Data Security in Service-Oriented Architectures

Dirk Henrici, Jochen Müller

AG Integrated Communication Systems
University of Kaiserslautern
PO Box 3049, D-67653 Kaiserslautern
{henrici,jmueller}@informatik.uni-kl.de

Abstract: Due to standardized interfaces and loose coupling of services, service-oriented architectures provide the possibility for close interaction between different organizations and communities. But this also introduces new risks: To have under control where which data is processed becomes increasingly difficult. This paper highlights that current approaches for ensuring data privacy and required security mechanisms are no longer adequate under these changing conditions and presents possible solutions discussed by researchers and developers. Additionally, economic implications of data privacy and security are considered.

1 Introduction

Service-oriented architectures are characterized by modular design and loose coupling of services. These properties of service-oriented architectures enable a high degree of interaction and cooperation of people and organizations while crossing company boundaries. The introduction of such new technologies is motivated by the desire of companies to increase flexibility, decrease cost and thus to gain a better competitive position.

But besides the advantages, the increasing internetworking of services and data access across company boundaries that already becomes common today lead to new risks: To control where which data is processed becomes increasingly difficult. Also access restrictions and other security provisions can often be bypassed relatively easily, for instance by exploiting vulnerabilities in operation systems, middleware, or applications. Thus, customer data and company secrets in general are in danger. New technology could make life more controlled, a threat feared by many people so that maintaining data privacy is of high interest. Companies need to maintain data privacy to meet the demands of their customers and to protect company secrets.

Today, customers and companies are often confronted with a choice between two options: Either use new technologies and services and live with the related privacy and security issues or opt out and live with current practices or workarounds. For example, one may use his credit card information in the Internet and thus a comfortable form of payment. The alternative is to use other forms like cash in advance which is time-consuming and much less convenient. If a service is no longer available by "traditional" means an opt-out option no longer exists. Already such a simple example shows that modern everyday life of companies as well as individuals is already paved with processes that have serious security and privacy issues.

But are companies and their customers really doomed to tolerate deficiencies in privacy protection and security for the sake of productivity and convenience? Is there really no viable solution for the creation of secure IT systems that are capable of managing the increasing interaction and data processing introduced by service-oriented architectures? This contribution points out that functionality and privacy need not be at odds if a suitable system design and up-to-date techniques are used.

The thesis is that using a proper system design and suitable mechanisms, privacy and security need not suffer for the sake of convenience or functionality. Therefore, revelation and usage of data needs to be restricted and controlled. Not only by legislative regulations but by technical means as well, so that abuse can be precluded effectively. A prerequisite for this are secure systems, because privacy can only be guaranteed in a secure system environment [He04]: If the IT systems are not secure, all efforts for preserving privacy can be easily bypassed.

Webservices are the most widespread mechanism for the implementation of service-oriented architectures. In this context, WS-Security (webservices security), SAML (security assertions markup language), XKMS (XML key management specification), XML-encryption, and XML-signature (see e. g. [To04]) are some of the standards intended to secure webservices. These standards will not be discussed in this contribution, because to employ them is not sufficient for the creation of secure IT systems that are capable to provide privacy. Instead, we will step back and present a more holistic view of the requirements for designing secure, privacy respecting systems.

In the following section, considering as example a contemporary service, the implications of system design for maintaining privacy are shown. Additionally, some not publicly well known building blocks for limiting relinquishment of data are introduced. The example shows that a system design respecting privacy may also have economic consequences.

In section 3 we highlight the current situation in creating secure, privacy respecting IT systems. It will be shown that the current proceeding is not adequate for the increasing demand for security and privacy and thus a paradigm shift is required. In section 4 we will present some guidelines that are intended as starting point for further consideration on the topic. The goal is to make the reader think of security and privacy from different perspectives than usual today.

Interestingly, from an economic point of view, especially privacy is often not a desired feature, at least at first glance. Because of that, economic implications are discussed in section 5.

2 Example: “Find-a-friend” service

In the field of location-based services many new applications emerge. One of them is the “find-a-friend” service, a lifestyle customer-to-customer application: Just like your instant messaging application that informs you when your friends are online, your location aware personal assistant could tell you when a friend is near your current location, for instance within half a mile.

Obviously, such a service handles sensitive information since the identity of the user needs to be associated with his current location [Be03]. This enables tracking of the individual by anybody who gets access to that data. The current location of a user needs to be compared with the locations of the user's friends in order to realize the desired service.

In the following we will design such a find-a-friend service. At first, we will use a straight forward approach. The focus will lie on functionality; privacy issues will be considered afterwards. After that, we will design a second system with the same features but with privacy as determining goal. There we will make use of advantageous techniques ensuring that as little information as possible is exposed. These techniques are partly not very known publicly yet, albeit being a research topic for years. For instance, in currently used applications the use of cryptography is often limited to hashing, enciphering, and deciphering and as a means of implementing authentication and authorization mechanisms. But there are many other possibilities that can be very useful to prevent revelation of sensitive data. Of course, the presented techniques can be applied to other applications as well.

2.1 Straight forward approach

Find-a-friend is a service, so we need an appropriate service provider. This would in practice be an independent company.

The operation of the system itself is then rather simple (see figure 1): Each user obtains its current location by GPS, according to the cell-ID of the user's mobile phone, or from any other available geo-information system and sends it regularly to the service provider. The latter keeps track of the user and all the other registered ones and gives a notification when the distance between two users being registered as friends falls below a specified value.

Of course, in this setup each user must trust the service provider since the user's identity and the user's location are revealed to it completely. With this data, an untrustworthy service provider can track the user and even estimate future locations of the user by creating a location profile and studying a user's habits. All the information could even be shared with other companies or government without knowledge of the user, thus enabling customized marketing and total surveillance. Imagine a sales representative equipped with all the data. He can offer products tailored to the audience and knows always where to contact you. Or imagine the police follow and analyze all of your recent and past steps (and the ones of the persons you have met) when you are under suspicion.

As establishing a good reputation and legislative regulations are the only limits for a provider not to misuse the confided data and as abuse is very simple and uncontrollable, a user probably wants a better protection of his privacy.

An obvious approach would be not to link the user's identity with its location directly. This can be done using pseudo-identities: In connection with the service provider each user does not reveal its real identity but uses a special identity, for instance a nickname, instead. Already in 1985, Chaum [Ch85] proposed the use of different account numbers or "digital pseudonyms" for each organization to make it harder for them to match or link data records.

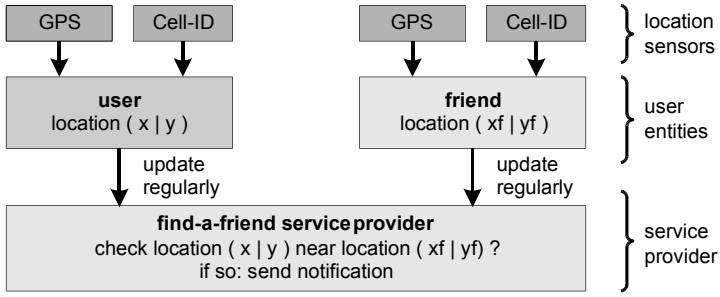

Figure 1: Possible system architecture for a find-a-friend service

Of course, it must be assured that the update messages from the user to the service provider do not reveal the user's real identity. In the paper of Chaum [Ch85] a technique for assuring unconditional untraceability is explained. Other approaches use "mixes" in the line of communication [Ch81] to circumvent traceability of messages.

Further, communication between user and service provider should be encrypted to keep the message exchange private. Using public key cryptography with a private key associated to the user's nickname and the corresponding public key given to the service-provider and another key pair vice-versa, the parties can also authenticate to each other.

Using all these techniques, the complexity of the system has increased significantly. But what about privacy? As long as the association between real identity and pseudonym can be kept private, everything is fine. But unfortunately, that is not easy. Users have to register as friends, and at this stage, the association between user and pseudonym needs to be known. This problem could be solved with public key cryptography: A user does not reveal its pseudonym in clear to his friends but one that is encrypted using the public key of the service provider.

Nevertheless, the association between user and its pseudonym can be unerringly revealed if the service provider matches the pseudonym world with the real world: Since location of the user in the real world is identical with the location stored in the provider's database for a particular nickname, matching is an easy task although fortunately it cannot be automatized.

As we see, all approaches enhancing privacy in the system have an effect but are not an overall satisfactory solution. With more effort and thus making the system even more complex, some issues could be addressed, but the actual problem lies in the system design itself: Too much data converges at the service provider. However, the mentioned building blocks like mixes or public key cryptography may be applicable in other scenarios, there solving the arising privacy and security issues.

2.2 An alternative approach

By giving sensitive data to a third party, the individual loses control of data usage: Given away, data can be used for both good and evil. But do we really need a service provider

whom we have to entrust sensitive information? A user could exchange locations with his friends directly. Unfortunately, this only shifts the problem. Now, he has to trust in all his friends that they do not abuse the data. Another question is, whether we want to tell our exact location even to our friends.

What we need is a way to check, whether two locations are near to each other without revealing the locations themselves. Surprisingly, such computation without revealing the preimages is possible!

The associated field of research called "secure two-party computation" (STC) was introduced by Yao [Ya82, Ya86]. In this context, the "millionaires' problem" has become famous: Two millionaires, Alice and Bob, want to know who of them is richer without revealing their actual wealth. Relevant applications for STC today are data mining or creation of statistics without revealing the data itself.

A primitive for secure two-party computation is "oblivious transfer" (OT) [Cr00]. In "chosen one-out-of-two OT", the basic building block, one party has two secret input bits, b_0 and b_1. The second party has a secret selection bit s. With an OT protocol, the second party learns the input bit (b_s) chosen by its selection bit without learning the other one (b_{1-s}). Thereby the first party does not obtain information about the selection bit.

Goldreich [Go87] proved that any secure multi-party computation problem is solvable using circuit evaluation protocols. While this approach is appealing in generality, for specific applications protocols with much lesser complexity can often be found. Solutions to some common problems are given in [Du01].

Another primitive for secure computation is "homomorphic encryption". An encryption scheme is homomorphic if $E(x) * E(y) = E(x + y)$. As can be seen, the arithmetic operation is performed on the ciphertext without preceding decryption and revelation of the preimages x and y.

Back to our find-a-friend scenario, using secure two-party computation it is possible to check whether a user and his friends are near to each other without revealing the locations or involving a trusted third party.

In [Fe01] an algorithm for calculating the L2-distance between two parties securely is described. With an extension that not the exact distance is revealed to the parties but only whether it is lower or higher a certain value, the algorithm is applicable to our problem.

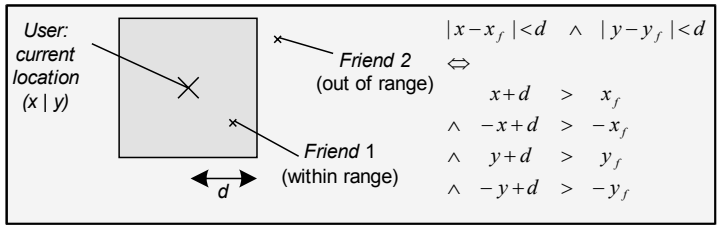

Figure 2: Find-a-friend with square area and its associated inequations

If we are satisfied with checking whether a friend is within a square area around the user we can employ the simpler vector dominance protocol described in [At00]. The protocol tests whether each element a_i of a vector $A = (a_1, \ldots, a_n)$ is higher than each element bi of a vector $B = (b_1, \ldots, b_i)$ without revealing anything about the vectors or the relationship of single elements themselves. In our scenario, the problem is described by setting $A = (x + d, -x + d, y + d, -y + d)$ and $B = (x_f, -x_f, y_f, -y_f)$ which can be gathered from figure 2.

Using this technique, we have developed a find-a-friend service that ensures that location information is kept private since no sensitive data at all is revealed to other parties. Note that the presented solution is a constructed example in that it is not suitable for real world application: The possibility of active attacks where a malicious party changes its own position from on calculation to another until it gets a location match is not considered. Nevertheless, comparison of the two presented solutions is a good starting point for discussing security and privacy issues and solutions.

3 Current Situation

In Germany, the "Bundesdatenschutzgesetz" [BDSG] (federal data protection law) is the legal framework that has to be followed. It aims at avoiding that personal data is gathered wherever possible. If personal data has to be processed its amount must be as low as possible and the data may only be stored and processed when a legitimate interest exists and with predetermination for specific purposes. The law leaves room for interpretation and does not recommend procedures and methods (besides blinded storage) for actually implementing data privacy. Also no statements regarding non personal data are given. Because of that, the German law gives no guideline; it just sets limits on storage and processing of personal data.

Storage and processing of non personal data is thus dependent on the requirements of companies and the demand of their customers and originators. Data which is critical for a company and may not be revealed to competitors is taken more care of than uncritical data. Therefore, the effort to maintain privacy of non personal data depends on its value.

In other terms, the risk that is associated with storage and processing of that kind of data shall be kept as low as possible. The risk can be calculated using the following risk equation: Risk = Threat x Vulnerability x Cost of Asset

By technical means, the level of protection can be increased and therewith the vulnerability can be kept low. Due to economics, the cost of keeping the vulnerability low and avoiding threats should never be higher than the risk. Thus an optimal balance needs to be found.

In practice, the risk equation is of limited use, because it is difficult to give numbers for threats, vulnerabilities, and costs. Nevertheless, it can be interpreted qualitatively to give us a guide how to design secure, privacy respecting systems from an economic point of view: Threats should be avoided and the vulnerability should be kept low with as few efforts as possible. Unfortunately, both design goals are often not properly addressed in today's IT. This leads to immense problems, because with the interconnection and

interaction of companies using service-oriented architectures, threats and vulnerabilities are likely to increase.

Avoidance of threats and limiting vulnerabilities is not the usual procedure today: Something bad must happen, at least to others, until an investment into securing IT takes place. Each threat must first be brought to mind, before people start to address it effectively. Many examples can be found to support this thesis. For instance, in March 2005 Time Warner lost backup tapes with data on 600,000 employees. In reaction of this event, the company started to encipher its backup data (see e. g. [CW05]).

Pure reaction to security incidents also leads to a patchwork of solutions instead of well-thought out homogeneous and consistent secure system design. Unfortunately, limiting vulnerabilities and thus IT security is today for many managers equal to firewalls, patching operating systems and applications with security fixes, enciphering data, and newly applying PKI (public key infrastructure). But since a chain is only as strong as its weakest link, a holistic view of IT security and data privacy is required. Therefore we need a paradigm shift to address the increasing requirements towards IT security and data privacy.

4 Measures

In the previous section we argued that the current proceeding for gaining IT security and data privacy is not adequate for the increasing requirements raised by service-oriented architectures and other new technologies.

But what can be done to meet current and future demand? In the following we will present a number of guidelines that help to implement secure, privacy respecting IT systems. The guidelines are not independent from each other but are often interwoven. Note that the following measures are not intended to be a complete list. They just have the goal to make the reader think of security and privacy from different perspectives.

4.1 Guard against threats and vulnerabilities

An important guideline is to act foresighted and to try to anticipate potential problems beforehand. In the worst case, harm already taken place cannot be undone any more: If any confidential data has become public, improving the security level afterwards can only prevent a similar future event but can by no means limit the damage already taken place. In case of the lost data, its usage is no longer under control of its owner and the data can be abused for any purpose.

However, the need for preventive measures is often not considered as much as necessary. A main reason is that the human perception of risk often does not match the actual risk. For instance, revelation of data does not seem like a big deal at first: Data is just some text and numbers, nothing concrete, nothing life threatening. The threatening things are the ones that can potentially be done with data. For example, these things may even be the loss of one's livelihood in case of identity theft. But such consequences are difficult to imagine and thus are easily misjudged.

Examples for taking reactive measures instead of preventive ones are antivirus applications and spyware checkers in case they are used for cleaning an infected system. It would be

better to prevent malware from getting executed and nesting in the system, for instance by email filters or by means of a securely designed operating system.

It is important to learn from mistakes made in former times so that mistakes are not repeated any more and to better foresee possible vulnerabilities. Nevertheless, if one takes a look at security hotfixes for operating systems and applications often the same types of mistakes and sometimes even the same mistakes occur repeatedly. One of many examples is Microsoft's incorrect application of the RC4 stream cipher [Wu05] in Word and Excel applications. The same mistake has already been made in 1999 in its operating system NT4.

A good approach for guarding against vulnerabilities is to separate applications and parts of an application that require different security levels cleanly. An example for maximum separation is sandboxing, i. e. running applications in a separated environment with only limited and defined possibility to interact with its surroundings. Examples are the Java-sandbox in which Java applications are executed or chroot-environments in Unix. Example of poor design is close interlocking of applications or even interlocking between applications and the operating system (e. g. the Internet Explorer as a core component of the operating system). Reasons to do things like that are either poor application design or the desire to gain maximum performance. But with today's computing power, gaining the endmost pinch of performance at the expense of clean design, lower complexity, and less vulnerability is obviously a bad trade-off.

4.2 Proper system design

Today, security and privacy are often seen as an add-on, i. e. an addition to an else insecure system. Therefore, development often takes place in the same manner: At first get the system to work; afterwards add security elements like enciphering or authentication. But as shown in the previous section and also in the find-a-friend example, proper system design is a crucial element in designing secure, privacy respecting IT systems. Thus, security and privacy are not orthogonal to the overall system architecture and therewith with those cannot be dealt with independently.

Another example of improper system design regarding security is the SMTP [Kl01] which is used to deliver emails in the Internet. The protocol was originally designed without having security in mind. This leads to the problem that today's mailboxes are filled up with spam, mostly having forged sender's addresses. Adding effective security features to the running mail system is almost impossible since it would require a redesign which would disrupt compatibility to the existing mail system. Similar issues exist for example in operating systems that cannot be easily redesigned without breaking application compatibility.

As shown in the find-a-friend example, considering security and privacy already at early stages of system design gives much more leeway in the selection of proper mechanisms and in their implementation. It has the additional advantage that the developer is urged to find a cleaner system design. This is caused by the fact that designing secure systems requires employing paradigms like separation of concern, layering, or modular assembly

that are also advantageous to overall software quality. Thus, secure systems tend to have a cleaner design and better code than insecure ones.

4.3 Elimination of causes instead of symptoms

In today's IT many examples can be found in which symptoms of problems are fixed but not the actual problems themselves. The actual problem is often improper system design lacking security considerations as presented in the previous section. To engineer a system with proper security out of an insecure one, usually a patchwork of security tools and fixes is needed for compensation. Such solutions are often incomplete and complicated.

Missing security mechanisms in current widespread operating systems [Lo98] are a good example to fortify this. Security in Unix and Windows is based on user accounts: Each process runs in the context of a certain user and has all the access privileges to resources (like files, network, and devices) that are granted to that user. With this proceeding, processes often have much more access privileges than necessary. For instance, a calculator application or an ordinary screensaver does not need to be able to access the network, a user's files or even system files. Thus, a security model based on user and process is required that enables a user to run applications in different roles with defined access privileges. A step in this direction is SELinux [Lo01] that aims to add such kinds of additional restrictions to the Linux operating system. It is obvious that with process-based security mechanisms many kinds of malware like viruses or trojans could no longer do serious harm and spread as easily as today. Also, malicious or faulty applications could no longer harm other applications, data of other applications, or even system files. But instead of adding proper security mechanisms and solving the actual problem, symptoms are cured by using personal firewalls, anti virus applications, and patching applications and the operating systems with regular bugfixes. Note that measures like the enumerated ones would still be a good supplement; but basing security solely on them is flawed.

The need of process based security in operating systems becomes clear when considering the workarounds used today: Different daemons like mail and web servers run in different user contexts to simulate process based security for them and thus separate them from each other and the rest of the system. This works well for files but not for other resources like the network. To limit network connectivity, "personal firewalls" can be set up so that only cleared application can access the network. This simulates process based security for the network as a resource.

Related to missing process based security in most operating systems is the partition of all applications into trusted and untrusted ones: After downloading an executable from the Internet a user gets a security warning informing him about the risks. The only possibilities are to deny execution or to give full access based on the user's rights. There is no possibility to run the executable in a separated environment with limited access to the system which would be a huge advantage for security. Similarly, personal firewalls and access lists usually only offer the options of denying, giving client access to the network or giving full access to the network. Restricting special functionality that is powerful but seldom needed (e. g. raw sockets) is usually not possible.

In general, a stepped security model is needed that enables granular setting of access restrictions based on user, a user's role, and the process itself. Such a proceeding would provide a much saver environment for system services as well as user applications and prevent malware from performing unwanted tasks effectively.

Besides that, when designing an application, it should be taken into consideration that it could have security vulnerabilities. This means that arrangements should be made for the case of bugs. Ideally, the effects of a security breach can be limited by another security border. This is a matter of trust: Each software module should be considered untrustworthy or only trusted under certain assumptions. Today, often all software modules are considered equally trustworthy so that security checks and input validation is often neglected in intern interfaces.

4.4 Utilization of all available building blocks

Today's most widespread technical building blocks for achieving security and data privacy are cryptographic hash functions and enciphering/deciphering using a number of algorithms. These are important constituent parts of widespread authentication and authorization software like Kerberos and lately public key infrastructures (PKI). But as highlighted in the find-a-friend example, many other building blocks exist that could be employed. Some examples are STC, polymorphic encryption, anonymization infrastructures, and pseudonymization infrastructures.

Although such powerful building blocks exist they are currently not used to protect privacy and security in a broad scope. This raises the interesting question what the causes for this are. Some potential ones are to be discussed in the following.

The most important one is that these building blocks and the possibilities they offer are still not very known outside the academic area. Because of that, system designers in industry cannot apply the concepts and the general public cannot demand their usage.

But even if a building block, for instance STC, is known to the developer and shall be used, the problem is that libraries implementing these building blocks are virtually unavailable. Because of that developers often have to implement the desired functionality by themselves which is such a lot of work that the option is often abandoned.

For building blocks that are intended to ensure privacy another drawback exists: Debugging of applications as well as management like logging and auditing is more complex. The reason is that for privacy reasons data is usually no longer kept centrally but split between several entities (shared secrets paradigm). Bringing such data together for debugging or auditing purposes becomes even more difficult if pseudonyms are used. Similarly, bringing data together for data mining purposes that does not harm privacy is more complex and can only be done with lower performance when privacy enabled applications are used instead of applications not respecting privacy. This counteracts the goal of companies to gather as much data as possible about their customers to be able to effectively learn about their requirements and interests.

4.5 Supporting the developer with standard libraries

As already mentioned in the section above, many building blocks that could help building secure and privacy respecting applications are not available for developers in form of libraries that can be easily used. The need for developers to implement such functionality on their own is error prone and not cost effective. Fortunately, this issue is more and more getting addressed, in particular in the open source community that is making many libraries available for anybody's use (see e. g. [Sf05]).

It is very important that developers stick to open standards. This enables a developer to use different implementations of a standard without requiring learning new concepts and substantial changes in their application. Therewith standards help to reduce cost and to ensure interoperability.

In the context of security and privacy, standards also are a necessity by other means: One should always use widespread standard techniques instead of proprietary ones. For instance, it cannot be mathematically proven today that algorithms (like cryptographic hash functions) have the characteristics one would ideally require. But standard cryptographic algorithms like the hash function SHA1 [Sh93] are much better understood and tested by researchers and the community than any proprietary algorithm will ever be. Additionally, proprietary algorithm's have often flaws because designing cryptographic algorithms often has many pitfalls. Thus the security of proprietary algorithms is often mainly based on obscurity. This is considered bad design in modern cryptography; today's cryptography is based on Kerckhoff's principle [Ke83] that a cryptographic system should not be required to be kept secret.

Employing libraries and open standards has the advantage that complexity is reduced by their use: A developer does not need to implement everything on his own but can employ complete, tested, ready-to-use code.

4.6 Making security and privacy part of the business model

Security and privacy should not be regarded as unwanted necessities that just need effort and thus generate cost for implementation. They should be regarded as an opportunity to distinguish a company from its competitors. Consideration of security and privacy can act as a sales argument that can have much significance to customers. This could make the investment into security and privacy profitable from an economic point of view as well. A more detailed consideration of economic implications is presented in the following chapter.

5 Economic Considerations

As stated in the introduction and pointed out in the scope of the example, privacy cannot be regarded as an add-on like an additional feature. Instead, privacy needs much consideration in early stages of system design. This is an additional effort that is at odds with the wish to get a running system as soon as possible and therewith a fast time-to-market.

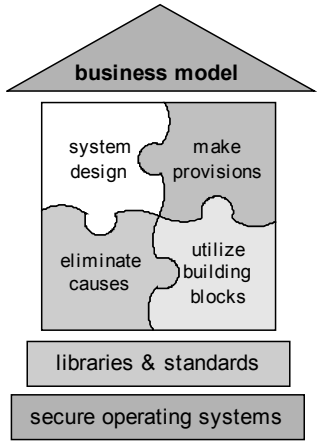

Figure 3: Interrelation of business model and IT security

This would not be a problem if a better protection of privacy would have positive effects that would outweigh the additional effort. So, what's the value of privacy? Are customers willing to pay for security and protection of their privacy?

Polls indicate that the general awareness regarding privacy issues is high and that its protection is rated important [EU03]. But a big problem is that privacy is a "soft" feature: Different people have different opinions and expectations about it. This is represented by the diversity of privacy legislation of different countries [Ep03] as well as in other literature (e. g. [Ja03]). Furthermore, the privacy level reached by a particular software or service is difficult to rate from the viewpoint of an outsider because it requires deep understanding of the internal processes.

To summarize, from an economic point of view, a system designer needs to gather what their target group really wants. After actual implementation, marketing has to make clear that an offered product or service corresponds to those expectations. Only then a customer entirely realizes the added value of a privacy-enabled product and is willing to pay for it.

As these are complex requirements that require coordination between different parties like social research, engineers, and sales, the revenue in customer satisfaction is consequently only worth the effort if the demand of customers for higher protection of privacy is high enough.

Interestingly, this demand seems to be not that high as one would expect due to results of the mentioned polls. Several reasons for this can be imagined: The first is, that functionality and possibilities are more concrete than protection of privacy and that customers are not aware of the things they could demand. Actual demand comes if one needs something because a lack of functionality has been experienced. For privacy this becomes only relevant after abuse of data has been experienced (or can be expected in the short term). Demand can also be created by some nice aspect that has been seen in another implementation of a system or service and thus becomes a desired feature. This will probably become more and

more relevant as soon as protection of privacy becomes realized as useful for marketing purposes.

Besides the missing recognition of the possibilities for the protection of privacy, it can be noticed that customers become used to privacy violations as long as they do not perceive them as a worrying issue. For instance, data collected by rebate companies (purchasing patterns etc.) that are used and sold for marketing purposes do not become a concern for a customer as long as the direct advantages (here: a discount) outweigh the perceived risk of privacy violations.

In consequence, today's customer demand for protection of privacy is not yet very high. But it will probably rise after increasing awareness caused by experienced abuse.

For a service provider, the implementation of privacy features is not desirable as long as customer demand is too low. The reason is that companies are interested in collecting as much data about customers as possible for Customer Relationship Management (CRM) purposes.

As we have seen in the find-a-friend-example, the protection of privacy can also affect the market itself: In the extreme, a service provider could become obsolete – and nobody pays for a service not being provided.

6 Conclusion

As shown by the find-a-friend service example, only an appropriate system design respecting privacy and security as fundamental prerequisite from the beginning of the design process on leads to optimal solutions.

Sensitive information should only be revealed when it is really necessary. Even when information divulgement seems to be inevitable at the first sight, with appropriate techniques often a better way can be found. Besides "standard" methods like encryption and decryption, techniques like secure two-party or even secure multi-party computation can be employed.

Unfortunately, albeit being well observed in research for years, these possibilities are still not known to most application designers and developers, decisions makers, and least to general public. Besides that, companies like service providers or government are not that interested in using all possible algorithms for ensuring privacy: They prefer collecting as much data as possible as long as customers or citizens, respectively, do not complain. Besides that, economic considerations lead to the observation that implementation of mechanisms for the protection of privacy is currently often undesirable for industry – at least at first sight.

In consequence, privacy advocates should not only complain about missing protection of privacy but do awareness training of the public. The public needs to know that powerful techniques for securing IT systems and implementing privacy protection exist. Only informed customers can create enough market demand and therewith pressure for industry to implement privacy protection in their software and services.

We highlighted that the emerging requirements introduced by interacting organizations using service-oriented architectures can only be addressed by conceptual changes in the design of operating systems and applications. We brought to mind that the technical means for securing IT systems and protection of privacy in privacy sensitive applications do exist. We also illustrated that doing much for security and privacy is not enough – the right things need to be done using suitable techniques. Therewith we want to contribute to raise awareness for security in today's IT systems and privacy issues in society and for the existing technical possibilities for solving privacy related problems.

References

[At00] Atallah, M. J.; Du, W: Secure Multi-party Computational Geometry. Lecture Notes in Computer Science, vol. 2125, pp. 165-179, 2000

[BDSG] Bundesdatenschutzgesetz, Bundesgesetzblatt (BGBl. I 2003 pp. 66ff), 2003. Web: http://www.bfd.bund.de/information/BDSG.pdf

[Be03] Beresford, A., Stajano, F.: Location Privacy in Pervasive Computing, IEEE Pervasive Computing, vol. 2, no. 1, pp. 46-55, 2003

[Ch85] Chaum, D.: Security without identification. Communications of the ACM, 28(10): 1030-1044, 1985

[Ch81] Chaum, D.: Untraceable Electronic Mail, Return Addresses, and Digital Pseudonyms. Communications of the ACM, vol. 24(2), pp. 24-88, 1981

[Cr00] Cramer, R.: Introduction to secure computation. Lecture Notes, University of Aarhus, Department for Computer Science, 2000

[CW05] Mearian, L.: Missing backup tapes spur encryption at Time Warner, ComputerWorld, Mai 2005; Website: http://www.computerworld.com/securitytopics/security/story/0,10801,101589,00.html

[Du01] Du, W.: A Study of Several Specific Secure Two-Party Computation Problems. Purdue University, West Lafayette, Indiana, 2001

[EU03] Special Eurobarometer "Data Protection", Survey of the European Opinion Research Group (EEIG), 2003

[Ep03] Privacy & Human Rights 2003, (see http://www.epic.org/bookstore/epic_books.html), 2003

[Fe01] Feigenbaum, J. et al.: Secure Multiparty Computation of Approximations. Proceedings of the 28th International Colloquium on Automata, Languages and Programming (ICALP '01), 2001

[Go87] Goldreich, O.; Micali, S.; Wigderson, A.: How to play any mental game. Proceedings of the Nineteenth Annual ACM Symposium on Theory of Computing, pp. 218-229, 1987

[He04] Henrici, D.; Müller, P.: Sicherheit und Privatsphäre in RFID-Systemen; VDE-Kongress 2004, VDE-Verlag, 2004

[Ja03] Jacobs, A. R.; Abowd, G. D.: A Framework for Comparing Perspectives on Privacy and Pervasive Technologies, IEEE Pervasive Computing, vol. 2, no. 4, pp. 78-84, 2003

[Ke83] Kerckhoffs, A.: La cryptographie militaire, Journal des sciences militaires, vol. IX, pp. 5–83, Jan. 1883, pp. 161–191, Feb. 1883

[Kl01] Klensin, J.: RFC 2821 - Simple Mail Transfer Protocol, Network Working Group, 2001

[Lo98] Loscocco, P. A. et al.: The Inevitability of Failure: The Flawed Assumption of Security in Modern Computing Environments, National Security Agency. 21st National Information Systems Security Conference, 1998

[Lo01] Loscocco, P. A.; Smalley, S. D.: Meeting Critical Security Objectives with Security-Enhanced Linux, Proceedings of Ottawa Linux Symposium, 2001

[Sh93] Secure Hash Standard, National Institute of Science and Technology USA, Federal Information Processing Standard (FIPS) 180-1, 1993

[Sf05] Sourceforge, Open Source software development website, 2005. Web: http://www.sf.net

[To04] Toms, A.: Threats, challenges and emerging standards in web services security. Technical Report HS- IKI-TR-04-001, University of Skövde, Department of Computer Science, Sweden, 2004

[Wu05] Wu, H.: The Misuse of RC4 in Microsoft Word and Excel, Cryptology ePrint Archive: Report 2005/007, 2005. Web: http://eprint.iacr.org/2005/007

[Ya82] Yao, A.C.: Protocols for secure computations. Proceedings 23rd Annual IEEE Symposium on Foundations of Computer Science, pp. 162-167, 1982

[Ya86] Yao, A.C.: How to generate and exchange secrets. Proceedings 27th IEEE Symposium on Foundations of Computer Science, pp. 162-167, 1986