A Framework for Evaluating the Usability and the Utility of PKI-enabled Applications

Tobias Straub[1] and Harald Baier[2]

[1] Darmstadt University of Technology, Computer Science Departement
D-64283 Darmstadt, Germany
tstraub@gkec.tu-darmstadt.de
[2] Darmstadt University of Technology, Darmstadt Centre of IT Security (DZI)
D-64283 Darmstadt, Germany
hbaier@dzi.tu-darmstadt.de

Abstract. Besides the pure technical features, the usability of a PKI-enabled application plays a crucial role since the best security application will fail in practice if its usability is insufficient.
We present a generic framework to evaluate the usability and utility of PKI-enabled applications with respect to their security features. Our approach is modeled on the Common Criteria methodology and consists of 15 evaluation categories that cover all the relevant topics, namely deployment, ergonomics, and technical features.

Keywords: Common Criteria, cryptography, ergonomics, evaluation, PKI, usability, usefulness, utility

1 Introduction

A public key infrastructure (PKI) based on key pairs and digital certificates is the fundamental architecture to enable the use of public key cryptography. A lot of today's commercial off-the-shelf (COTS) software has already implemented some form of PKI support, consider e.g. email clients, web browsers, file encryption software, groupware, VPN clients to name only a few. These so-called *PKI-enabled* applications are the focus of attention in this paper.

In order to achieve the security goals authenticity, integrity, and confidentiality, a PKI-enabled application has not only to provide an appropriate implementation, but also a sufficient level of usability. The user *must* be able to make use of the security features in practice, otherwise the best security application will fail. The last statement is due to the fact that users' behaviour is often the biggest risk in security software. This is confirmed by a survey prepared by the KPMG consulting group in cooperation with the security magazine kes [12].

[1] The first author's work was supported by the German National Science Foundation (DFG) as part of the PhD program "Enabling Technologies for Electronic Commerce" at the Darmstadt University of Technology.
[2] The paper was written while the second author was working within the project SicAri, funded by the German Ministry for Education and Research.

Incidents caused by users are at the top of the list (52%) of security threats. To draw a comparison, malware like viruses, worms, or Trojan horses is responsible for only 25% of the security incidents. However, both areas are often linked together.

The present paper grew out of a usability study to evaluate how applications can be secured using PKI [2]. We carried out this project in 2003 by the order of Microsoft Deutschland GmbH, examining a total of 38 different products ranging from standard applications (email client/server, web browser/server, groupware), access control tools for local and remote login (WLAN, VPN) to CA software and cryptographic APIs for application developers.

The main contribution of the paper at hand is the development of a generic framework to evaluate the usability and utility of PKI-enabled applications. An evaluation yields a numeric score which can be weighted to fit the actual priorities and external factors like the intended use case or budget constraints. In our opinion, such a score offers a good basis for making a decision in favour of a particular PKI-enabled application. The framework may as well be useful as a requirements specification for application designers. We are not aware of any comparable framework for evaluating PKI-enabled applications.

2 Related Work

In this section, we will briefly familiarize the reader with the subject of usability in the context of security. We use a taxonomy which is due to Nielsen [9]: *Usability* and *utility* are considered common sub-categories of the category *usefulness*. Usefulness and other characteristics like costs or reliability are aspects of a system's *practical acceptance*. The question whether a system, in principle, has the functionality necessary for a certain task is a matter of utility, whereas usability refers to how this functionality can be used in practice.

The design of human-computer interfaces in general has been studied extensively (see e.g. [9] for an introduction). However, this does not hold for the special case of security applications, not to mention PKI-enabled software. Research in this field has been somewhat focused on the usage of password-based security mechanisms since they are a wide-spread authentication method (see e.g. [10, 1] for a survey).

It has become a common belief that users' behaviour is often the biggest security risk [12, 11]. Nevertheless, a paradigm shift towards the *user-centered* design approaches, which is urgently needed, has failed to appear [5, 1]. As Davis [3] pointed out, users have to pay a certain price for the technological benefits of public-key compared to symmetric-key cryptography. Systems using public-key cryptography transfer responsibilities to the users that are otherwise being centrally handled by a server or administrator. Among these burdens are the management of keys, certificates, and status information.

An empirical evaluation of a PGP-capable email client confirmed the assumption that this is in fact too hard to handle for average users. Whitten and Tygar [13] showed that in practice this software does not achieve a sufficient level

of security due to its usability deficiencies. Their paper also identifies a number of security-inherent properties which are listed below. These properties need to be addressed by an implementation; we will often refer to them in Section 3.

(P1) Unmotivated user property Since security is hardly ever a user's primary goal, he cannot be expected to pay too much attention to it or, for instance, go through voluminous manuals before using the software.

(P2) Abstraction property Especially in the field of public-key cryptography and PKI, it is a difficult task to find intelligible yet precise (real world) metaphors for abstract mathematical objects like key pairs or certificates.

(P3) Lack of feedback property Applications should above all enforce security in a seamless way. In cases where interaction is required, the user should be provided with the necessary feedback to allow for an appropriate decision. Presenting a complicated matter in a short dialogue is challenging.

(P4) Barn door property Dangerous and irrevocable mistakes must be ruled out from the start. If a secret has been revealed at a certain point of time, say during an insecure network connection, one cannot say whether an attacker might already know it.

(P5) Weakest link property Security is considered to be a chain the weakest link of which determines the strength of the whole system. As a consequence, users need to comprehensively take care of an application's security settings.

Several standards define guidelines for ergonomic user interfaces (UI) in general, for instance ISO 9241 [7]. Taking the before-mentioned properties into account, an adjustment of the design principles to the context of security software is necessary [4]. The relevance of the criterions of ISO 9241 Part 10 ("Dialogue principles") for security software is outlined in Figure 1. The modified principles can serve both as a practical guideline for UI designers and for evaluation purposes. Our framework presented in Section 3 will also refer to these principles sometimes.

Another usability evaluation framework is due to Kaiser and Reichenbach [8]. It classifies different types of usability and security problems taking user errors and their security relevance as an indicator. This framework distinguishes between problems that can be traced back to a user's lack of security competence and problems which arise despite a user's skills. The motivation to do so is the observation that these two problem classes need different treatments during a redesign process of the software.

A very important framework for the evaluation of IT products and systems with respect to their security mechanisms are the Common Criteria [6] which are a result of merging standards from several European countries (ITSEC), the U.S. ("Orange Book"), and Canada. Despite their complexity, usability has only marginal relevance in the Common Criteria.

3 The Framework

In this section, we present our framework to measure the usability and utility of PKI-enabled applications with respect to their security features. The rating pro-

general principle	relevance for security applications
error tolerance	Error prevention is considered the most important principle because of P4 and P5.
suitability for individualisation, controllability	Users should be guided more rigorously where needed to avoid weaknesses due to misconfiguration or mistakes (P4).
suitability for learning	Humans tend towards the "trial and error" method when using an application for the first time (P1). This must not lead to severe errors.
self-descriptiveness	Presenting security mechanisms and error messages clearly and precisely is crucial (P2, P3).
conformity to user expectations, consistency	The application should use a correct, homogeneous and usual terminology to describe things (P3).
suitability for the task	Users want to accomplish security-related tasks easily and efficiently. The design should take into account that users have varying abilities (P1).

Fig. 1. UI design principles for security applications (partially based on [4]).

cess resembles that of evaluating security products as proposed in the Common Criteria [6]. Since the Common Criteria methodology is an ISO standard and receives worldwide acceptance, we consider it a reasonable basis for our approach, too. Throughout our framework we will often use the term *user*. Depending on the concrete scenario, this may either denote an end-user or an administrator.

3.1 Evaluation Categories

In all, our framework consists of 15 evaluation categories which are evenly spread in three groups named *deployment issues*, *ergonomics*, and *security features*. The motivation behind this choice is Schneier's famous saying "security is a process, not a product" [11]. We thus have to follow an integral approach and must not restrict ourselves solely to ergonomics or technical features. To illustrate the ideas behind each category and to simplify the evaluator's task, we propose a number of exemplary questions. These *central questions* have to be investigated throughout the evaluation to obtain a category score.

In order to make use of a PKI-enabled product, it has to be deployed. For this reason, the first category group is named *deployment issues*. For instance, the user has to obtain the product and install its components which often means that he has to be familiarized with the PKI features before being able to use them. We turn in detail to the deployment issues in Section 3.3. With property P1 in mind, this category should not be underestimated.

Second, the handling of a PKI-enabled application in service is discussed in the category group *ergonomics*. In a classical sense this category group is the core of any usability. Typical categories of ergonomics are the composition of the menus and dialogues, respectively, and the help system available. We present the central questions of ergonomics in Section 3.4.

The categories defined so far may easily be adapted to other use cases than PKI. However, the best product with respect to ergonomics is pointless, if the use case specific security features are substandard. Our third category group *security features* takes the properties P4 and P5 into account. It deals with the algorithms and key lengths in use, the certificate management, and how the validation of a certificate may be done. We turn to the security features in Section 3.5.

3.2 Usability Profiles

In order to obtain a quantitative measure, an evaluator will award points for each category. The result of each category is an integer score in the interval $[-2, 2]$ where 0 denotes an average result, ± 1 a little above/below average result and ± 2 an outstanding/substandard result. The guidelines on how to assign points are part of a so-called *usability profile*. A usability profile has to be defined before the evaluation actually takes place. This can be done based on the central questions of our framework. Each usability profile has to list sample answers and a guideline on how to award points. We provide appropriate clues in the following sections.

The definition of the usability profile may for instance be done by the evaluator himself or by a usability specialist involved in the process. An evaluation result has always to be interpreted with respect to the underlying usability profile. We point out that this procedure is analogous to the definition of so-called protection profiles as described in the Common Criteria.

Our framework allows for an adaptation to different environments since the partial results of a category group are weighted to get an overall score. The weight definition depends on the concrete usage scenario and is part of the usability profile. Figure 2 shows three coarse examples for scenarios with *high security* requirements and skilled personnel (e.g. in the military or a research department), *enterprise* usage with low deployment and helpdesk budgets (with average security demands), and a *private* usage scenario. Let us quickly explain the proposed weighting in Figure 2 in case of private usage: A private user is in general not supported by an administrator, that is he has to deploy the PKI-enabled product himself. In addition, the unmotivated user property P1 requires an easy to deploy and easy to install application. Therefore, deployment is rather important which is why we assign a weighting of 40% to this category group. The

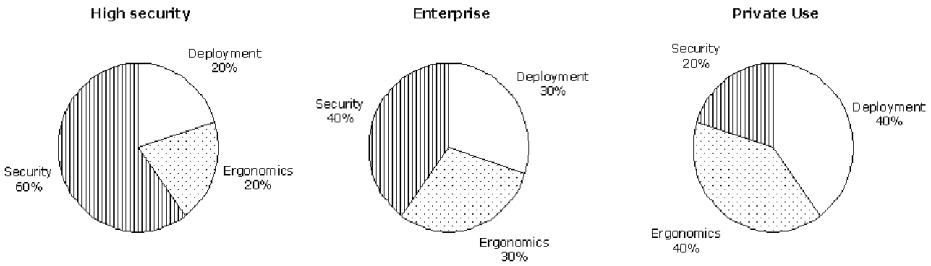

Fig. 2. Sample weightings of the category groups.

properties P2, P3, and P4 motivate a weighting of 40% for the group ergonomics, too. Due to the small assets and threats in this scenario, the security features are of secondary importance and thus weighted by 20%.

3.3 Deployment Issues

We first turn to the category group *deployment issues* which describes the pre-arrangements necessary to set up the system. The central questions listed in the following mainly address property P1, i.e. the user's little motivation to really use built-in security features of existing software or to change to another application of the same kind for the sake of better support of PKI features. Suppose that a computer administrator or an end-user chooses to increase the level of enterprise-wide or personal security, respectively. Then it should be made as easy as possible to realize this plan. People who once tried to improve security but did not succeed are likely to get frustrated and to give up.

In all, the deployment issues comprise the following categories: The first step is *gathering information* about candidate products which is closely related to the question how the software itself can be *obtained*. Since sophisticated applications tend to have special infrastructure demands, it is necessary to consider *technical requirements* before *installing and configuring* the application. During the latter two steps, there may arise the need for technical *support*. Finally, it should be assessed how much *training* is required for the product to be used securely. We present in detail the central questions for each category in what follows.

Gathering Information and Obtaining the Software If an evaluator has no a priori knowledge of appropriate products, it is important that comprehensive information, maybe even an evaluation version, of the software is easily available. In addition, analysis from professional journals or experience reports found in news groups provide more neutral and objective information. It goes without saying that the easier information and the product itself are available, the better the rating is in this category. Central questions are:

- How much meta information (surveys, tests etc.) comparing different products of the same kind is available?
- Is there already enough experience about a product of recent date?
- Does the vendor offer precise product information on his web pages?
- How complicated is it to obtain the product, can it be purchased online?
- Is it possible to upgrade an evaluation copy to a full version simply by an activation code, i.e. without re-installing and re-configuring the software?

Technical Requirements A successful installation often depends on the technical environment in use. For instance, the hardware of a five-year old PC is not sufficient to run current server applications due to performance and memory restrictions. Again, we refer to the unmotivated user property and stress that a good PKI-enabled application should make it easy for users getting started with

it. It should ideally not require high-end hardware nor additional software (e.g. special plugins) to do its task properly. These requirements can only be rated in the light of the actual prerequisites, since some software, e.g. for a certification authority, may in fact have very particular needs. We thus propose that the easier it is for the user to fulfill the requirements of the software, the better the application is rated. In this context, "easier" has to be interpreted in terms of time or money.

- Is it possible to install and operate the application with existing hard- and software?
- Are multiple operating systems supported? Which ones?
- Does the installation require additional programs?
- If the application needs additional hard- or software: Which/how many third-party products are supported?

Installation and Configuration The background of this category is that proper installation and initial configuration are crucial to security with respect to the barn door property P4. On the one hand, there should be enough support, e.g. an installation wizard or a list of commonly used configurations, to guide the user. On the other hand, the default configuration must provide an appropriate security level. This is due to the consideration that it may be a security novice who is installing the software and who is above all not interested in the security features themselves but in getting the software running. An example is the selection of cryptographic parameters like key sizes. While this does not affect the functionality of the software, it is indeed security-relevant.

- Is there an installation wizard guiding an inexperienced user?
- Does the product offer to choose from predefined scenarios?
- How difficult is it to adapt the configuration to individual needs?
- How does the software check user settings for consistency and security?
- Which level of security does the default configuration provide?
- How much expertise is necessary to successfully install and configure the application for the first time?

Technical Support Even if the PKI-enabled software may have advanced installation and configuration wizards, a user may sooner or later need additional help. Therefore, we direct our attention to the quality of support, too. Once again, the criterions in this category strongly depend on the actual application scenario. The rating scheme is analogous to that of the category *technical requirements*.

- What kind of support does the vendor (the supplier) offer? How expensive is that support? Is it charged per request or is there an annual fee?
- How can the user contact the support team? (e.g. phone, email) At what times? (7x24 or office hours only)
- Are there FAQ or manual pages on the Internet?
- Is the information provided in the user's native language?

Training The deployment of every new application should ideally be accompanied by training lessons for users. As we all know, this is more wishful thinking than reality. Due to the abstraction property P2, teaching users to deal with security-related tasks may in particular gain importance where ergonomic deficiencies have to be compensated.

- Which amount of training for the intended users is absolutely necessary to provide an appropriate usage?
- Does the software provide an interactive guided tour explaining its basic functionality?
- How many companies do offer training lessons? At what price?
- Which books do provide course materials?

3.4 Ergonomics

One may argue that *deployment issues*, which we have discussed in the previous section, also fall into the range of usability. To make clear the following difference, we decided to name the current section "ergonomics": Contrary to *deployment issues* which cover non-recurring actions during the setup process, *software ergonomics* is an everyday concern which in general affects much more tasks and people. It may be tolerable that the installation phase requires expert knowledge (which means that the category group *deployment issues* is assigned a smaller weight in the usability profile). But this does not hold for the "operational" phase where people are expected to actively use the PKI-enabled application to get their work securely done.

A natural category to start with is the UI design concerning *menus and dialogues*. The information given there should be self-descriptive using a consistent terminology, whereas its organization should try to anticipate how users will operate the application (cf. Figure 1). Because of property P1, security software should follow the maxim to hide complexity from the average user, acting in a seamless way. Hence, the second category is the *transparency of security features*. Since it cannot be guaranteed a hundred percent that the respective PKI-enabled application will always work without interaction, *warning and error messages* are treated in the corresponding section. It is exactly in this situation when a user may question the *help system*, probably for the first time (à propos "trial and error", cf. Figure 1). From the viewpoint of security, it all depends on how the application *reacts in potentially threatening situations* since errors should be prevented at all costs.

Menus and Dialogues This category groups two types of UI components, the menus being the more static and dialogues being the more dynamic elements. Once again, we think of the user as a security novice and unmotivated subject (P1). Therefore, on the one hand it should be possible to intuitively find *all* the security-relevant menu entries because of P5. On the other hand, dialogues should show a careful design due to the lack of feedback property. Where users

are supposed to change security parameters, there should be clues to underline sensitive settings and a possibility to restore a previous or even a default configuration. A general requirement is that technical terms stemming from PKI and cryptography should be used in a precise and consistent manner.

- Does the organization and structure of PKI-relevant menus follow the general principles used throughout the application?
- How are the security-relevant menu items organized? Are they easy to find and grouped reasonably?
- Does the visual design of the dialogue reflect its level of security relevance? (e.g. special symbols for low, medium, high security relevance)
- Are the technical terms chosen well and intelligible? (even more important if the application does not come in its native language)

Transparency of the Security Features This category refers to a common paradigm to encounter the unmotivated user property. UI experts agree that an ordinary user should at first get into contact with as few security features and settings as possible. However, the more he is interested in the details, the more information the application should provide. This serves a twofold purpose: First, to increase a user's trust in the application by giving further insights. A second aspect is the suitability for learning (cf. Figure 1). A standard technique, for instance, to both hide complexity and allow users to get deeper into it, is an "Advanced Settings" button.

- To what extent does the user have to get involved in security settings to be able to use the features?
- Do the default settings meet the requirements of an ordinary user?
- How difficult is it to change the transparency settings?
- Can a user adjust the transparency level according to his individual skills? (e.g. ordinary user, interested layperson, expert)
- Is every necessary piece of information accessible at all? (e.g. details about cryptographic algorithms and protocols)

Warning and Error Messages In situations where the user or his communication partner behaves wrong or in a non-standard way, this category plays an important role. A user's decision when confronted with a warning or error message is often crucial for security. The user must understand the warnings and behave in an appropriate way. Otherwise he will just click the "close" or the "continue" button. It is therefore indispensable that an explanation of alternative actions and their consequences is given. A good practice is not to force the user to make an immediate decision, which is the case when using modal dialogue windows, but to give him the possibility of proceeding in another way, e.g. browsing the help pages or postponing the task. Consequently, the abstraction property, the barn door property, and the weakest link property are the basis for the following central questions.

- How many details are given in message dialogues?
- Does the text explain what to improve if a single step of a complex PKI operation, e.g. signature verification, fails?
- Are there warnings in case of a security-sensitive operation? What advice is given in such a situation? (e.g. if the chosen key lengths are too small)
- How precise, understandable, and insistent is the information given? Does the message encourage to ignore it?
- Are there useful links to matching topics in the help system?

Help System The built-in help system of an application is the primary source to get information during usage. The help system should be designed to suffice an average user in most of the cases. We think it advisable to have a short glossary of commonly used technical terms of cryptography and PKI, as well as a short introduction to the topic on a rather easily intelligible level, but with links to more detailed information. A nice feature would for example be an integrated FAQ section. While it is common practice to refer the user to the vendor's web site for further help using hyperlinks in dialogues, we vote against it. For the sake of convenience this mechanism should only be used for non-critical, additional, and up-to-date information. But not for answering standard requests since a user may not be connected to the Internet at the moment a question arises. We consider it a must to have context-sensitive help in all security-related menus and dialogues, at least in the most critical ones. The central questions touch the same areas (P2, P4, P5) in the current category than in the previous one.

- Which of the main technical terms of PKI and cryptography are explained? (e.g. certificate, key pair, certification authority) Are the explanations correct and intelligible?
- How detailed is the introduction to PKI and its concepts? Is the text written in an abstract fashion or are there useful examples?
- Is a context-sensitive direct help implemented? If yes, in which way does it support the user? Are there further links to the help system?
- In which way is the user assisted when accomplishing a complex task? (e.g. when installing his own private key)
- Are there links to external information sources about PKI and current developments?

Reaction in Potentially Threatening Situations In order to get a reasonable measure on the utility of a PKI-enabled application, we have to test its reaction in critical situations. We consider this worth a separate category due to the potentially high damage that may be caused in case of failure. A typical scenario is a digitally signed message which has been tampered with. Then the application has to reject the document and explain in detail not only the reason of failure, but give a concrete guidance which measures the user has to take next. In case the local key or certificate store is manipulated, the application must react accordingly and trigger an alert. It is also important how status information is handled. Suppose, the application wants to use an X.509 certificate

which specifies the URL of a CRL distribution point or an OCSP responder, but the corresponding server is unreachable.

- What is the behaviour of the application in case of a certificate that cannot be validated successfully due to an unknown issuer?
- Does the application tell the user how to check the fingerprint of a certificate when introducing a new CA to the system?
- How does the program react in case when a source of status information specified in the certificate is unreachable?
- Does the application has the ability to automatically check for and download security updates and patches?

3.5 Security Features

The subject of the third and last category group are the security features of the PKI-enabled application under evaluation. These features relate indirectly to the issue of usability as decisions on the technical layer often have an influence on UI design. If, for instance, the application does not implement functionality which is necessary for using PKI in a certain environment or does not support a common standard, users are urged to find a way to work around this problem (probably weakening the system). Security on the cryptographic layer and utility of a PKI-enabled product are thus crucial factors for user acceptance, too. With regard to properties P4 and P5, we emphasize that the application should provide a proper implementation of the security mechanisms since it is virtually impossible for the user to detect flaws on this layer.

The arrangement of the following categories is relatively straight-forward: First of all, we review *algorithms and parameters* which belong to the cryptographic layer. The next category revolves around the *handling of secret keys*. This will lead us to the *certificate management* and how the application deals with *status information*. A last category is named *advanced functionality*.

Algorithms and Parameters In order to achieve compatibility towards different communication partners, an appropriate set of cryptographic algorithms should be available. We emphasize that a reasonable selection of methods should include standard and widely deployed algorithms. This does not call for as many as possible and moreover exotic algorithms as this may bear additional risks (at least to confuse the user). However, it is important that the cryptographic building blocks rely on independent mathematical problems, e.g. integer factorisation and discrete logarithm in an elliptic curve (EC) group. This is because of the observation that mathematical or technical progress may render a single algorithm insecure. The cryptography inside the application cannot be treated separately since its effectiveness also depends on the protocols built on top of it. Thus, the word algorithms extends to protocols in this section. The second aspect in this context are the cryptographic parameters, typically including key sizes and lengths of message digests.

- How many different combinations of cryptographic algorithms and parameters are available?
- Does the selection comprise state-of-the-art algorithms? (e.g. AES, RIPEMD, EC cryptography)
- Is it possible to choose individually from the set of combinations of algorithms and parameters? Does the system warn against weak combinations?
- Are all weak combinations disabled in the default settings?
- Are cryptographic primitives based on independent mathematical problems?
- Does the application implement a strong pseudo-random number generator? What source of random is used as seed?

Secret Key Handling This category is about how secret keys are generated, protected, applied, transferred, and destroyed by the application. In this paragraph, we use the term secret key in a generic way. While it is often replaced by the term private key in the context of a public key scheme, a secret key in a narrower sense is used for a symmetric algorithm, e.g. as an ephemeral session key. It goes without saying, that secret keys have to be protected carefully by versatile technical precautions.

- Is the application able to generate keys and key pairs on behalf of the user?
- What kind of storage devices for secret keys are supported? (e.g. smart card with corresponding reader, USB token, HSM, or soft token)
- Which cryptographic and other technical precautions are provided to secure a secret key when stored on the hard disc or in memory?
- How is a secret key enabled for usage? Is it possible to enable a batch processing? (e.g. for bulk signature generation)
- In case a secret key is exportable: Are there efforts to enforce a certain password complexity when using password-based encryption?
- Can key data be securely destroyed, e.g. by multiply over-writing the corresponding area on the hard disc?

Certificate Management We now turn to the management of third party certificates. Two things are important in this context: In our opinion, the application should first support the import of certificates via different channels to facilitate this task for the user. This means that a lookup on a certificate server or – in the PGP terminology – *key server* should be supported, using different protocols like LDAP, HTTP, FTP, or simply retrieving the certificate from the file system. The second question is how the integrity of the certificate database is guaranteed. If the certificates are stored on the hard disc, a typical method is to use a password-based MAC or a central storage location to ensure that the database has not been tampered with.

- How does the import and retrieval of certificates take place?
- Is it possible to read/write a certificate from/to a hard token?
- Does the application cope with certificate trust lists (CTL) for import or export?

- Which technical measures are taken to protect the certificate store?
- How can the user assign and withdraw trust to third party certificates? Is this possible on a fine-grained basis?
- Can the certificate handling be delegated to a central point or authority? (e.g. the OS or an administrator)

Status Information Especially with regard to signature verification in the context of non-repudiation, the ability to gather information about the validity of a certificate at a certain point of time is crucial. It is also important to guarantee a timely revocation of public keys used for encryption. However, status information often is not retrieved automatically or processed properly by PKI-enabled applications. As of today, different methods are available to answer a validation request. We point out that the answer to such a request is nontrivial, as there are different validity models.

- Does the PKI-enabled application automatically gather status information?
- Which mechanisms for status information are supported? (e.g. CRL, delta CRL, indirect CRL, online requests via OCSP)
- Does the application make use of a CRL distribution point entry in an X.509 certificate?
- Does the application allow to manually import status information obtained from another source?
- Is there a mechanism to archive validation material?
- Does the application ensure that the machine's local time is correct?

Advanced Functionality Finally, we come up with some additional features. While some of these features are well known in the community, they are often not implemented by current PKI-enabled applications. Particular use cases, like for instance the processing of signed documents which is subject to digital signature legislation, may have very special requirements. In environments with high security demands, e.g. for governmental use, a certain level of security evaluation may be an absolute must.

- May an administrator enforce a group policy concerning e.g. key export or maximal usage periods for cryptographic keys?
- Is the application ready for enhanced concepts like cross-certification or bridge CAs? Are time stamping services supported?
- In case the application processes X.509 certificates: Which extensions are implemented, are critical extensions always handled correctly? Does the application require proprietary extensions? Is the application compliant to major standards? (e.g. the PKIX or ISIS-MTT profile)
- Does the application support key sharing/backup? (e.g. à la Shamir)
- Has the application undergone an evaluation of its security features? (e.g. according to the Common Criteria)

4 Conclusions

We have presented a new and generic framework to evaluate the usefulness of PKI-enabled applications. It is not restricted solely to usability, but also treats deployment issues and security features to get utility ratings, too. Our framework can be adapted to a multitude of PKI use cases by means of a usability profile. Due to space restrictions, we could not give a sample profile here.

We designed this framework having the high PKI-inherent complexity in mind which requires special attention. However, the framework could be extended to cover general security applications as well.

Acknowledgements

We received helpful critique from the anonymous reviewers. We also thank Gerold Hübner (Microsoft Deutschland GmbH) for fruitful discussions on the topic.

References

1. A. Adams and M. A. Sasse. Users are Not the Enemy: Why Users Compromise Security Mechanisms and How to Take Remedial Measures. *Communications of the ACM* 42 (12), 1999, 41–46.
2. J. Buchmann, H. Baier, and T. Straub. Absicherung von Anwendungen mit der Unterstützung von Public-Key-Infrastrukturen – Benutzbarkeitsstudie im Auftrag der Microsoft Deutschland GmbH, 2003. (in German)
3. D. Davis. Compliance Defects in Public-Key Cryptography. *6th USENIX Security Symposium*, San Jose, USA, 1996.
4. D. Gerd tom Markotten and J. Kaiser. Usable Security – challenges and model for e-commerce systems, *Wirtschaftsinformatik* (6), 2000, 531–538.
5. U. Holmström. User-centered design of security software. *17th International Symposium on Human Factors in Telecommunications*, Copenhagen, Denmark, 1999.
6. ISO 15408: Common Criteria for Information Technology Security Evaluation (CC) Version 2.0, 1998.
7. ISO 9241: Ergonomic requirements for office work with visual display terminals.
8. J. Kaiser and M. Reichenbach. Evaluating security tools towards usable security. *IFIP 17th World Computer Congress*, Montreal, Canada, 2002.
9. J. Nielsen. Usability Engineering. AP Professional, Cambridge, 1993.
10. M. A. Sasse. Computer Security: Anatomy of a Usability Disaster, and a Plan for Recovery. *CHI 2003, Workshop on Human-Computer Interaction and Security Systems*, Fort Lauderdale, USA, 2003.
11. B. Schneier. Secrets and Lies, *Wiley*, 2000.
12. J. Voßbein and R. Voßbein. KES/KPMG-Sicherheitsstudie: Lagebericht zur IT-Sicherheit. *kes* 3 and 4, 2002, available online http://www.kes.info. (in German)
13. A. Whitten and J. D. Tygar. Why Johnny Can't Encrpyt: A Usability Evaluation of PGP 5.0. *8th USENIX Security Symposium*, Washington DC, USA, 1999.