Secure Cloud Computing

Wolfgang A. Halang[1], Maytiyanin Komkhao[2], and Sunantha Sodsee[3]

[1] Chair of Computer Engineering, Fernuniversität in Hagen, Germany
wolfgang.halang@fernuni-hagen.de
[2] Faculty of Science and Technology,
Rajamangala University of Technology Phra Nakhon, Bangkok, Thailand
maytiyanin.k@rmutp.ac.th
[3] Faculty of Information Technology,
King Mongkut's University of Technology North Bangkok, Thailand
sunanthas@kmutnb.ac.th

Abstract. The security risks of cloud computing include loss of control over data and programs stored in the cloud, spying out these data and unnoticed changing of user software by the cloud provider, malware intrusion into the server, eavesdropping during data transmission as well as sabotage by attackers able to fake authorised users. It will be shown here how these security risks can effectively be coped with. Only for preventing the cloud provider from wrong-doing no technical solution is available. The intrusion of malware into cloud servers and its malicious effects can be rendered impossible by hardware-supported architectural features. Eavesdropping and gaining unauthorised access to clouds can be prevented by information-theoretically secure data encryption with one-time keys. A cryptosystem is presented, which does not only work with one-time keys, but allows any plaintext to be encrypted by a randomly selected element out of a large set of possible ciphertexts. By obliterating the boundaries between data items encrypted together, this system removes another toehold for cryptanalysis.

Keywords: Cloud computing, malware prevention, hardware-based security, security by design, eavesdropping, unbreakable encryption.

1 Introduction

Cloud computing means providing services such as data storage, data processing and file repository by a central server to remote users via the Internet. Since already in the 1960ies mainframe computers with so-called time-sharing operating systems were connected via telecommunication lines to user terminals, cloud computing is half a century older than its name and the present hype about it. At that time, the security risks associated with this kind of centralised computing were not so severe, however, as they are today. These risks include loss of control over data and programs stored in the cloud, spying out these data and unnoticed changing of user software by the cloud provider, malware intrusion into the server, eavesdropping during data transmission as well as sabotage by attackers able to fake authorised users.

S. Boonkrong et al. (eds.), *Recent Advances in Information and Communication Technology,* Advances in Intelligent Systems and Computing 265,
DOI: 10.1007/978-3-319-06538-0_30, © Springer International Publishing Switzerland 2014

In this paper, it will be shown how these security risks can effectively be coped with. Only for preventing the cloud provider from wrong-doing no technical solution is available. The intrusion of malware and its malicious effects are made impossible by hardware-supported architectural features. Eavesdropping and gaining unauthorised access to a cloud is prevented by information-theoretically secure data encryption with one-time keys.

The technical security measures outlined herein are all not new, but there are consistently disregarded for decades by mainstream academic computer science as well as by the IT industry. The reasons for this disregard are up to speculation. Maybe, it is just ignorance or the inability to take notice of published results. But for academic circles it could also be the unwillingness to recognise certain problems as solved preventing further research with its possibility to publish more papers. The branch of the IT industry supplying malware detection software and related tools would simply lose its complete business. The suppliers of computers want to sell new computers before the old ones cease functioning, and the customers of operating systems, which as software never wear out, need to be convinced by errors and new features to buy new operating system versions. Hence, of each generation of new hardware and operating systems not more than only slight improvements with respect to security can be expected to allow for many more generations to be marketed. And, finally, we should not forget those who are interested in spying out personal data such as secret services, governments and big corporations.

2 Spying and Sabotage by the Cloud Provider

If one uses a cloud as a data depository, only, and encrypts the data in an unbreakable way as, for instance, outlined in Section 4, then the cloud provider can neither spy out these data nor modify them without being noticed. The provider could just destroy the data which, however, does not make any sense.

There is on-going research, e.g. [1], on the possibility to enable processing of encrypted data without the need to decrypt them first. To carry out a dyadic operation f on two arguments x and y encrypted with the function e this requires, for instance, the existence of an operator g, such that

$$g(e(x), e(y)) = e(f(x, y))$$

holds. This constraint is extremely restrictive. For most combinations of e and f no g will exist. If for a given f a g could exist, then the encryptions allowing this will most likely be rather weak and, hence, easily breakable. Furthermore, to be able to apply an operator g, it must always be known where in a ciphertext the encryptions of two arguments can be found. This offers another point of departure for cryptanalytic attack. In Section 4 it will be shown that the locations of encrypted values should and can be blurred as well.

As a result, one has no other means than to trust the cloud provider if one wants to process data in a cloud. This should not be a problem if the cloud is

operated by the own organisation. A trustable cloud provider could also be a co-operative society in which one is a member. An example for this is DATEV e.G. (www.datev.de) in Nuremberg, which was founded in 1966 by tax accountants, lawyers, chartered accountants and their clients and which has presently some 40,000 members.

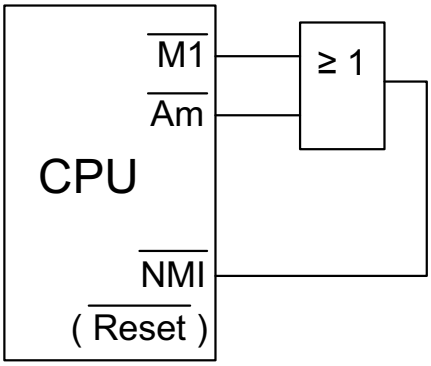
3 Hardware-Supported Malware Prevention

The easiest approach to prevent malware intrusion into cloud servers is just to run the server software on classical mainframe computers, because up to now malware and hackers have never succeeded in entering such a machine [13]. Thus, security updates have almost never been required for their operating systems and their most important subsystems.

An analysis of the various intruders, particularly in form of programs and executable Internet content with malicious intentions, which are making their way into differently designed computers, reveals that they are based on some common principles of operation. If these operation principles are thwarted by appropriate measures, malware is prevented from spreading and from launching its destructive effects. There is a single measure, which already makes any form of malware intrusion impossible, viz. separation of program and data memory. The now for two thirds of a century predominant and thoughtlessly perpetuated Von Neumann architecture with its minimalistic principles is totally inadequate for systems that need to be secure, as it does not separate data from instructions and, thus, does not permit to protect both kinds of information properly. The Harvard architecture, on the other hand, provides this separation throughout and, therefore, represents an adequate construction principle. It is interesting to note that the Harvard architecture is even older than Von Neumann's, and that it actually dates back to Konrad Zuse's first working computer Z1 of 1936. In [2] it was shown that the Harvard architecture can be emulated on Von Neumann computers. It is remarkable to note that this emulation could be implemented with just a single logic gate with two inputs, as shown in Fig. 1.

The constructive security measures discussed in the sequel are less restrictive than the Harvard architecture. Nevertheless, they disable the operation principles of all known malevolent programs in effective ways. In devising them, great importance was attached to the presented solutions being simple and easy to duplicate, in order to be understood and applied without any problems by the users of computers, as unnecessary complexity is the enemy of any effort towards enhancing security.

Software with malicious intentions often interferes with application programs or even with operating system routines in order to manipulate them for its destructive purpose or to deactivate software-implemented security functions. Here a memory segmentation measure as developed in [3] takes effect. It reliably prevents unauthorised accesses to the storage areas of operating system and application programs. To this end, a hardware-supervised segmentation of memory is employed, which protects program code against modifications not permitted by a hardware-implemented write-protection.



Let the main storage be divided into the lower half for program storage and the upper half for data storage. If the processor tries to fetch an instruction (indicated by its output line $\overline{M1}$) from data storage (indicated by an address with the most significant address line $\overline{Am}$ active), then the circuitry shown causes a non-maskable interrupt or a processor reset.

Fig. 1. Emulating the Harvard architecture on a Von Neumann computer

In contrast to programs, data are subject to frequent modifications. Therefore, a hardware-implemented write-protection as above is not feasible for handling reasons. Data can be protected against programs for spying out and modification, however, by a context-sensitive memory allocation scheme [4]. Applying this measure, any unauthorised access to data is precluded. To this end, a system's mass storage, in particular the data area, is further subdivided by a partitioning into context-dependent segments. In an installation mode it is precisely specified which accesses to these segments are permitted to the programs. This is oriented at the data to be protected and not the programs, i.e. in general to each program there exist several data segments separated from one another. In other words, this method permits memory references to any application program and operating system service only by means of using access functions write-protected by hardware, which release the storage areas required for the respective application case for writing and reading or just for reading accesses.

In order not to endanger the advantages of memory areas write-protected by hardware measures during the installation phases of programs, it is necessary to accommodate service programs and their databases also in areas write-protected by hardware and separated from the program area. During installation phases, a hardware device according to [5] constructively excludes attackers from gaining administrator rights by authenticating an authorised person by biometrical or other means.

Destructive programs and software-based aggression from the Internet often use components of digital systems, which they would not need for their feigned nominal functions. Here the hardware-supported security measure detailed in [6] takes effect. It lets any program, any interpretable file and any executable Internet content first disclose which resources are required for execution. The disclosure of a program's nominal functions enables to install boundary values for systems and to supervise their operation. By this and, at any instant, by locking all resources not needed at that time by means of hardware, it is warranted that the desired nominal functionality is observed.

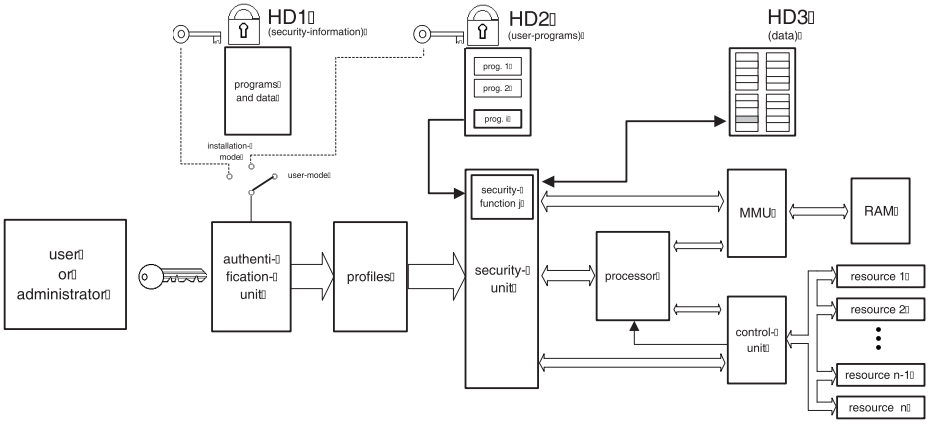


Fig. 2. Hardware-supported security measures

Utilising the measures outlined above, computers are effectively protected against inadmissible accesses, and become immune to intruders and espionage. This holds in particular for still unknown attack patterns or malware, too, because there is no more need for databases of malicious code or attack prototypes, which become obsolete within hours anyway due to the swift spreading of current malware via the Internet. In addition, separation and structuring considerably facilitates the maintainability of computers. Systems protected by the mentioned measures exhibit, on the basis of disclosing their nominal functions, of the permanent supervision against set bounds, of the context-sensitive allocation of data and of the impossibility to attack operating systems and application programs, a degree of robustness which allows them to maintain their functionality despite some failing application programs. In essence, with their features

- Data and instructions are separated throughout.
- Authentications are not influenceable by software.
- Protection systems are not attackable themselves; their implementation is proven correct and safely protected against modifications not permitted.
- The protection of systems cannot be put out of effect during the installation phases of application programs or of operating system components as well.
- All storage levels (main memory, mass storage etc.) are protected throughout against unauthorised accesses by means of authentication-dependent virtual address spaces.
- Constraints and nominal functionalities of programs are defined in installation phases, and permanently supervised during operation. Their observance is guaranteed under real-time conditions.
- To protect data against effects of software errors or malicious interpretable files, and to enable context-sensitive memory allocation, programs can be instantiated with access functions.
- Requests arriving from the outside are always placed first in separate and enclosed data areas to be preprocessed there.

these measures guarantee, with reference to [11], the observance of the protection objectives

Privacy: unauthorised gain of information is made impossible, i.e. spying out of data is obviated,
Integrity: unauthorised modification of information is precluded,
Availability: unauthorised influence on the functionality is precluded and
Attributability: at any point in time the responsible persons can be identified with certainty.

4 Information-Theoretically Secure Data Encryption

In information and communication technology, increasingly datasets of any size are exchanged between computers in form of streams via data networks. This holds particularly for the communication between cloud servers and cloud clients. To guarantee the confidentiality of such messages' contents, a plentitude of methods for the encryption of the data streams was developed [10]. Currently used encryption methods usually employ the same keys during longer periods of time, lending themselves to cryptanalytic attacks. It was shown, for instance, that the rather widespread asymmetrical RSA-cipher with 768 bits long keys has at least theoretically been broken. The symmetrical cryptosystem DES is already regarded as unsafe, too. Other ciphers such as 3DES or AES are still being considered safe, but only because the presently available computing power is insufficient to carry out simple brute-force attacks. In some countries law requires to deposit the keys used with certain agencies. Thus, these countries' secret services do not need any cryptanalysis whatsoever to spy out encrypted data.

In consequence, only perfectly secure one-time encryption is feasible in the long run. Perfect security is achieved, if encryption of a plaintext yields with equal probability any possible chiphertext, and if it is absolutely impossible to conclude from the ciphertext to the plaintext. According to the theorem of Shannon [12] fundamental for information theory, a cryptosystem is only then regarded as perfectly safe, if the number of possible keys is at least as large as the number of possible messages. Hence, also the number of keys is at least as large as the one of possible ciphertexts which, in turn, must be at least as large as the number of possible plaintexts.

Based on these considerations, in the sequel a cryptosystem is presented, which does not only work with one-time keys. It allows any plaintext to be encrypted by a randomly selected element out of a large set of possible ciphertexts, and it obliterates the boundaries between data items encrypted together. Thus, it is impossible to conclude from boundaries between data items in ciphertexts to the boundaries of data items in the plaintexts, removing another toehold for cryptanalysis.

The general operational principle of cryptosystems can be described mathematically as follows. Let sequences of message symbols (plaintext) $s_0,\ s_1,\ \ldots$ from an arbitrary alphabet S to be transmitted. Messages are encrypted by the sender with an encryption function and, after transmission, decrypted by the

receiver with the inverse decryption function. Generally, both functions are publically known according to Kerckhoffs' principle [8], but parameterised with a secret key K, which is agreed upon by the the communicating units via a confidential and authentic channel. In dependence upon this key a sequence of states

$$\sigma_{t+1} = f(\sigma_t, K) \tag{1}$$

is generated both in the sender and the receiver at discrete points in time $t \geq 0$ with the state transfer function f, and a key stream

$$z_t = g(\sigma_t, K)$$

is generated with the key stream generation function g. The initial state σ_0 may be known publically, or may also be derived from the key K. With the key stream, plaintext is then transformed in a state-dependent way by the invertible mapping

$$c_t = h(z_t, s_t) \tag{2}$$

to ciphertext, which is decryptable by applying the inverse mapping

$$s_t = h^{-1}(z_t, c_t).$$

The key stream sequence must be as similar to a genuine random sequence as possible. In case of self-synchronising stream ciphers, the determination of the state σ_{t+1} additionally depends on the last-generated cipher symbols $c_t, \ldots, c_{t-l+1}$ with fixed l, $l \geq 1$.

It is a common feature of all known cryptosystems that they subject the data elements to be transmitted, may that be bits, alphanumerical characters or bytes containing binary data, may they be single or in groups, always as unchanged entities to encryption. The information-theoretical model of cryptosystems according to Shannon [12] is founded on this restrictive basic assumption as well. Consequently, information such as the boundaries between data elements and their number perpetuates observably and not encrypted into the ciphertext: as a rule, to any plaintext symbol there corresponds exactly one ciphertext symbol. Since even block ciphers seldom work with data entities exceeding 256 bits, the symbols in plaintexts and in ciphertexts are ordered in the same sequences or, at least, their positions lie very close together. Thus, corresponding symbols in plain- and ciphertext can rather easily be associated with one another, which considerably facilitates code breaking.

A solution of this problem [7] is based on the observation that, ultimately, in the technical realisation of all cryptosystems the symbols of the plaintext alphabets (or modifications thereof) and of the ciphertext alphabets are all represented by binary encodings. Correspondingly, for encryption the most general among all possible forms of replacing one bit pattern by another one is utilised. In course of this encryption, particularly the boundaries between the plaintext symbols are blurred, the binary positions of several plaintext symbols are functionally interrelated with each other, and for any bit pattern to be encrypted an encryption is randomly selected out of a corresponding set.

The encryption according to [7] differs from the state of the art outlined above as follows. A state sequence is generated similar to Eq. (1). In every state σ_t, however, a number m_t of bit positions to be encrypted is determined according to an arbitrarily selectable method. The parameter m_t can – and should – be different from the number of bit positions, with which the plaintext alphabet is encoded, whereby the boundaries between the plaintext symbols are annihilated. Then, for m_t bit positions each, an encryption with $n > m_t$ bit positions is determined by means of a state-dependent relation

$$R_t \subset \{0,1\}^{m_t} \times \{0,1\}^n.$$

Here, the parameter n may not be smaller than m_t, as information would get lost otherwise, and it should not be equal to m_t either, in order to prevent the disadvantages mentioned above. Contrary to the function h of Eq. (2), the relation R_t does not need to be a mapping: it is even desirable that with every element in $\{0,1\}^{m_t}$ as many elements of $\{0,1\}^n$ as possible are related by R_t, allowing to randomly select among them one as encryption. Moreover, every element in $\{0,1\}^n$ should be a valid enciphering of an element in $\{0,1\}^{m_t}$, to completely exhaust the encryption possibilities available. Unique decryptability is given, when the inverse relation is a surjective (onto) mapping:

$$R_t^{-1} : \{0,1\}^n \longrightarrow \{0,1\}^{m_t}.$$

Contrary to Kerckhoffs' principle, this decryption function is not known publically – and the relation R_t used for encryption is not only publically unknown, but no function either. Publically known is only, that R_t^{-1} is a totally arbitrary mapping among all possible ones mapping the finite set $\{0,1\}^n$ onto another finite set $\{0,1\}^{m_t}$.

The conditions mentioned above already hamper to a very large extent breaking a data encryption performed according to [7]. A decryption would only be possible, if an attacker had such an amount of ciphertexts available as required by pertaining analyses – totally disregarding the necessary computational power. The following further measures prevent, however, that sufficiently long encipherings, generated with a certain choice of a parameter set and an encryption relation, arise in the first place.

During operation, an encryption unit can – at randomly selected points in time – sufficiently often vary the parameter m_t between 1 and the length of an input register and, thus, modify the encryption relation correspondingly. The unit as well as an inversely constructed and working decryption unit can coordinate sporadic modifications of the encryption using a protocol. In order to transmit between the units as few details on the encryption relation as possible for confidentiality reasons, just the instants of re-parametrisations ought to be co-ordinated. The respective new parameter values and identifiers of the en- and decryption relations to be applied should, however, be generated in both units with synchronously running algorithms. Random number generators based on chaos-theoretical principles [9], for instance, are very suitable for this purpose.

By selecting the parameters $m_t \neq k$ and $n > m_t$ it is inherently achieved, that it cannot be concluded in an easy way anymore from the boundaries between

the symbols in a ciphertext to the boundaries of the symbols in the plaintext data stream. For $n > m_t$, the set of possible encryption elements is embedded in a considerably larger image set, significantly impeding code analysis for an attacker. The number of all possible relations $R_t \subset \{0,1\}^{m_t} \times \{0,1\}^n$, for which R_t^{-1} is a surjective mapping, amounts to $\frac{2^n!}{(2^n - 2^{m_t})!}$. For the choice $m_t = 17$ and $n = 24$, considered to be practically feasible, this is in the order of magnitude of $10^{946,701}$ different relations – an extremely big number. The set of these relations comprises, among others, all possibilities to permutate bits in their respective positions, to insert $n - m_t$ redundant bits, which may each have an arbitrary of both possible values 0 or 1, at $\binom{2^n}{2^{n-m_t}}$ positions in the output bit patterns as well as to functionally interrelate the values of the bit positions of the encryption elements in a fully general way.

5 Conclusion

The security risks brought about by cloud computing have been identified. If clouds are just used as data depositories and unbreakable encryption is employed, then cloud providers can neither spy out these data nor modify them without being noticed. The on-going research on enabling the processing of encrypted data without having to decrypt them is doomed to failure since the conditions, under which this might be possible, are too restrictive by far. Therefore, in this respect the only choices are not to use cloud computing, to trust the provider or to own the cloud.

Currently applied cryptosystems to secure confidential data transmission to or from clouds have either already been broken or are expected to be broken soon. Also, their keys need to be deposited with government agencies in some countries. Therefore, to prevent eavesdropping and gaining unauthorised access to clouds, information-theoretically secure one-time encryption is the method of choice. It was shown that encryption with one-time keys can even be enhanced by encrypting any plaintext by a randomly selected element out of a large set of possible ciphertexts. Blurring the boundaries between data items encrypted together, it is made impossible to conclude from boundaries between data items in ciphertexts to the boundaries of data items in the plaintexts. Thus, a toehold for cryptanalysis left open by a silent assumption in Shannon's communication theory is eliminated. The operation principles of malware – present and future – intruding both cloud servers and client computers can easily thwarted if one resorts to hardware-supported measures, whereas software-based ones can inherently not meet the expectations. The simple measure of separating program and data memories, i.e. the Harvard architecture, is fully effective. Perpetuating the inadequate Von Neumann architecture any longer is dangerous and does not make sense. If the Harvard architecture should be considered too restrictive, it is also possible to efficiently harden the Von Neumann architecture with several hardware-based security features. For the majority of these features, here it was referred to patent applications of the year 2000 [3,4,6], which were not granted

later. The patent office showed the applicants that they had re-invented the wheel, and could prove that similar ideas had been published already decades before. Fourteen years have passed since then, but the ignorance of the professional circles persists.

References

1. Fahrnberger, G.: SecureString 2.0 – A Cryptosystem for Computing on Encrypted Character Strings in Clouds. In: Eichler, G., Gumzej, R. (eds.) Networked Information Systems. Fortschr.-Ber. 10, 826, pp. 226–240. VDI Verlag, Düsseldorf (2013)
2. Halang, W.A., Witte, M.: A Virus-Resistent Network Interface. In: Górski, J. (ed.) SAFECOMP 1993, pp. 349–357. Springer, Heidelberg (1993)
3. Halang, W.A., Fitz, R.: Speichersegmentierung in Datenverarbeitungsanlagen zum Schutz vor unbefugtem Eindringen. German patent application DE 100 31 212 A1 (2000)
4. Halang, W.A., Fitz, R.: Kontextsensitive Speicherzuordnung in Datenverarbeitungsanlagen zum Schutz vor unbefugtem Ausspähen und Manipulieren von Daten. German patent application DE 100 31 209 A1 (2000)
5. Halang, W.A., Fitz, R.: Gerätetechnische Schreibschutzkopplung zum Schutz digitaler Datenverarbeitungsanlagen vor Eindringlingen während der Installationsphase von Programmen. German patent 10051941 since 20 October (2000)
6. Halang, W.A., Fitz, R.: Offenbarendes Verfahren zur Überwachung ausführbarer oder interpretierbarer Daten in digitalen Datenverarbeitungsanlagen mittels gerätetechnischer Einrichtungen. German patent application DE 100 55 118 A1 (2000)
7. Halang, W.A., Komkhao, M., Sodsee, S.: A Stream Cipher Obliterating Data Element Boundaries. Thai Patent Registration (2014)
8. Kerckhoffs, A.: La cryptographie militaire. Journal des Sciences Militaires. 9. Serie (1883)
9. Li, P.: Spatiotemporal Chaos-based Multimedia Cryptosystems. Fortschr.-Ber. 10, 777. VDI-Verlag, Düsseldorf (2007)
10. Menezes, A.J., van Oorschot, P.C., Vanstone, S.A.: Handbook of Applied Cryptography. CRC Press, Boca Raton (1997)
11. Rannenberg, K., Pfitzmann, A., Müller, G.: Sicherheit, insbesondere mehrseitige IT-Sicherheit. In: Mehrseitige Sicherheit in der Kommunikationstechnik, pp. 21–29. Addison-Wesley, Bonn (1997)
12. Shannon, C.E.: Communication Theory of Secrecy Systems. Bell System Technical Journal 28, 656–715 (1949)
13. Spruthm, W.G., Rosenstiel, W.: Revitalisierung der akademischen Großrechnerausbildung. Informatik Spektrum 34(3), 295–303 (2011)