the internal structure of the NN, that is, the number of layers and nodes in the neural network, and the shape of the activation functions, such as a threshold activation function. The only parameters that Bob wishes to protect are the NN weights and the offset value of the activation functions. This is a reasonable assumption since the NN weights and offsets are likely to be the result of an extensive and expensive training procedure carried out by Bob to set up an accurate diagnosis service.

With the solutions described in the previous sections in mind, the design of a protocol for private ECG classification is now straightforward. Let us focus on a single neuron in the NN. The operation the neuron performs consists of computing the weighted sum of the neuron inputs and offset, followed by thresholding the resulting weighted sum

$$v(i) = \sum_{j=1}^{M} x(i,j)w(i,j) + b(i),$$
$$y(i) = \begin{cases} 0 & \text{if } v(i) \le 0, \\ y(i) & \text{otherwise.} \end{cases} \quad (24)$$

Here $x(i,j)$ are the M inputs of the ith neuron, $w(i,j)$ are the corresponding weights, and $b(i)$ is the offset of the ith neuron. The weighted sum and offset addition can be computed privately by Bob using additively homomorphic encryption. Bob knows $w(i,j)$ and $b(i)$ as plaintext values, and he has an encrypted version of $x(i,j)$. For the first layer of the NN, $[\![x(i,j)]\!]$ are provided by Alice, for the other layers they are the encrypted output of previous neurons. Effectively, Bob uses (6) to obtain



[FIG6] **Block diagram of privacy-protected ECG classification.**

the encrypted output $[\![v(i)]\!]$. Then Bob and Alice run a secure comparison protocol (as in "Arithmetic Comparison Protocol" or "MPC Using Garbled Circuits"). The output is the encrypted value $[\![y(i)]\!]$, which Bob uses as input to the neurons of the next layer. The final classification result is the maximum of the outputs of the final layer in the neural network. To find this maximum, Alice and Bob run a protocol similar to the one described for the face recognition system in the section "Privacy-Protected Face Recognition." The performance of this privacy-preserving ECG analysis algorithm is that a single heart beat can be classified in 5–10 s [62].

The above privacy-preserving algorithm illustrates how encrypted signal processing is used to protect the privacy of users' medical information. With the technological advancement of genome sequencing and DNA profiling, the privacy protection of medical data will have to be taken to the next level. For instance, abuse of DNA information may open the door for discrimination based on genomics. Initial privacy protection approaches employ cryptographic protocols that compare DNA sequences while hiding the data of the query DNA sequence and the searched DNA sequence [63], [64].

BIOMETRIC MATCHING

The handling of biometric signals for person identification and access control requires privacy protection since biometric data

is indissolubly associated to the identity of the owner. In the section "Privacy-Protected Face Recognition," we have already seen a privacy-protected system for biometric-based person recognition. This section is concerned with privacy protection in generic biometric systems [65].

A biometric matching protocol consists of four steps: feature extraction, distance calculation, minimum distance selection, and thresholding. Biometric systems differ significantly with regard to the choice of features and the feature-based distance measure. In the section "Privacy-Protected Face Recognition," features were computed from encrypted face images. Alternatively, the features might also be extracted directly from the plaintext data by the owner of the biometric signal in cases where is it acceptable if the owner of the biometric data is aware of the features used by the server. The computational advantage is that more complicated features can be easily computed. Furthermore, the amount of data that needs to be transmitted and encrypted is smaller in case features are extracted on the client side; this might also reduce the bottleneck of communicating large amounts of encrypted data.

Some of the feature-based distance measures are not very suitable for implementing in a privacy-protected setting as they require complicated two-party protocols, for instance if the biometric systems is based on minutiae [66]. It is therefore up to the designers of a biometrics system to select a proper set of features for which distances can be calculated easily in a privacy-protected fashion. The final steps of biometric matching are the minimum distance selection and thresholding, which can always be implemented using the protocols explained in the section "Privacy-Protected Face Recognition."

Fingerprint images are the most widely used biometric traits. A particular feature choice is the fingercode representation [68], which is illustrated in Figure 7. The fingerprint image is first divided into radial and angular sectors. The sectors are then filtered by a number of Gabor filters to determine the energy content of each sector into different frequency bands. Finally, the feature vector is formed by the sequence of a properly quantized version of sector energies. The fingercode representation is attractive for privacy-protected biometric matching since it allows for the use of the Euclidean distances to compare fingerprints [67], [69]. Computing Euclidean distances on encrypted features has been discussed in detail in the section "Privacy-Protected Face Recognition." The protocol for privacy-preserving fingercode-based authentication presented in [67] permits to perform fingercode matching against a database with 100 entries in about 40 s.

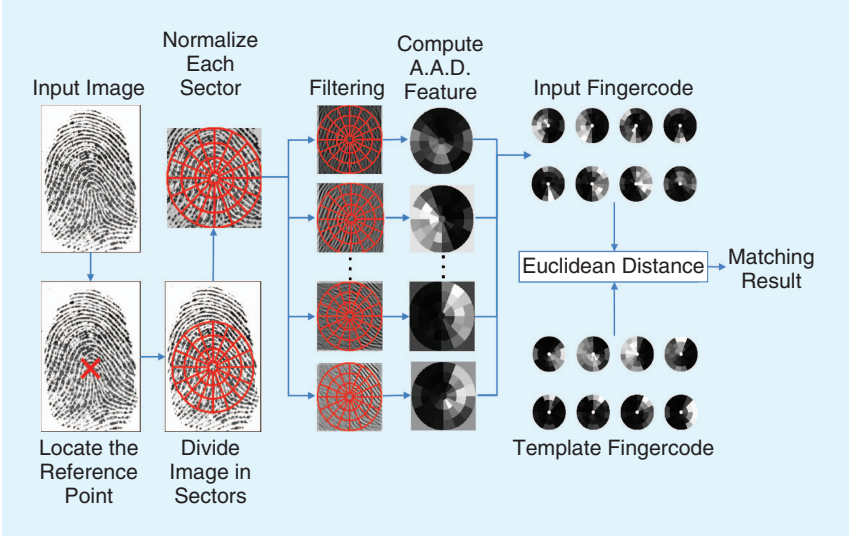
Iris images are increasingly used for biometric authentication. The images are first transformed into a binary template [70] by thresholding, then Hamming distances between templates are calculated using exclusive OR (XOR) operations. A privacy-preserving solution that computes the binary template from encrypted fingerprint images is described in [65]. We might also assume that the binary template is extracted from plaintext images. Since computing a Hamming distance with XORs is easily translated into a Boolean circuit, garbled circuits (see "MPC Using Garbled Circuits") then suit privacy-preserving iris-based authentication particularly well. The iris-based authentication system of [71] matches 2,048-b long iris templates in 60 ms.

WATERMARK DETECTION AND FINGERPRINTING FOR DRM

Digital watermarks offer solutions to digital rights management (DRM) of music, photographs, and videos, such as proving ownership. Robust watermarks have been designed to survive common media processing operations and watermark-removal attacks such as compression and filtering unless an unreasonable amount of damage has been inflicted upon it [72], [73]. A watermark consists of a very small but structured amount of noise that is added to media such that it imperceptible to the user. Watermarking algorithms are usually publicly known. However, the particular structure of an individual watermark is kept secret as this knowledge makes it very easy to remove the watermark. We consider the simplest watermarking approach of all, specifically the one that just adds a small amount of pseudorandom noise $w(i)$ to the signal [74]. The seed that drives the pseudorandom generator for $w(i)$ essentially represents a secret key S

$$\hat{x}(i) = x(i) + \alpha w_S(i). \tag{25}$$

The parameter α controls the signal-to-watermark ratio, and $\hat{x}(i)$ is the watermarked signal. In watermarking, two security issues arise for which encrypted signal processing offers a solution, particularly secure watermark detection and privacy-protected watermark embedding,



[FIG7] Fingercode representation of fingerprints. (Figure used with permission from [67].)

which is also known as anonymous fingerprinting or buyer-seller protocols.

Secure watermark detection addresses the problem that an individual or device needs S to detect the presence of a particular watermark $w_S(i)$ [75]–[77]. This is because watermark detection is carried out by correlating $\hat{x}(i)$ and $w_S(i)$, and comparing the correlation ρ to a detection threshold T

$$\rho = \sum_{i=1}^{M} \hat{x}(i) w_S(i) \geq T \;\Rightarrow\; \text{watermark detected.} \tag{26}$$

We immediately see that the detector's knowledge of S to regenerate $w_S(i)$ makes it an easy target for adversaries wishing to remove the watermark from $\hat{x}(i)$. At the same time we realize that (26) is just an inner product as in (5), which offers us the possibility to encrypt one of the terms. A proposed solution is to homomorphically encrypt $w_S(i)$ with a private-public key pair (SK, PK). The detector computes the correlation using the encrypted watermark $\mathcal{E}_{PK}(w_S(i)) = [\![w_S(i)]\!]$ as

$$\mathcal{E}_{PK}(\rho) = \left[\!\!\left[\sum_{i=1}^{M} \hat{x}(i) w_S(i) \right]\!\!\right] = \prod_{i=1}^{M} [\![w_S(i)]\!]^{\hat{x}(i)}. \tag{27}$$

Since neither $w_S(i)$ nor S can be obtained from $[\![w_S(i)]\!]$, the detector will not leak information about the watermark. Clearly, the final detection result is known only after comparing $\mathcal{E}_{PK}(\rho)$ and T. As in the previous sections, a two-party protocol needs to be run with the party that holds the decryption key SK, for example the watermark embedder or a privacy service provider as in the section "Privacy-Protected Content Recommendation."

In anonymous fingerprinting, a merchant embeds a buyer's identity into a purchased signal [78]. The buyer provides an identity watermark $w_S(i)$ to the merchant, who embeds it into the purchased signal $x(i)$ and sends the fingerprinted signal $\hat{x}(i)$ to the buyer. If the merchant finds an illegally redistributed copy of $\hat{x}(i)$, he can recover the identity from the watermark and confront the violating buyer. The problem with this scenario is that the merchant can act as adversary by claiming that he found an illegally redistributed copy while this is just the copy he sent to the buyer.

A desirable setup is that the merchant anonymously embeds the buyer's identity, that is, the watermark, into the purchased signal [79]–[81]. To this end, we again use an additively homomorphic public key cryptosystem. If we let the buyer encrypt his identity watermark using public key PK, and the buyer sends $[\![w_S(i)]\!]$ to the merchant, then the merchant embeds the encrypted watermark sample by sample as follows:

$$\mathcal{E}_{PK}(\hat{x}(i)) = [\![\hat{x}(i) + w_S(i)]\!] = [\![\hat{x}(i)]\!] \cdot [\![w_S(i)]\!]. \tag{28}$$

Note that the result of this operation is an encrypted fingerprinted signal $[\![\hat{x}(i)]\!]$, which only the buyer can decrypt using his private key SK. Since the merchant does not see the plaintext identity watermark $w_S(i)$ and he has no access to SK, he is unable to produce $\hat{x}(i)$ himself and hence cannot act as adversary.

THE PROCESSING OF ENCRYPTED SIGNALS CONSTITUTES AN EXCITING COMBINATION OF CRYPTOGRAPHY AND SIGNAL PROCESSING.

SMART GRIDS

Smart grids are increasingly seen as a vital solution for the better modeling and management of power grids. Although there are clear benefits to deploying smart grids, the accurate and fine-grained measurement of household energy consumption triggers serious privacy concerns [82]. A vast amount of sensitive information can be derived from such measurements, e.g., types of electrical devices being used and the number of inhabitants. Since the inclusion of smart meters is essential to facilitate better grid management, privacy aspects of smart grids—including privacy-preserving billing and data collection—have been addressed in the literature in recent years [83]–[87].

MPC techniques are used to privately compute the total consumption in a smart grid without disclosing individual consumptions [88]. If x_i is the measurement obtained from ith smart meter sm_i, then in a neighborhood with N_s smart meters the total consumption calculated by a substation is $\mathcal{T} = \sum_{i=1}^{N_s} x_i$. To hide the individual measurements, sm_i splits its measurement x_i into N_s random parts $x_i(j)$ (usually called shares) such that $x_i = \sum_j x_i(j) \bmod n$, where n is a large integer. Each share is then encrypted using the public key PK_j of the jth smart meter sm_j.

Next, sm_i sends the encrypted shares to the substation except for its share $x_i(i)$. After receiving all encrypted shares from all smart meters, the substation multiplies the $N_s - 1$ shares that are encrypted with the jth smart meter's key PK_j. Since we use an additively homomorphic cryptosystem, we obtain

$$\tilde{\mathcal{T}}_j = \prod_{i=1(i\neq j)}^{N_s} \mathcal{E}_{PK_j}(x_i(j)) = \mathcal{E}_{PK_j}\left(\sum_{i=1(i\neq j)}^{N_s} x_i(j) \right). \tag{29}$$

The substation returns the ciphertext result $\tilde{\mathcal{T}}_j$ to sm_j for decryption. After decryption, sm_j adds its share $x_j(j)$ to the plaintext, and sends back the result to the substation. Finally, the substation adds the contributions from all smart meters to obtain the total consumption $\mathcal{T}$. The above method protects the individual measurements x_i using homomorphic encryption, but it introduces a significant data expansion, a large number of encryptions and considerable data transfer. Alternative methods are being studied that use the computationally less demanding data perturbation approaches of the section "Using Blinding" [89].

FUTURE DEVELOPMENTS IN PRIVACY-PROTECTED SIGNAL PROCESSING

The processing of encrypted signals constitutes an exciting combination of cryptography and signal processing.

This article has provided an introduction to various aspects of encrypted signal processing, in particular for the purpose of privacy protection. Encrypted signal processing can also be considered a specific approach to computational privacy, privacy by design, or privacy-enhancing technologies (PETs) [90]. In the section "Other Emerging Applications," however, we also indicated other reasons for manipulating encrypted signals.

THE CHALLENGE WILL BE TO FIND ENCRYPTED SIGNAL PROCESSING SOLUTIONS THAT BALANCE THE USER'S AND THE SERVICE PROVIDER'S INTERESTS IN A NEGOTIABLE FASHION.

The application of cryptographic techniques in the context of signal processing poses many new challenges for cryptographic researchers and signal processing researchers alike, as well as for the practitioners of signal processing algorithms in business. We elaborate on these challenges in this section.

We have seen several instances in which the combination of multiplicative and additive operations on encrypted data is highly desirable. This would remove the need for the expensive secure multiplication protocol (11) in, for instance, (21) and (23). In the cryptographic community, this challenge is known as the search for a fully algebraically homomorphic cryptosystem [91], [21], [92]. While promising initial results have been found [21], it is unclear whether an efficient and generally applicable algebraically homomorphic cryptosystem exists. It is clear, however, that the efficiency of many applied cryptography solutions, including encrypted signal processing, would benefit highly from such a cryptosystem. Compromises exist in the form of "somewhat" homomorphic cryptoschemes, which offer a limited number of multiplication operations, but which are also much more efficient than fully homomorphic schemes [22], [93].

Homomorphic cryptosystems are well suited to typical linear signal processing operations, and we have seen quite a number of examples in this article. Nevertheless, MPC protocols are unavoidable. Generic MPC solutions are often far too expensive, and it is therefore important to optimize the efficiency of MPC protocols using specific domain knowledge. In the signal processing domain, relevant domain knowledge is that encryptions and multiplications nearly always operate on samples that take on a limited number of different values. In many cases these sample values can be represented by 8–16-b integers. This knowledge can be exploited by selecting the DGK cryptosystem [94]–[96], which is tailored to input and output data with only a limited number of possible different values.

We have also seen another way of exploiting the same domain knowledge, specifically to pack multiple 8–16-b values into a single number consisting of n bits, where n is the modulus of the field in which the encrypted operations take place. For instance, if $n = 1,024$ then we can pack around 60 samples into a single value. If done properly, the packing operation reduces the number of homomorphic operations and MPC protocols by the same factor of 60, as was illustrated by the example in Table 7. Data packing is effectively a way to parallelize some of the computationally more expensive operations such as secure multiplication. Parallel processing has been widely studied for computationally intensive signal processing algorithms; that knowledge can now also be used for speeding up encrypted signal processing.

The applications illustrated in this article represent important fundamental signal processing operations such as linear filters and squared error distances. These operations fit generically to homomorphic cryptosystems. Signal processing operations to which the homomorphic property cannot be generically applied—for instance, signal quantization—can be formulated as series of Boolean operations to which the garbled circuits approach can be generically applied. In this tutorial article we have given an overview of carefully designed protocols for privacy-protected signal processing. Whether these tailor-made signal processing protocols are more efficient than the generic garbled circuit approach is subject of future research. A related challenging research question is if there exist fundamental building blocks for privacy-protected signal processing, other than those based on the homomorphic property and garbled circuits. We believe this question is an interesting challenge for the signal processing and cryptographic community alike.

At this point we will also mention that privacy-protected versions of signal processing algorithms have computation and storage demands that very much surpass their plaintext counterparts. Even though the relative complexity increase is huge, the challenge for research in encrypted signal processing is the question if privacy can be reached at all with, in absolute terms, reasonable efforts. We believe that the recent progress in encrypted signal processing presented in this tutorial article shows that these feasible solutions are at the horizon, but that there is also ample room for further research and efficiency improvements.

The final challenge that we should like to mention concerns security models, efficiency and business aspects simultaneously. In cryptography, fundamental security concepts have been formalized through, for instance, indistinguishability of ciphertexts (under various attacks). Encrypted signal processing might need to develop its own security concepts, for instance to distinguish between the privacy of the user data and the privacy of the service provider's algorithm. We believe that such security definitions are a potential area for future research, and that they can be beneficial in further shaping the field of encrypted signal processing.

As we pointed out in the section "Security Models," a malicious adversary model might be too aggressive for many encrypted signal processing applications. Also, if we had to deal with the malicious adversary model, many, if not all, of the solutions described in this article would need to be reconsidered, most likely resulting in computationally more complex protocols involving commitment [15] and zero-knowledge schemes

[16], or alternative forms of garbled circuits [17], [18]. At the same time, the honest-but-curious model is not perfect either. There are two reasons for this. First, users might try to influence the "correctness" of the outcome of the signal processing operations by submitting manipulated input values. For instance, in the privacy-protected collaborative filtering, a user might try to submit extremely high ratings on certain items to influence the collaborative filtering outcome for all other users. In a plaintext version, the recommendation service provider can observe and prevent such behavior. But this is not the case if the recommendation service provider deals only with encrypted values. Since the assumption regarding semihonest behavior does not suffice, and security in the presence of malicious adversaries is excessive and expensive to achieve, alternative security models need to be investigated for privacy-protected signal processing. The covert adversary model [97] attempts to faithfully model the adversarial behavior in commercial, political and social settings. Covert adversaries have the property that they might deviate arbitrarily from the protocol specification in an attempt to cheat, but do not wish to be "caught" doing so. The application of the covert adversary model to privacy-protected signal processing is a new direction that is worth pursuing.

The second reason why the honest-but-curious model is not always suitable is that it puts the service provider in a position that might eventually make the service no longer commercially interesting or even impossible. For instance, in collaborative filtering, the recommendation service provider currently learns the interests of its users, allowing it to develop personalized content advertisement models. Without knowing at least the aggregated interests of users, the recommendation service provider's business model might fail, and users will no longer have access to the recommendation service. We clearly see that there can also be good reasons why service providers are curious. Therefore, where in current practices the user's personal data is completely in the hands of service providers, and in the described encrypted signal processing solutions user's data could potentially be completely hidden from service providers, the challenge will be to find encrypted signal processing solutions that balance the user's and the service provider's interests in a negotiable fashion.

ACKNOWLEDGMENTS

This article is the result of ongoing collaboration between signal processing and cryptography researchers. The authors thank them for their contributions to the furthering of encrypted signal processing, and for the joint papers that have resulted from various collaborations. Particular thanks go to (in alphabetical order) Michael Beye (Delft University of Technology), Tziano Bianchi (University of Florence), Dario Catalano (University of Catania), Martin Franz (Technische Universität Darmstadt), Ton Kalker (DTS, Inc., US), Stefan Katzenbeisser (Technische Universität Darmstadt), Riccardo Lazzeretti (University of Siena), Jorge Guajardo Merchan (Robert Bosch LLC, US), Claudio Orlandi (Bar-llan University), Alessandro Piva (University of Florence), Ahmad-Reza Sadeghi (Technische Universität Darmstadt), Tomas Toft (University of Aarhus), and Thijs Veugen (TNO-ICT, The Netherlands). The authors also wish to thank the anonymous reviewers for their careful reading and detailed suggestions to improve the article.

AUTHORS

R. (Inald) L. Lagendijk (R.L.Lagendijk@TUDelft.nl) received his M.Sc. and Ph.D. degrees in electrical engineering from Delft University of Technology (TU Delft) in 1985 and 1990, respectively. He was a visiting scientist in the Electronic Image Processing Laboratories, Eastman Kodak Research, Rochester, New York, in 1991 and visiting professor at Microsoft Research and Tsinghua University, Beijing, China, in 2000 and 2003, respectively. He was a consultant at Philips Research Eindhoven from 2002 to 2005. Since 1999, he has been a full professor at TU Delft in the field of multimedia signal processing, where he holds the chair position of multimedia signal processing. He is the author of *Iterative Identification and Restoration of Images* (Kluwer, 1991) and coauthor of *Motion Analysis and Image Sequence Processing* (Kluwer, 1993) and *Image and Video Databases: Restoration, Watermarking, and Retrieval* (Elsevier, 2000). He was on the conference organizing committees of the International Conference on Image Processing in 2001, 2003, 2006, and 2011. He has been a member of the IEEE Signal Processing Society's Technical Committee on Image and Multidimensional Signal Processing as well as associate editor of *IEEE Transactions on Image Processing*, *IEEE Transactions on Signal Processing's Supplement on Secure Digital Media*, and *IEEE Transactions on Information Forensics and Security*. He is an elected member of the Royal Netherlands Academy of Arts and Sciences (KNAW). He is a Fellow of the IEEE.

Zekeriya Erkin (Z.Erkin@TUDelft.nl) received his B.Sc. and M.Sc. degrees in computer engineering from Istanbul Technical University in 2002 and 2005, respectively. He received his Ph.D. degree in secure signal processing from TU Delft in 2010. He participated in the European Commission (EC)-funded project Signal Processing in the Encrypted Domain (SPEED). He was a short-term visiting researcher at Aarhus University and the University of California Irvine in 2009 and 2011, respectively. His research interests are watermarking, steganography, and privacy protection in online social networks, trusted health-care systems, and smart metering systems. He is currently a postdoctoral researcher in the Information Security and Privacy Lab, TU Delft.

Mauro Barni (barni@dii.unisi.it) graduated with a degree in electronic engineering at the University of Florence in 1991. He received the Ph.D. degree in informatics and telecommunications in 1995. He has carried out his research activity for over 20 years, first in the Department of Electronics and Telecommunication of the University of Florence, then in the Department of Information Engineering of the University of Siena, where he is an associate professor. He is author/coauthor of approximately 250 papers published in international journals and conference proceedings and holds four patents in the field of digital watermarking and image authentication. He is a coauthor of *Watermarking Systems*

Engineering: Enabling Digital Assets Security and Other Applications (Dekker Inc., 2004). He coordinated the EC-funded project SPEED. He was the founding editor of EURASIP's *Journal on Information Security*. He is an associate editor of *IEEE Transactions on Circuits and Systems for Video Technology* and *IEEE Transactions on Information Forensics and Security*. He was the chair of the IEEE Information Forensic and Security Technical Committee from 2010 to 2011.

REFERENCES

[1] O. Bowcott. (2008, Oct. 20). Interpol Wants Facial Recognition Database to Catch Suspects. [Online]. Available: http://www.guardian.co.uk/world/2008/oct/20/interpol-facial-recognition

[2] T. Grose. (2008, Feb. 11). When surveillance cameras talk. *Time Mag.* [Online]. Available: http://www.time.com/time/world/article/0,8599,1711972,00.html

[3] M. Magnier. (2008, Aug. 07). Many eyes will watch visitors. *Los Angeles Times* [Online]. Available: http://articles.latimes.com/2008/aug/07/world/fg-snoop7

[4] G. Adomavicius and A. Tuzhilin, "Toward the next generation of recommender systems: A survey of the state-of-the-art and possible extensions," *IEEE Trans. Knowledge Data Eng.*, vol. 17, no. 6, pp. 734–749, 2005.

[5] N. Ramakrishnan, B. Keller, B. J. Mirza, A. Y. Grama, and G. Karypis, "Privacy risks in recommender systems," *IEEE Internet Comput.*, vol. 5, no. 6, pp. 54–62, 2001.

[6] W. Lu, A. L. Varna, A. Swaminathan, and M. Wu, "Secure image retrieval through feature protection," in *Proc. IEEE Int. Conf. Acoust. Speech Signal Processing (ICASSP)*, Washington, DC, 2009, pp. 1533–1536.

[7] S. Rane, S. Wei, and A. Vetro, "Privacy-preserving approximation of L1 distance for multimedia applications," in *Proc. IEEE Int. Conf. Multimedia Expo (ICME)*, 2010, pp. 492–497.

[8] J. K. Pillai, V. M. Patel, R. Chellappa, and N. K. Ratha, "Secure and robust IRIS recognition using random projections and sparse representations," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 33, no. 9, pp. 1877–1893, Sept. 2011.

[9] O. Goldreich, *Foundations of Cryptography II*. Cambridge, U.K.: Cambridge Univ. Press, 2004.

[10] P. Comesana, L. Perez-Freire, and F. Perez-Gonzalez, "Blind Newton sensitivity attack," *IEE Proc. Inform. Security*, vol. 153, no. 3, pp. 115–125, 2006.

[11] C. Fontaine and F. Galand, "A survey of homomorphic encryption for nonspecialists," *Eurasip J. Inform. Security*, vol. 10.1155/2007/13801, 2007.

[12] Z. Erkin, A. Piva, S. Katzenbeisser, R. L. Lagendijk, J. Shokrollahi, G. Neven, and M. Barni, "Protection and retrieval of encrypted multimedia content: When cryptography meets signal processing," *Eurasip J. Inform. Security*, vol. 0.1155/2007/78943, 2007.

[13] T. Bianchi, A. Piva, and M. Barni, "Composite signal representation for fast and storage-efficient processing of encrypted signals," *IEEE Trans. Inform. Forensics Sec.*, vol. 5, no. 1, pp. 180–187, 2010.

[14] T. Okamoto and S. Uchiyama, "A new public-key cryptosystem as secure as factoring," in *Proc. EUROCRYPT*, 1998, pp. 308–318.

[15] G. Bassard, D. Chaum, and C. Crepeau, "Minimum disclosure proofs of knowledge," *J. Comput. Syst. Sci.*, vol. 37, no. 2, pp. 156–189, Oct. 1986.

[16] O. Goldreich, S. Micali, and A. Wigderson, "Proofs that yield nothing but their validity for all languages in NP have zero-knowledge proof systems," *J. ACM*, vol. 38, no. 3, pp. 691–729, 1991.

[17] Y. Huang, D. Evans, J. Katz, and L. Malka, "Faster secure two-party computation using garbled circuits," in *Proc. 20th USENIX Sec. Symp.*, San Francisco, Aug. 2011.

[18] J. B. Nielsen, P. S. Nordholt, C. Orlandi, and S. S. Burra, "A new approach to practical active-secure two-party computation," *Cryptology ePrint Archive*, vol. arXiv:1202.3052v1 [cs.CR], 2011.

[19] T. Bianchi, A. Piva, and M. Barni, "Encrypted domain DCT based on homomorphic encryption," *Eurasip J. Inform. Security*, vol. 10.1155/2009/716357, 2009.

[20] T. Bianchi, A. Piva, and M. Barni, "On the implementation of the discrete Fourier transform in the encrypted domain," *IEEE Trans. Inform. Forensics Sec.*, vol. 4, no. 1, pp. 86–97, Mar. 2009.

[21] C. Gentry, "Fully homomorphic encryption using ideal lattices," in *Proc. 41st Annu. ACM Symp. Theory Comput.*, New York, 2009, pp. 169–178.

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

[23] C. Gentry and S. Halevi, "Implementing gentry's fully-homomorphic encryption scheme," Cryptology ePrint Archive Rep. 520, 2010 [Online]. Available: http://eprint.iacr.org

[24] A. C. Yao, "Protocols for secure computations," in *Proc. IEEE Symp. Found. Comput. Sci.*, 1982, pp. 160–164.

[25] R. Cramer, I. Damgård, and J. B. Nielsen, "Multiparty computation from threshold homomorphic encryption," in *Proc. Int. Conf. Theory Appl. Cryptographic Tech. (EUROCRYPT)*, London, U.K., 2001, pp. 280–299.

[26] C. Orlandi, "Is multiparty computation any good in practice?," in *Proc. IEEE Int. Conf. Acous. Speech Signal Processing.* May 2011, pp. 5848–5851.

[27] R. Agrawal and R. Srikant, "Privacy-preserving data mining," *SIGMOD Rec.*, vol. 29, no. 2, pp. 439–450, May 2000.

[28] M. A. Turk and A. P. Pentland, "Eigenfaces for recognition," *J. Cognitive Neurosci.*, vol. 3, no. 1, pp. 71–86, 1991.

[29] M. A. Turk and A. P. Pentland, "Face recognition using eigenfaces," in *Proc. IEEE Comput. Soc. Conf. Comput. Vision Pattern Recognit.*, 1991, pp. 586–591.

[30] Z. Erkin, M. Franz, S. Katzenbeisser, J. Guajardo, R. L. Lagendijk, and T. Toft, "Privacy-preserving face recognition," in *Proc. Symp. Privacy Enhanced Technol.*, Seattle, Aug. 2009, pp. 235–253.

[31] A.-R. Sadeghi, T. Schneider, and I. Wehrenberg, "Efficient privacy-preserving face recognition," in *Proc. Int. Conf. Inform. Security Cryptology*. Dec. 2009, LNCS [Online]. Available: http://eprint.iacr.org/2009/507

[32] M. Osadchy, B. Pinkas, A. Jarrous, and B. Moskovich, "Scifi: A system for secure face identification," in *Proc. IEEE Symp. Security Privacy*, 2010, pp. 239–254.

[33] A. Senior, A. Oankanti, and A. Hampapur, "Enabling video privacy through computer vision," *IEEE Security Privacy Mag.*, vol. 3, no. 3, pp. 50–57, 2005.

[34] F. Dufaux and T. Ebrahimi, "Scrambling for video surveillance with privacy," in *Proc. Conf. Comput. Vision Pattern Recognit. Workshop*, 2006.

[35] T. Boult, "Pico: Privacy through invertible cryptographic obscuration," *Comput. Vision Interactive Intell. Environ.*, 0-7695-2524-5, pp. 27–38, 2005.

[36] E. M. Newton, L. Sweeney, and B. Malin, "Preserving privacy by de-identifying face images," *IEEE Trans. Knowledge Data Eng.*, vol. 17, no. 2, pp. 232–243, 2005.

[37] J. Schiff, M. Meingast, D. Mulligan, S. Sastry, and K. Goldberg, "Respectful cameras: Detecting visual markers in realtime to address privacy concerns," in *Proc. Int. Conf. Intell. Robots Syst.*, 2007, pp. 971–978.

[38] A. K. Jain, M. N. Murty, and P. J. Flynn, "Data clustering: A review," *ACM Comput. Surveys*, vol. 31, no. 3, pp. 264–323, 1999.

[39] Y. Lindell and B. Pinkas, "Privacy preserving data mining," *Advances in Cryptology (CRYPTO 2000)*, pp. 36–54, 2000.

[40] J. Vaidya, C.W. Clifton, and Y. M. Zhu, *Privacy Preserving Data Mining* (Advances in Information Security, vol. 19). New York: Springer-Verlag, 2006.

[41] V. S. Verykios, E. Bertino, I. N. Fovino, L. P. Provenza, Y. Saygin, and Y. Theodoridis, "State-of-the-art in privacy preserving data mining," *SIGMOD Rec.*, vol. 33, no. 1, pp. 50–57, 2004.

[42] J. Vaidya and C. Clifton, "Privacy-preserving K-means clustering over vertically partitioned data," in *Proc. 9th ACM SIGKDD Int. Conf. Knowledge Discovery Data Mining*, New York, 2003, pp. 206–215.

[43] G. Jagannathan and R. N. Wright, "Privacy preserving distributed K-means clustering over arbitrarily partitioned data," in *Proc. 11th ACM SIGKDD Int. Conf. Knowledge Discovery Data Mining*, 2005, pp. 593–599.

[44] G. Jagannathan, K. Pillaipakkamnatt, and R. N. Wright, "A new privacy-preserving distributed K-clustering algorithm," in *Proc. 6th SIAM Int. Conf. Data Mining*, 2006.

[45] P. Bunn and R. Ostrovsky, "Secure two-party K-means clustering," in *Proc. 14th ACM Conf. Comput. Commun. Security*, 2007, pp. 486–497.

[46] B. Goethals, S. Laur, H. Lipmaa, and T. Mielikainen, "On private scalar product computation for privacy-preserving data mining," in *Proc. 7th Int. Conf. Inform. Security Cryptology*, 2004, pp. 2–3.

[47] S. R. M. Oliveira and O. R. Zaiane, "Achieving privacy preservation when sharing data for clustering," *Lecture Notes in Computer Science*, New York: Springer, 2004, pp. 67–82.

[48] Z. Erkin, T. Veugen, T. Toft, and R. L. Lagendijk, "Privacy-preserving user clustering in a social network," in *Proc. 1st IEEE Workshop Inform. Forensics Security*, 2009, pp. 96–100.

[49] M. R. T. Beye, Z. Erkin, and R. L. Lagendijk, "Efficient privacy preserving k-means clustering in a three-party setting," in *Proc. IEEE Workshop Inform. Forensics Security*, 2011, pp. 1–6.

[50] B. Sarwar, G. Karypis, J. Konstan, and J. Reidl, "Item-based collaborative filtering recommendation algorithms," in *Proc. 10th Int. Conf. World Wide Web*, 2001, pp. 285–295.

[51] M. Deshpande and G. Karypis, "Item-based top-N recommendation algorithms," *ACM Trans. Inform Syst.*, vol. 22, no. 1, pp. 143–177, 2004.

[52] H. Polat and W. Du, "SVD-based collaborative filtering with privacy," in *Proc. ACM Symp. Appl. Comput.*, New York, 2005, pp. 791–795.

[53] H. Polat and W. Du, "Privacy-preserving collaborative filtering using randomized perturbation techniques," in *Proc. IEEE Int. Conf. Data Mining*, 2003, pp. 625–628.

[54] F. McSherry and I. Mironov, "Differentially private recommender systems: Building privacy into the net," in *Proc. 15th ACM SIGKDD Int. Conf. Knowledge Discovery Data Mining*, 2009, pp. 627–636.

[55] M. Atallah, M. Bykova, J. Li, K. Frikken, and M. Topkara, "Private collaborative forecasting and benchmarking," in *Proc. ACM Workshop Privacy Electron. Soc.*, 2004, pp. 103–114.

[56] J. F. Canny, "Collaborative filtering with privacy via factor analysis," in *Proc. ACM SIGIR*, 2002, pp. 238–245.

[57] J. F. Canny, "Collaborative filtering with privacy," in *Proc. IEEE Symp. Security Privacy*, 2002, pp. 45–57.

[58] Z. Erkin, M. R. T. Beye, T. Veugen, and R. L. Lagendijk, "Efficiently computing private recommendations," in *Proc. Int. Conf. Acoust. Speech Signal Processing*, 2011, pp. 5864–5867.

[59] Z. Erkin, T. Veugen, T. Toft, and R. L. Lagendijk, "Generating private recommendations efficiently using homomorphic encryption and data packing," *IEEE Trans. Inform. Forensics Security*, vol. 7, no. 3, pp. 1053–1066, 2012.

[60] S. Kamara, P. Mohassel, and M. Raykova, "Outsourcing multi-party computation," Cryptology ePrint Archive Rep. 272, 2011 [Online]. Available: http://eprint.iacr.org

[61] M. Barni, P. Failla, R. Lazzeretti, A. Paus, A.-R. Sadeghi, T. Schneider, and V. Kolesnikov, "Efficient privacy-preserving classification of ECG signals," in *Proc. 1st IEEE Workshop Inform. Forensics Security*, 2009, pp. 91–95.

[62] M. Barni, P. Failla, R. Lazzeretti, A.-R. Sadeghi, and T. Schneider, "Privacy-preserving ECG classification with branching programs and neural networks," *IEEE Trans. Inform. Forensics Security*, vol. 6, no. 2, pp. 452–468, June 2011.

[63] J. R. Troncoso-Pastoriza, S. Katzenbeisser, and M. Celik, "Privacy preserving error resilient DNA searching through oblivious automata," in *Proc. 14th ACM Conf. Comput. Commun. Security*, 2007, pp. 519–528.

[64] R. Wang, X. F. Wang, Z. Li, H. X. Tang, M. K. Reiter, and Z. Dong, "Privacy-preserving genomic computation through program specialization," in *Proc. 16th ACM Conf. Comput. Commun. Security (CCS)*, New York, 2009, pp. 338–347.

[65] Y. Huang, L. Malka, D. Evans, and J. Katz, "Efficient privacy-preserving biometric identification," in *Proc. 18th Network Distributed Syst. Security Conf.*, 2011.

[66] D. Maltoni, D. Maio, A. K. Jain, and S. Prabhakar, *Handbook of Fingerprint Recognition*. London, U.K.: Springer-Verlag, 2009.

[67] M. Barni, T. Bianchi, D. Catalano, M. Di Raimondo, R. Donida Labati, P. Failla, D. Fiore, R. Lazzeretti, V. Piuri, A. Piva, and F. Scotti, "A privacy-compliant fingerprint recognition system based on homomorphic encryption and fingercode templates," in *Proc. IEEE 4th Int. Conf. Biometrics Theory Appl. Syst.*, Washington DC, 2010, pp. 1–7.

[68] A. K. Jain, S. Prabhakar, L. Hong, and S. Pankanti, "Filterbank-based fingerprint matching," *IEEE Trans. Image Processing*, vol. 9, no. 5, pp. 846–859, 2000.

[69] M. Barni, T. Bianchi, D. Catalano, M. Di Raimondo, R. Donida Labati, P. Failla, D. Fiore, R. Lazzeretti, V. Piuri, A. Piva, and F. Scotti, "Privacy-preserving fingercode authentication," in *Proc. ACM Multimedia Security Workshop*, Rome, 2010, pp. 231–240.

[70] J. Daugman, "How iris recognition works," *IEEE Trans. Circuits Syst. Video Technol.*, vol. 4, no. 1, pp. 21–30, 2004.

[71] Y. Luo, S. C. Samson Cheung, T. Pignata, R. Lazzeretti, and M. Barni, "An efficient protocol for private iris-code matching by means of garbled circuits," in *Proc. IEEE Int. Conf. Image Processing*, 2012.

[72] S. Katzenbeisser and F. Petitcolas, *Information Hiding Techniques for Steganography and Digital Watermarking*. Norwood, MA: Artech House, 2000.

[73] G. C. Langelaar, I. Setyawan, and R. L. Lagendijk, "Watermarking digital image and video data: A state-of-the-art overview," *IEEE Signal Processing Mag.*, vol. 17, no. 5, pp. 20–46, 2000.

[74] I. Pitas, "A method for signature casting on digital images," in *Proc. IEEE Int. Conf. Image Processing*, 1996, vol. III, pp. 215–218.

[75] S. Craver, "Zero knowledge watermark detection," in *Proc. Int. Workshop Inform. Hiding*, 1999, pp. 101–116.

[76] A. Adelsbach and A.-R. Sadeghi, "Zero-knowledge watermark detection and proof of ownership.," in *Proc. Int. Workshop Inform. Hiding*, 2001, pp. 273–288.

[77] A. Adelsbach, S. Katzenbeisser, and A.-R. Sadeghi, "Watermark detection with zero-knowledge disclosure," *ACM Multimedia Syst. J.* vol. 9, no. 3, pp. 266–278, 2003.

[78] N. R. Wagner, "Fingerprinting," in *Proc. IEEE Symp. Security Privacy*, 1983, pp. 18.

[79] B. Pfitzmann and M. Waidner, "Anonymous fingerprinting," in *Proc. Eurocrypt*, 1997, vol. 1233, pp. 88–102.

[80] B. Pfitzmann and A.-R. Sadeghi, "Anonymous fingerprinting with direct non-repudiation," in *Proc. Asiacrypt*, 2000, vol. 1976, pp. 401–414.

[81] M. Kuribayashi and H. Tanaka, "Fingerprinting protocol for images based on additive homomorphic property," *IEEE Trans. Image Processing*, vol. 14, no. 12, pp. 2129–2139, 2005.

[82] R. Anderson and S. Fuloria, "On the security economics of electricity metering," in *Proc. 9th Workshop Econ. Inform. Security*, 2010, pp. 1–18.

[83] M. Kohlweiss and G. Danezis, "Differentially private billing with rebates," in *Proc. Inform. Hiding Conf.,* LNCS, 2011.

[84] K. Kursawe, G. Danezis, and M. Kohlweiss, "Privacy-friendly aggregation for the smart-grid," in *Proc. Symp. Privacy Enhanced Technol.*, 2011, pp. 175–191.

[85] A. Rial and G. Danezis, "Privacy-preserving smart metering," in *Proc. 10th Annu. ACM Workshop Privacy Electron. Soc.*, 2011, pp. 49–60.

[86] E. Shi, T.-H. Chan, E. G. Rieffel, R. Chow, and D. Song, "Privacy-preserving aggregation of time-series data," in *Proc. 18th Annu. Netw. Distributed Syst. Security Symp.*, 2011.

[87] Z. Erkin and G. Tsudik, "Private computation of spatial and temporal power consumption with smart meters," in *Proc. Int. Conf. Applied Cryptography Network Security*, 2012, pp. 561–577.

[88] F. D. Garcia and B. Jacobs, "Privacy-friendly energy-metering via homomorphic encryption," in *Proc. 6th Workshop Security Trust Manage.*, 2010, LNCS 6710, pp. 226–238.

[89] G. Acs and C. Castelluccia, "I have a DREAM! (differentially private smart metering)," in *Proc. Inform. Hiding Conf.*, 2011, LNCS, pp. 1–10.

[90] M. J. van den Hoven, "Design for values and values for design," *Inform. Age*, vol. 7, no. 2, pp. 4–7, 2005.

[91] F. Armknecht and A.-R. Sadeghi, "A new approach for algebraically homomorphic encryption," Cryptology ePrint Archive Rep. 422, 2008 [Online]. Available: http://eprint.iacr.org/

[92] (2009). [Online]. Available: http://www.i-programmer.info/news/112-theory/2330-darpa-spends-20-million-on-homomorphic-encryption.html

[93] Z. Brakerski and V. Vaikuntanathan, "Efficient fully homomorphic encryption from (standard) LWE," in *Proc. IEEE Symp. Found. Comput. Sci. (FOCS)*, 2011, pp. 97–106.

[94] I. Damgård, M. Geisler, and M. Krøigård, "Homomorphic encryption and secure comparison," *Int. J. Appl. Cryptography*, vol. 1, no. 1, pp. 22–31, 2008.

[95] I. Damgård, M. Geisler, and M. Krøigård, "Efficient and secure comparison for on-line auctions," in *Proc. Australasian Conf. Inform. Security Privacy*, 2007, LNCS 4586, pp. 416–430.

[96] I. Damgård, M. Geisler, and M. Krøigård, "A correction to 'Efficient and secure comparison for on-line auctions,'" Cryptology ePrint Archive Rep. 321, 2008 [Online]. Available: http://eprint.iacr.org/

[97] Y. Aumann and Y. Lindell, "Security against covert adversaries: Efficient protocols for realistic adversaries," *J. Cryptology*, vol. 23, no. 2, pp. 281–343, 2010.

[98] A. J. Menezes, P. C. van Oorschot, and S. A. Vanstone, *Handbook of Applied Cryptography*. Boca Raton, FL: CRC Press, 1996.

[99] *Data Encryption Standard (DES)*, Number FIPS PUB 46-3, Federal Information Processing Standards Publication, Nat. Inst. of Standards and Technol., Gaithersburg, MD, 1999.

[100] *Advanced Encryption Standard (AES)*, Federal Information Processing Standards Publication, National Institute of Standards and Technology, Gaithersburg, MD, 2001.

[101] R. L. Rivest, A. Shamir, and L. M. Adleman, "A method for obtaining digital signature and public-key cryptosystems," *Commun. Assoc. Comput. Mach.*, vol. 21, no. 2, pp. 120–126, 1978.

[102] T. El Gamal, "A public key cryptosystem and a signature scheme based on discrete logarithms," *IEEE Trans. Inform. Theory*, vol. 31, no. 4, pp. 469–472, 1985.

[103] P. Paillier, "Public-key cryptosystems based on composite degree residuosity classes," in *Proc. Advances Cryptology (EUROCRYPT)*, 1999, LNCS 1592, pp. 223–238.

[104] S. Goldwasser and S. Micali, "Probabilistic encryption and how to play mental poker keeping secret all partial information," in *Proc. ACM Symp. Theory Comput.*, 1982, pp. 365–377.

[105] I. F. Blake and V. Kolesnikov, "Conditional encrypted mapping and comparing encrypted numbers," in *Proc. Financial Cryptography Data Security*, 2006, LNCS 4107, pp. 206–220.

[106] J. A. Garay, B. Schoenmakers, and J. Villegas, "Practical and secure solutions for integer comparison," in *Proc. Public Key Cryptography*, 2007, LNCS 4450, pp. 330–342.

[107] M. Naor, B. Pinkas, and R. Sumner, "Privacy preserving auctions and mechanism design," in *Proc. ACM Conf. Electron. Commerce*, 1999, pp. 129–139.

[108] P. Tuyls, A. H. M. Akkermans, T. A. M. Kevenaar, G. Jan Schrijen, A. M. Bazen, and R. N. J. Veldhuis, "Practical biometric authentication with template protection," in *Proc. Int. Conf. Audio-Video-Based Biometric Person Authentication*, 2005, LNCS 3546, pp. 436–446.

[109] V. Kolesnikov, A.-R. Sadeghi, and T. Schneider, "Improved garbled circuit building blocks and applications to auctions and computing minima," in *Proc. 8th Int. Conf. Cryptology Network Security*, 2009, LNCS, pp. 1–20.

[110] C. Hazay and Y. Lindell, *Efficient Secure Two-Party Protocols* (Information Security and Cryptography). New York: Springer, 2010.

[SP]