# Privacy in Cloud Computing

Erik Boss Aram Verstegen

August 23, 2013

**Abstract**

We summarize our concerns for observed and potential privacy risks associated with cloud computing in a broad sense, subsequently review the legal aspects concerning data protection from a European perspective, followed by a summary of three related and applicable privacy-enhancing technologies uncovered in our literature study, and conclude by reflecting on the state of affairs.

## 1 Introduction

In recent years there has been a lot of positive buzz surrounding the idea of *cloud computing*: a systems design trend that allows applications to run as ubiquitous services by providing abstractions to the lower layers of application architecture. These layers can be distinguished as *Software*, *Platform* and *Infrastructure* and each of these has a different target audience which can buy these facilities "as a service", i.e., Software/Platform/Infrastructure as a Service (SaaS, PaaS, IaaS). Such abstractions bring benefits like implicit redundancy, centralized security measures and, most importantly, reduces capital and operational expenses for customers by the increased efficiency at which each layer can be managed.

That being said, there is also a note of concern with things like centralized information management, giving up physical segregation for logical segregation, near-infinite storage capacity, cheap, ubiquitous computing power and the legal issues surrounding information processing systems whose infrastructure scale transcends national borders.

In cloud computing it's apparent that there is a very strong connection between security and privacy. If any central component of cloud infrastructure were to be compromised it could almost certainly be used to disclose very privacy-sensitive information, given the scale of cloud infrastructure providers and what kind of private information is (or can be) recorded using applications running "in the cloud". In fact, there is no proof or technically feasible way to prove that such a compromise has not occurred already.

Some of these issues culminate in the recently much-publicized case of mass surveillance by the United States National Security Agency and its partners overseas. As more facts on this case are being made public we are better equipped to make informed choices in how we interact with the manifestations of this technology trend. We no longer need speculation on a worst-case estimate alone to build our adversary model, we now have concrete evidence that our data privacy and computational privacy are at risk.

In this article we will go over some of the potential and actual privacy-related problems that we have encountered as part of our study on the technical and legal domains of cloud computing. Starting off, in section 2 we give a fairly broad overview of the privacy-related issues encountered within the cloud computing era. This will lead into an analysis of the current legal situation regarding cloud computing from a local perspective in section 3. In section 4 we will go over the technical details and specifically describe some of the solutions to privacy-related problems in cloud computing by means of Privacy Enhancing Technologies (PETs). Finally we reflect on our findings and give concluding remarks in section 5.

# 2 Privacy problems in cloud computing

While centralization of infrastructure should generally imply better basic physical and network security, it may lead to greater fragility of the networked ecosystem as a whole, whereby attacks have greater impact than in a strictly distributed environment. Consider how acts of observation, manipulation or destruction would be more effective when carried out against the central node in a star-shaped network rather than anywhere in a well-connected network. There exists more redundancy in the more connected network, which helps make the infrastructure more robust against any such disruptions.

## 2.1 Third Parties

It seems to us the main privacy problem with the cloud paradigm is the trend all but forces users to relinquish control of previously self-hosted information to fewer and fewer third parties. The competitive advantage achieved with scale allows a few key players to lead the Cloud market with an overwhelming majority. The IaaS market is dominated by Amazon who have a 35% market share, their closest competitor is IBM with only a 5% share. PaaS seems to be a more competitive field, with the top three players Salesforce, Amazon and Microsoft each taking roughly 15% of the market. [1] SaaS is the most diversified layer in the cloud paradigm, the problems we see here are not in market share but in sheer scale. Large U.S.-based companies like Facebook, Twitter and Google openly construct behavioral profiles of their millions of 'users' on behalf of their 'clients' - and are severely treading on user privacy in the process. Google has announced their own 'Compute Engine' IaaS service only last year, but we feel they may very well have the capacity to oust Amazon from the leading position in the years to come.

## 2.2 Cause for Concern

Having only a few companies in physical possession of so much personal data is cause for concern in light of the aforementioned fragility that large, centralized information systems are susceptible to with regards to observation, manipulation or destruction of data. Of course these companies do not have a monolithic infrastructure behind their facade; but insiders, legally mandated government agents and clandestine intruders may (hypothetically) reach far into their systems and attain full access to any information stored there.

In the light of recent revelations related to U.S. surveillance programmes, this appears to be much less hypothetical than we initially assumed. The expectation

of privacy when dealing with such companies not only lacks a factual foundation to begin with, but now even the notional expectation of privacy has been severely diminished. At the risk of being cynical: the upshot here is that the public's perception of cloud-related privacy matters is more in line with reality now that they are better informed.

### 2.3 Computational Privacy?

We should discern the difference between privacy of *data* and privacy in *computation*. To illustrate: a data file with private information can be kept secret while hosted in 'hostile territory' by means of conventional encryption schemes - provided the general cryptographic caveats like choice of cipher and operation mode, key generation/management/distribution et cetera are properly addressed - but the required encryption operations and the party that applies them must also operate *consistently* for this scheme to be secure.

Data privacy is something we as computer scientists understand well in terms of secrecy, but computational privacy has actually been dauntingly hard to make sense of to us - we see backdoors *all the way down*. Whereas with personal computing the integrity of the system can be verified down to the hardware level - systems can be tested to confirm to the specifications using *reverse engineering* methods on hardware and software and this can be validated externally - the cloud environment only allows the most cursory glance at the inner workings.

The problem is exacerbated by the fact that virtualization technologies abstract systems from the hardware layer, and require systems built upon them to similarly trust this technology during computation. And again there is centralization in these technologies, leading to concerns about (virtually) universal backdoors in all systems built upon a handful of such technologies.

### 2.4 Formalisms and Philosophy

The strength of encryption schemes can be proven in relation to the hardness of a mathematical problem, but we have no analogous formalism at our disposal with which to gauge consistency in who or what performs these algorithms - let alone enforcing *provable* integrity through formal methods in this black box environment.

John Callas quite recently opined on this matter and lucidly identified it to be a deeply philosophical problem - Gödel's second incompleteness theorem, which states a system cannot be proven consistent from within that very system. [2] His response to these problems is then to concede to a system that is simply 'good enough'. We should therefore construct systems that conform to our specifications and employ an *external* verifier system - this is the best we can do and is also the basic premise of the proposed PETs.

## 3 Legal analysis

In this section we will review the relevant legal aspects surrounding cloud computing. We will focus on the European legal framework since it best fits our own perspective, but moreover because appears to be the most thorough legislation with regards to legal privacy guarantees. This perspective is directly contrasted by the American framework, which promotes self-regulation and seems to take a more conservative

stance toward privacy by providing more legal foothold for law enforcement and intelligence agencies to investigate.

In a cloud computing setting, the main legal problems boil down to two issues: accountability of non-EU companies and multi-tenancy of cloud applications. Essentially, the problems emerge from different users belonging to different jurisdictions using a system that is possibly in yet another jurisdiction.

## 3.1 European Law

The Council of Europe has recognized the right to privacy as a human right from its inception, and as such has included an obligation on member states to uphold this right in its European Convention on Human Rights (ECHR) which was signed by *all* of its member states. ECHR Article 8, "Right to respect for private and family life", gives a broad definition of the right to privacy which has been upheld by the European Court of Human Rights in a broad interpretation.

In the same vein the EU Data Protection Directive 95/46/EC not only forbids privacy violations but in fact bring positive obligations on member states to uphold the right to privacy. [3] The default principle of the law is that personal data (that is, any data relating to a natural person) should not be collected or processed *at all*, except when explicit requirements are met. In principle, personal data may only be collected for the explicit purposes that the subjects consent to. The collected data must be kept up-to-date, remain accurate, be available to view and correct for the subject, and only be stored as long as it is actually needed. Processing of collected data *must* have a legitimate purpose; such as to uphold a contract, comply with legal obligations or if this processing is in the express interest of the data subject.

Furthermore the data processors must take measures to safeguard the data against leaking, tampering or data loss, and can be held accountable by the local supervisory authority (such as the Dutch College Bescherming Persoonsgegevens, CBP). There even exists a special category for *sensitive* personal data (to wit: "racial or ethnic origin, political opinions, religious or philosophical beliefs, trade-union membership, and [...] data concerning health or sex life") whose processing shall be prohibited by member states.

So we see there is a strong legal basis to uphold privacy in EU member states. The definition of private data is broad, the directive states clearly that the norm is privacy by default and the articles appear to cover the ideal of data privacy in every conceivable form - only then are additional exemptions added to this core principle.

A key provision in the directive is that personal data may not be transferred to 'third countries', which do not have compatible legislature that uphold the same standard of privacy. This feature is problematic in transnational business relationships between companies in member states and those in such third countries. This is of course directly applicable to e-commerce in general, and the cloud computing paradigm, where dividing lines are more blurred, in particular.

From infrastructure to software, the responsibilities concerning privacy of provider and client change. In an IaaS setting, the provider can (and is legally obligated to) only do so much to provide basic security safeguards. These responsibilities lie primarily in providing availability of the infrastructure as well as providing hypervisor and physical security, although IaaS providers do not consider security features a competitive advantage. The IaaS provider's client who becomes a data controller is the one responsible towards their respective clients for building secure applications

or platforms on top of this infrastructure under European law.

The data controller is ultimately accountable. In the PaaS setting the responsibilities lie somewhere halfway between IaaS and SaaS - the provider needs to provide a secure platform but it is the client's job to provide the necessary safeguards for any application built on top of this platform.

### 3.1.1 Working party

The Article 29 Working Party, an advisory body composed of members of the local supervisory authorities, has published its (draft) opinion on Cloud Computing in which the authors argue that clients of cloud computing providers themselves become data controllers, and cannot simply delegate this responsibility to the provider from which they receive their services. Therefore they recommend that their clients (data controllers) should select a provider which can guarantee compliance with the data protection legislation. They also recommend cloud customers to do an extensive risk analysis before they commit to a cloud provider which exposes them to more (legal or technical) risks than they are prepared for.

IaaS market leader Amazon has committed to this idea by allowing its customers to restrict their cloud storage systems to operate only within European borders and thereby safeguard the data under European law. Microsoft has expressed they are unable to provide any guarantees for this, and opine that any other company could not do so either. The large American communication companies we are concerned about operate as U.S. companies and are not prepared to keep data locked to our region. Therefore our European rights are not transitive to their systems under current legislation.

### 3.1.2 Privacy Online

Complementary to the Data Protection Directive is the E-privacy Directive (Directive on Privacy and Electronic Communications), which does not override but augments the Data Protection Directive, by additionally obliging member states to protect the privacy of both natural and legal persons in electronic communications. The main obligation to providers is to provide adequate security and to inform users of security breaches, but it also covers things like excessive data retention, spam e-mail and the use of tracking cookies without explicit consent. This last provision has spawned controversy as it is deemed to be unenforceable in practice. Some of the most vocal critics are a group of developers who have created an open source solution to implement a system to use cookies that is in accordance with the directive. Since the interpretation of the law has changed over time, their software has become largely irrelevant and this prompted them to start the campaign `nocookielaw.com`.

## 3.2 Safe Harbor in the United States of America

The United States Department of Commerce has worked with the EU to adapt the Data Protection Directive principles into what is called the "Safe Harbor" principles which US companies can opt in to. This essentially allowed US companies to do business with the EU, which has the more stringent data protection legislation. This is merely a commercial incentive, not law. Companies providing safe harbor facilities are expected to be self-regulating in this regard.

### 3.3 Risks of Warrantless Wiretapping

For all the good it does, the Data Protection Directive also provides an exemption from some of the main articles: allowing for member states to implement legislation which restricts the scope of several key obligations put upon them by the directive to safeguard national security and similar aspects of national concern. The directive allows for some key principles (relating to data quality and transparency) to be bypassed in such cases. As such it enables, for instance, the EU Data Retention Directive (2006/24/EC) to be implemented. This provision *could* allow for member state's participation in blanket surveillance programmes as recently uncovered in the United States.

The 2001 "Uniting and Strengthening America by Providing Appropriate Tools Required to Intercept and Obstruct Terrorism" (USA PATRIOT) act is the United States legislation presented to combat terrorist threats, expanding the definition to include domestic terrorism and greatly broadening the legal requirements for intelligence gathering in this context. [4] It allows United States government agencies to demand access to records of any businesses that operate on US soil in the interest of national security.

The PATRIOT act has also greatly expanded the applicability of the *National Security Letter*, which behave like a subpoena (but have less judicial oversight) for transactional records and data pertaining to natural persons in the interest of national security. These letters come under a gag order which means the recipient is not allowed to inform anybody of the inquiry.

The "Foreign Intelligence Surveillance Act" (FISA, 1978) gives provisions for electronic surveillance of foreign powers and their agents, not excluding domestic surveillance. FISA warrants are governed by a highly secretive court. Only the number of issued and denied warrants are made public. The available records show that of the over 30.000 warrants applied for up to 2012, only 11 warrants have ever been denied. This is because in stead of relying on probable cause, the applicant (the U.S. government) need only demonstrate that the subject of the warrant is a foreign power or an agent thereof.

Over the course of 2006 to 2012 this law has been repeatedly amended; to: relieve restrictions on surveillance of overseas entities even if they are U.S. citizens, to generally provide greater legal surveillance and/or 'acquisition' [1] capability, extend surveillance to more types of electronic communications, provide procedures to direct and/or compel providers to provide assistance, granting said providers immunity from lawsuits, to provide the government retroactive immunity for (emergency) warrantless surveillance, extend the period and ease the procedure with which to make the aforementioned warrantless wiretapping legal, and allows the government to legally certify its own surveillance programs. We're not sure if that is everything related to FISA, but it should give an idea of the reach of this shady branch of government.

Earlier this year an anonymous source, allegedly from within the dutch intelligence service AIVD, was interviewed by the newspaper Telegraaf on the subject of the NSA's blanket surveillance. [5] He confided in them the fact that 1) there are more such programs still unpublicised, 2) the gathered information is accessible to Dutch agents as part of their international cooperation, 3) information inside

[1] Actually it seems some legal acrobatics are employed; here 'acquisition' means something different from surveillance, as it relates to communications with at least one party outside their national borders.

companies that are unwilling to cooperate can be subversively obtained by 'activating' embedded agents within these companies, 4) the full rap sheet of any target of surveillance can be made available to the agent via e-mail within the course of several minutes.

Our initial concern for the strength our European laws give was already heightened when we noted the provisions that let member states loosen some of the obligations in light of national security matters. We still have faith in judicial oversight on this side of the Atlantic ocean, most such intrusive AIVD surveillance tasks require a public prosecutor or magistrate to be publicly accountable. From this new point of view it seems like the all the required oversight can be circumvented entirely if the agents simply ask their helpful overseas colleagues. This thread of the story *demands* further investigation.

### 3.4 Looking ahead

It should be noted that, currently, the protections the European legislation provides do not extend beyond its borders. This Data Protection Directive is set to be succeeded by the General Data Protection Regulation, which amends the code to extend its reach beyond the European borders, and applies everywhere information about European citizens is collected. It also mandates that data breaches must be reported to the Data Protection Authority, and stipulates serious fines for failure to comply. Note, however, that this regulation is still under review and thus subject to change.

## 4 Privacy-Enhancing Technologies

Our worries surrounding illegal wiretapping makes the more general, commercial side of our legal considerations seem somewhat moot. What good is a strong system of privacy when it can be circumvented not only by member states of the European Council, but also the ostensibly most powerful intelligence apparatus on the planet could permit itself to dive into our data. The legal protection we enjoy is a nice-to-have, but in light of these concerns we must enforce our privacy by technical means if we take the adversary seriously.

Say, for instance, that we have a situation in which we want to search in a database. Now, if we apply techniques to search using encrypted queries in an encrypted database, we stop processing any personal data that is not linked to the communications protocol (like IP addresses). Such techniques, are in a way a win-win solution. They reduce liability risks for providers and provide privacy for users.

What we need are systems that give guarantees about the privacy of our data even in a setting with a hostile third party. The most obvious, and probably even the only, solution is to make sure that even the cloud provider cannot, by any internal means, access the data. If the cloud provider cannot do so, we also prevent mass privacy violations by parties (ab)using the cloud providers access, whether they be governmental or criminal in nature. In this section we shall attempt to describe a few ways in which to accomplish this to some degree for some applications. In particular we shall discuss searchable encryption (SE), fully homomorphic encryption (FHE) and functional encryption (FE). The first we discuss as a somewhat older, but still very applicable system. The second we discuss on its own merits but also as a

building block for the third PET, i.e., functional encryption. Note that we shall not attempt to describe any given concept with the greatest level of (mathematical) detail, instead we shall focus on supplying the definitions and the intuition required for understanding how we can use these technologies to aid in our endeavour to make cloud computing more privacy-friendly. We will, however, be the most extensive on the subject of functional encryption since there have been some nice developments in that area over the past half year or so.

Before we discuss the aforementioned technologies it is import to note that the application of cryptography to provide privacy may not be generally feasible in a cloud setting. Van Dijk and Juels [6] formally defined a class hierarchy for cloud applications and went on to prove that for two out of three, even powerful primitives such as fully homomorphic encryption, are not enough to provide privacy. They considered three classes of cloud applications, namely

1. Private single-client computing, computation on data of a client such that only this client can learn the output;
2. Private multi-client computing, computation on data of multiple clients such that the output may be selectively learned only by certain clients using some form of access control;
3. Stateful multi-client computing, similar to private multi-client computing except that the access control policy depends on previous application execution.

Note that this last class actually contains a great many typical cloud applications, e.g., social networks. Therefore the result that neither the second nor the third class actually realizes privacy in the general sense is quite important; some form of trusted state or execution is always required in those two classes. We do feel, however, that even in the face of such a result the study of the techniques in the next few sections is worthwhile and useful because they can provide privacy in specific applications and use cases. In a sense, the result is also quite reassuring, i.e., as long as data need not be shared among clients it is indeed very possible to get privacy using more general methods like FHE.

## 4.1 Searchable Encryption

One application of particular interest in a cloud computing setting is that of searching through data. It is not hard to imagine that, in a world where we store more and more data in the cloud, we would like good and secure ways to search through said data. However, we may not trust the cloud provider with knowing what we are searching for nor what sort of data is stored. Enter searchable encryption, a scheme where both the stored data and the queries are kept secret. In this section, we shall first describe a fairly simple system of searchable encryption as devised by Boneh [7] called Public-Key Encryption with Keyword Search (PEKS).

Boneh describes the system in specific, but easily generalizable, setting, i.e., the routing of encrypted e-mail messages depending on the occurrence of certain keywords in this message. The goal is to test whether a keyword occurs in a message without learning any other information about the message. Note that such a goal can conceivably be achieved using the functional encryption scheme that we shall later discuss but we do not necessarily need the full power of such a scheme for this relatively simple goal. This scenario, when storing the encrypted mail on an IMAP server, is analogous to a cloud provider storing such messages.

The scheme is, conceptually at least, not very complicated. Let us say $B$ wants to send $A$ an encrypted message using the PEKS scheme. The message would look like this:

$$\left[E_{A_{pub}}(m), \text{PEKS}(A_{pub}, W_1), \ldots, \text{PEKS}(A_{pub}, W_k)\right]$$

where $A_{pub}$ is $A$'s public key, $m$ is the message $E_k(m)$ is the encryption of $m$ under $k$, PEKS is the algorithm that provides searchable encryption and $W_1, \ldots, W_k$ are the keywords associated with $m$.

For the scheme to work we need the following four algorithms:

- KeyGen($s$): Generates public/private keypair $(A_{pub}, A_{priv})$ given security parameter $s$;
- PEKS($A_{pub}, W$): Given the public key and a word $W$, provide a searchable encryption of $W$;
- Trapdoor($A_{priv}, W$): Produce a trapdoor for $W$ given the private key and a word $W$;
- Test($A_{pub}, S, T_W$): Given the public key, a searchable encryption $S = \text{PEKS}(A_{pub}, W')$ and the trapdoor $T = \text{Trapdoor}(A_{priv}, W)$, test whether $W = W'$ and output 'yes' or 'no' accordingly.

A query would then consist of sending a trapdoor of a message to the server and this server could then, given the existence of the Test algorithm, return all messages containing this keyword. Boneh notes that this scheme, as is, is semantically secure against an adaptive chosen keyword attack but not against a chosen ciphertext attack. It can, however, be adapted to provide such a guarantee.

It turns out that we can construct these four algorithms using any trapdoor permutation (other constructions are given, but we shall forego those). We need two guarantees: a polynomially bounded number of keywords and an encryption system that is source-indistinguishable, i.e., it is computationally hard to determine with which key a given ciphertext was encrypted with. In order to now construct the PEKS system we need to define the four algorithms in terms suitable to this particular construction, i.e., with trapdoor permutations and source-indistinguishability. The algorithm can be constructed as follows, where $\Sigma$ is the family of keywords, polynomial in size:

- KeyGen: Generate a public/private keypair $PK_W/Priv_W$ for each $W \in \Sigma$. The resulting public key $A_{pub}$ is $A_{pub} = \{PK_W | W \in Sigma\}$;
- PEKS($A_{pub}, W$): Output PEKS($A_{pub}, W$) = $(M, E_{PK_W}(M))$ for a randomly picked $M = 0, 1^s$;
- Trapdoor($A_{priv}, W$): Simply output $T_W = Priv_W$;
- Test($A_{pub}, S, T_W$): Test whether the decryption $D_{T_W}(S) = 0^s$ and output 'yes' or 'no' accordingly.

While this construction is very simple and generic in its application, we can do much better in terms of efficiency. The fact that (seemingly) we require a very, very large public key is a detriment to the applicability of this scheme. However, if we use a more specific construction (or a more optimized version of the above) we can avoid this problem. In fact, in the very same paper Boneh explains a construction based on hash functions and bilinear maps that is much more efficient.

Concerning searchable encryption, we can definitely see applications for such a system. Given that it should be more efficient than, say, FHE we can probably use

such a system for doing keyword searches in cloud-stored documents. Note that this is a particular version of searchable encryption; for other situations we may want more specialized schemes, especially if we want to do complicated queries. Also, a scheme such as this exposes the access pattern of the user requesting the search, i.e., the provider may not learn the content of the messages but it can learn which messages are frequently accessed and so forth. A solution to this problem is given by Boneh in [8]. In fact, Li [9] argues that the fact that this scheme only provides exact-keyword search makes it unsuitable for cloud computing. They go on to propose a system for doing fuzzy searches. This is, although quite a bit more fancy, necessarily more complex as well.

## 4.2 Fully Homomorphic Encryption

We feel comfortable saying that fully homomorphic encryption (FHE) is one of the more interesting and promising advances in cryptography of the last decade. It allows for performing complex dynamically-chosen computations on data while said data remains encrypted. In the context of cloud computing, FHE is especially interesting. Take, for instance, one of the services a cloud provider might provide, namely the outsourcing of computational power. Using the ability to compute arbitrary functions over encrypted data allows one to use such external computation power without exposing what is being computed (and which data is used). It is important to note that at the time of writing efficient ways to implement an FHE scheme do not yet exist in any practical sense. However, progress is being made in this area.

The first properly working construction of FHE is attributed to Gentry [10], who also construed a blueprint of sorts for building a FHE system. A full, precise explanation of FHE as per Gentry's construction, or any of the later ones is truly outside the scope of this particular paper. Instead, we refer the reader to [11] for a more complete, but not too in-depth, overview of FHE and to [12] for a fairly recent construction of FHE that is not directly based on Gentry's original blueprint but on the (Ring) Learning with Errors problem. What we shall do is to provide a more general overview of what FHE entails and what its properties generally are. For the sake of consistency with the next section we shall use the definitions and notation from [13], which are in turn adapted from [11].

We can define FHE scheme as a quadruple of polynomial time algorithms

$$(\text{FHE.KeyGen}, \text{FHE.Enc}, \text{FHE.Dec}, \text{FHE.Eval})$$

They are defined as follows:

1. $\text{FHE.KeyGen}(1^k)$: A probabilistic algorithm that given security parameter $k$, outputs public/private keypair $pk/sk$;
2. $\text{FHE.Enc}(pk, x \in \{0,1\})$: A probabilistic algorithm that given the public key and an input bit outputs ciphertext $\psi$;
3. $\text{FHE.Dec}(sk, \psi)$: A deterministic algorithm that given the secret key and a ciphertext outputs the message $x^* \in \{0,1\}$;
4. $\text{FHE.Eval}(pk, C, \psi_1, \psi_2, \ldots, \psi_n)$: A deterministic algorithm that given the public key, a circuit taking $n$ bits as input and $n$ ciphertexts, outputs a ciphertext $\psi_C$.

A construction of such a scheme is *homomorphic* if given $\psi = \text{FHE.Eval}(pk, C, \psi_1, \ldots, \psi_n)$ the chance that $\text{FHE.Dec}(sk, \psi) \neq C(\psi_1, \ldots, \psi_n)$ is negligible. It is *compact* if the output size of FHE.Eval is at most polynomial in the security parameter $k$. The construction is fully homomorphic if for all arithmetic circuits $C$ over $GF(2)$ the scheme is both compact and homomorphic.

The construction in the next section will use a *leveled* FHE scheme. Such a scheme is defined as an FHE scheme where FHE.KeyGen takes a second parameters $1^d$ and the scheme is fully homomorphic for all $d$-depth arithmetic circuits over $GF(2)$. We suppose, as per [11], that other types of circuits than arithmetic ones are possible, but for the sake of ease we shall not consider those.

## 4.3 Functional Encryption

In this section we shall discuss a method of constructing a functional encryption (FE) scheme due to Goldwasser [13]. The scheme is based on fully homomorphic encryption (FHE) and a particular type of attribute-based encryption (ABE). Functional encryption allows for the computation of a function $f$ on encrypted data without learning anything about the input data or the function itself, it only learns the output. In theory this means that you can give your cloud provider (or some other party) your encrypted data and certain keys that allow for the computation of certain functions. To reuse Goldwasser's example, this allows us to, say, compute a spam-detection function on encrypted mail that learns nothing about the contents of said mail except whether it was classified as spam. This spam-detection function can therefore be run by a third party without risk of compromising the contents of the message. This particular FE scheme appears to have a great many applications, key among which are reusable garbled circuits and token-based program obfuscation as well as publicly verifiable computation delegation. It also manages to solve one of the problems of FHE, which is the fact that anyone with a public key can compute on the encrypted data and get an encrypted result. This is, in a sense, both an important feature and an important bug. Sometimes we want, like in the spam filter example, to give a third party the ability to decrypt the result. For FHE this would entail giving up the entire private key, which may not be what one wants. FE solves this by giving a key for a specific purpose (or function) to this service, thereby opening up new possibilities. This, in theory, allows for complex data access policies determined by the client of a cloud service. Note again, however, that for now the reliance on FHE of this scheme makes for a very inefficient system and it is, as of yet, not feasible to implement.

As stated, we have need of three ingredients to make this particular functional encryption scheme work: FHE, a specific variant of ABE and a traditional Yao garbling scheme. In particular we shall use the leveled FHE scheme, which we already defined, and a something the authors call two-outcome ABE. The garbled circuit will be explained after the explanation for the ABE.

The difference with a normal ABE and a two-outcome ABE is simple, where we normally encrypt only one message with an attribute and a key, now we encrypt two messages at once. The decryption than outputs the first message if the attribute evaluates to 0 and the second if it evaluates to 1. We shall see a more formal definition shortly. In general terms, however, ABE is a scheme that manages the encoding of an access policy into the encryption and decryption.

Given a class of predicates $\mathcal{P}$ defined by circuits with $n$-bit input and a message

space $M$ we can define the two-outcome ABE using four algorithms:

1. ABE. Setup($1^k$): Given the security parameter, output public/private master keys $fmpk/fmsk$;
2. ABE. KeyGen($fmsk, P$): Given the master secret key and a predicate $P \in \mathcal{P}$, output a predicate key $fsk_P$ pertaining to $P$;
3. ABE. Enc($fmpk, x, M_0, M_1$): Given the master public key, an attribute and two messages, output the ciphertext $c$;
4. ABE. Dec($fsk_P, c$): Given the predicate key and the ciphertext, output $M^* \in M$.

Such a scheme is correct if it indeed outputs ABE. Dec($fsk_P, c$) = $M_1$ if $P(x) = 1$ and $M_0$ otherwise.

Now that we have both ABE and FHE we only need to define the Yao garbling scheme. Intuitively, suppose we want to compute $C(x)$, where $C$ is the circuit and $x$ the input and we do not wish for information to leak about $C$ or $x$. We now encode $C$ using a one-way function in a garbled circuit $\Gamma$ that still computes the same underlying function. That is, it does this if we also encode the input to the circuit $x$ as $c$ such that $\Gamma(c) = C(x)$. Now we can give this circuit to some third party with some input and they can compute the outcome without learning more than black-box knowledge about the circuit and the input.

More precisely, such a garbling scheme consists of three algorithms for garbling a given family of circuits $\mathcal{C} = \{\mathcal{C}_n\}_{n\in\mathbb{N}}$ where $\mathcal{C}_n$ is a set of boolean circuits taking $n$-bit inputs. The algorithms are as follows:

1. Gb. Garble($1^k, C$): Given the security parameter and a circuit taking $n$-bit inputs, output the secret key $sk$ and the garbled circuit $\Gamma$;
2. Gb. Enc($sk, x$): Given the secret key and some input $x \in 0, 1^n$ to the original circuit, output an encoding $c$ of $x$ whose size notably does not rely on the size of $C$;
3. Gb. Eval($\Gamma, c$): Given the garbled circuit and an encoded input, output $\Gamma(c)$ for which holds that $\Gamma(c) = C(x)$.

In the standard Yao construction the secret consists of labels, $sk = \{L_i^0, L_i^1\}_{i=1}^n$ and the encoding of $x$ looks like $c = (L_1^{x_1}, \ldots, L_n^{x_n})$.

Rejoice, we now have all necessary components to create the functional encryption scheme and all the goodness that comes from it. Recall that in functional encryption we wish to compute $f(x)$ without exposing $f$ or $x$. First of all, we define what a functional encryption scheme looks like. Consider the class of function $\mathcal{F} = \{\mathcal{F}_n\}_{n\ in\mathbb{N}}$ as boolean functions with $n$-bit input. Given that, an FE scheme consists of the following algorithms:

1. FE. Setup($1^k$): Given the security parameter, output public/private master keypair $fmpk/fmsk$;
2. FE. KeyGen($fmsk, f$): Given the master secret key and a function $f \in \mathcal{F}$ with $n$-bit input, output function key $fsk_f$;
3. FE. Enc($fmpk, x$): Given the master public key and an input $x \in \{0, 1\}^n$, output ciphertext $c$;
4. FE. Dec($fsk_f, c$): Given the function key and a ciphertext, output a value $y$.

The scheme is correct if, given $c = \text{FE.Enc}(fmpk, x)$, it holds that $\text{FE.Dec}(fsk_f, c) = y = f(x)$ with high probability.

Given the definition, we shall conclude this by also giving the actual construction; which, in fact, is the reason we had to do all these definitions. The construction entails describing these last four algorithms and we shall go through them in order. The FE scheme works by outputting one bit at a time, repeating the algorithm allows for more than one bit of output. Let $\lambda = \lambda k$, the ciphertext length of the FHE scheme.

**Setup** $\text{FE.Setup}(1^k)$:

$$(fmpk_i, fmsk_i) \leftarrow \text{ABE.Setup}(1^k), \text{ for } i \in [\lambda]$$

Output as public/private master keypair $MPK = (fmpk_1, \ldots, fmpk_\lambda)$ and $MSK = (fmsk_1, \ldots, fmsk_\lambda)$.

**Key generation** $\text{FE.KeyGen}(MSK, f)$:
Let $hpk$ be a FHE public key and $\psi_1, \ldots, psi_n$ the corresponding ciphertexts and $\text{FHE.Eval}_f^i(hpk, \psi_1, \ldots, \psi_n)$ is the $i$-th bit of the evaluation of $f$ on $\psi_1, \ldots, \psi_n$.

1. $$fsk_i \leftarrow \text{ABE.KeyGen}(fmsk_i, \text{FHE.Eval}_f^i \text{ for } i \in [\lambda]$$
   Run the ABE key generation for the functions $\text{FHE.Eval}_f^i$ in order to get the secret keys, using different master keys all the time.
2. Output $fsk_f = (fsk_1, \ldots, fsk_\lambda)$.

**Encryption** $\text{FE.Enc}(MPK, x)$:

1. First generate a fresh FHE keypair $(hpk, hsk) \leftarrow \text{FHE.KeyGen}(1^k)$. Then encrypt each bit of $x$: $\psi_i = \text{FHE.Enc}(hpk, x_i)$ and denote all these encryptions as $\psi$.
2. $$\left(\Gamma, \{L_i^0, L_i^1\}_{i=1}^\lambda\right) \leftarrow \text{Gb.Garble}(1^k, \text{FHE.Dec}(hsk, \cdot))$$
   Generate the garbled circuit for the FHE decryption algorithm using $hsk$ as secret key.
3. $$c_i \leftarrow \text{ABE.Enc}(fmpk_i, (hpk, \psi), L_i^0, L_i^1) \text{ for } i \in [\lambda]$$
   Generate ciphertexts by ABE encryption using results of step 1 and 2.
4. Output $c = (c_1, \ldots, c_\lambda, \Gamma)$ as the ciphertext.

**Decryption** $\text{FE.Dec}(fsk_f, c)$:

1. $$L_i^{d_i} \leftarrow \text{ABE.Dec}(fsk_i, c_i) \text{ for } i \in [\lambda]$$
   Recover the labels for the garbled circuit using the ABE decryption. In particular, note that $d_i = \text{FHE.Eval}_f^i(hpk, \psi)$.

2.

$$\text{Gb. Eval}(\Gamma, L_1^{d_1}, \ldots, L_\lambda^{d_\lambda}) = \text{FHE. Dec}(hsk, d_1 d_2 \ldots d_\lambda) = f(x)$$

Given the garbled circuit and the necessary labels, run the garbled evaluation function in order to extract $f(x)$, which was our goal all along.

That is all there is to Goldwasser's construction of a FE scheme using garbled circuits, FHE and ABE. This by itself is, for all the reasons outlined in the beginning of this section, a wonderful achievement. However, he goes to use this FE scheme to construct reusable garbled circuits, something that was not possible until now. We should note that these reusable garbled circuits do require the use of a symmetric secret key since the encodings need to be encrypted under said key. This is both a curse and a blessing really. On the one hand, this is not an issue in normal single-use garbled circuits. On the other hand, it allows for what Goldwasser denotes as token-based obfuscation. Consider the applicability of given out an obfuscated program with a certain number of tokens representing the input we allow computation on.

# 5 Conclusion

Cloud computing is still becoming increasingly more popular and thus making sure we "do it right" becomes more important as well. In this paper we have attempted to provide an overview of the problems, both legal and technical, surrounding privacy and cloud computing. Furthermore, we have described three privacy-enhancing technologies, searchable encryption, fully homomorphic encryption and functional encryption. Searchable encryption was discussed as a way to solve a certain problem, in particular we discussed the PEKS system for keyword search under encryption. Fully homomorphic encryption we treated both on its own merits and as a building block for the functional encryption scheme described in the last section. It remains unfortunate that all of these solutions still remain too inefficient for real-life applications. Further research will have show whether this can be remedied or whether we have to wait for Moore's Law to play catch-up. That should, in any case, alleviate the pain of the performance issues a little.

On a more personal note, we feel that the legal framework in Europe, especially considering the new and improved data protection regulation, is among the best in the world. We hope that sanity prevails in the face of over 2000 amendments to said regulation.

The apparent ease with which the intelligence community can sidestep the legal framework for privacy protection in the digital realm is deeply troubling to us. We hope, but do not fully expect, this issue to spark a public investigation and eventually gain the required transparency and judicial oversight.

In conclusion, we would like to emphasize through repetition that doing cloud computing privacy right is very, very important lest we succumb to the commonly held belief that 'privacy is dead'. **It does not have to be**.

# References

[1] TeleGeography. Amazons Cloud Iaas and PaaS investments pay off. `http://www.telegeography.com/products/commsupdate/articles/2013/03/11/amazons-cloud-iaas-and-paas-investments-pay-off/`, 2013.

[2] Cryptome. Jon Callas on What Snowden Is Telling Us. `http://cryptome.org/2013/08/callas-snowden.htm`, 2013. Discussion thread between Jon Callas and Zooko Wilcox-O'Hearn on verifiable software.

[3] EU Directive. 95/46/EC of the European Parliament and of the Council of 24 October 1995 on the protection of individuals with regard to the processing of personal data and on the free movement of such data. *Official Journal of the EC*, 23:6, 1995.

[4] Stanley Mailman. *Uniting and Strengthening America by Providing Appropriate Tools Required to Intercept and Obstruct Terrorism (USA PATRIOT ACT) Act of 2001: An Analysis*. LexisNexis, 2002.

[5] Bart Olmer. Ook AIVD bespiedt internetter. `http://www.telegraaf.nl/digitaal/21638965/__Ook_AIVD_bespiedt_online__.html`, 2013.

[6] Marten Van Dijk and Ari Juels. On the impossibility of cryptography alone for privacy-preserving cloud computing. In *Proceedings of the 5th USENIX conference on Hot topics in security*, pages 1–8. USENIX Association, 2010.

[7] Dan Boneh, Giovanni Di Crescenzo, Rafail Ostrovsky, and Giuseppe Persiano. Public key encryption with keyword search. In *Advances in Cryptology-Eurocrypt 2004*, pages 506–522. Springer, 2004.

[8] Dan Boneh, Eyal Kushilevitz, Rafail Ostrovsky, and William E Skeith III. Public key encryption that allows pir queries. In *Advances in Cryptology-CRYPTO 2007*, pages 50–67. Springer, 2007.

[9] Jin Li, Qian Wang, Cong Wang, Ning Cao, Kui Ren, and Wenjing Lou. Fuzzy keyword search over encrypted data in cloud computing. In *INFOCOM, 2010 Proceedings IEEE*, pages 1–5. IEEE, 2010.

[10] Craig Gentry. *A fully homomorphic encryption scheme*. PhD thesis, Stanford University, 2009.

[11] Vinod Vaikuntanathan. Computing blindfolded: New developments in fully homomorphic encryption. In *Foundations of Computer Science (FOCS), 2011 IEEE 52nd Annual Symposium on*, pages 5–16. IEEE, 2011.

[12] Zvika Brakerski, Craig Gentry, and Vinod Vaikuntanathan. (leveled) fully homomorphic encryption without bootstrapping. In *Proceedings of the 3rd Innovations in Theoretical Computer Science Conference*, pages 309–325. ACM, 2012.

[13] Shafi Goldwasser, Yael Tauman Kalai, Raluca A Popa, Vinod Vaikuntanathan, and Nickolai Zeldovich. Reusable garbled circuits and succinct functional encryption. In *STOC*, pages 555–564, 2013.