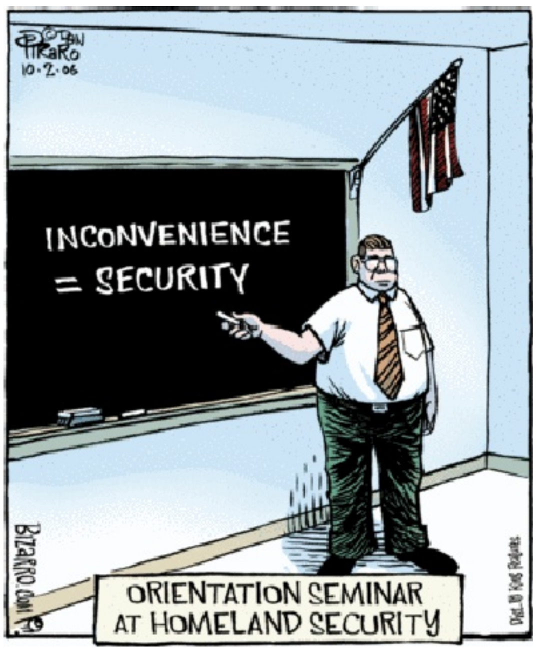
Usability of authentication in web applications – a literature review

Anders Bjerg Pedersen (andersbp@me.com)

July 8, 2010

Abstract: Since Saltzer and Schroeder in 1975 recognised the importance of usability in security applications, research on the subject has been limited. In this review we investigate the interaction between usability and security in software products, with specific attention put on authentication on the web. We describe the typology of users and their perception towards security, compare various current authentication methods, and list and evaluate different sets of usability guidelines and principles for secure interaction design. On the basis of user-centered design we outline current development methods adapted for secure systems design and have a look at evaluation methods and the usability studies employing them. Finally, we propose suggestions for further work in the area.



Contents

1 Users, perception, and trust **4**
1.1 The users of online authentication 4
1.2 Users' perceptions of security 6
1.3 Mental models and trust 8
1.4 A few notes on privacy 11

2 Authentication methods **11**
2.1 Passwords 12
2.2 Public key infrastructures (PKI) 13
2.3 Two-factor authentication 15
2.4 Other authentication methods 16

3 Usability factors **16**
3.1 Definitions of usability and guidelines 17
3.2 User-centered security and design 19

4 Developing usable security **22**

5 Testing usability of security **25**
5.1 Evaluation methods 26
5.2 Usability studies 27

6 Conclusions and further directions **29**

References **31**

Introduction

"Security systems typically attempt to introduce barriers (such as passwords or other authentication mechanisms) while HCI designers attempt to remove such barriers."

Dourish et al. [11], p3

The above quote is very meaning to the primary reason for conducting this review: security and usability in software development are two important factors that until now have been unable to cooperate in the same development process. They have always been seen by both research communities as inconsistent with each other, as their individual goals present obstacles for one another. Since 1975 it has been proposed that usability is an important part of secure software design, but research seems to have largely ignored this observation, leading to flawed software designs that radiate the lack of interoperability between the two parties.

In this literature review we challenge the above statement and try to find evidence of principles, methods, and tools that aid the simultaneous coexistence and cooperation of usability and security in the never-ending strive to create better usability, more clarity of concepts, and fewer errors for the end-users. Our focus is primarily on web-based authentication systems, but we take a more general approach to the literature and outline consequences for authentication where relevant.

The first section deals with describing and understanding the users of online authentication, their perceptions towards security as well as discussing notions of risk, trust, and privacy in this context. Section 2 goes along a different path, as we investigate three of the most common authentication methods used on the internet today: passwords, public key infrastructures, and multi-factor authentication, providing references and case studies as we go along. Section 3 deals with those usability factors important when designing usable security. In section 4 we investigate current methods and practices for developing usable security software, and in section 5 we do the same for evaluation methods and frameworks for testing the usability of finished products. We conclude the review by outlining recommendations for further research and development.

In order to benefit from this review, some basic knowledge of human-computer interaction (HCI) as well as authentication techniques will be helpful but not required. An excellent and broad introduction to most of the topics covered in this review can be found in the comprehensive collection edited by Cranor and Garfinkle [6].

1 Users, perception, and trust

This first section deals primarily with the users of online authentication. In order to establish development and testing methods as well as to design usable software in general, one must investigate the end-users of the product being developed and their perception towards the concepts on which the software builds. A car manufacturer not having some knowledge of both the customers as well as the rules and tasks they face is not going to be able to produce comfortable or practical cars fit for the users' needs. We also touch on the notions of risk, trust and privacy and discuss some properties of the mental models associated with security.

1.1 The users of online authentication

Every student attending a usability course will have been told again and again to examine the users of the product being developed at a very early stage in the design process. The focus of this review is those authentication methods that regular people use on a daily basis, so who are these users then? Initially, we would state that our user domain is populated with every type of person imaginable, of every age, sex, educational level and so forth. Without a thorough insight into the characteristics of the users utilising a system, the design is potentially going to be confusing or mis-interpretable, and the users might decide not to use it or use it in the wrong way. The latter is a huge problem when dealing with security systems, e.g. authentication.

We are, in other words, interested in a form of classification or *typology* of our target audience. Albeit the study being of some age, Sheehan [30] tries to create such a typology of American internet users by comparing federal and commercial surveys, as well as conducting his own survey. His starting point is an existing segmentation of users into varying degrees of privacy concerns:

- *Fundamentalists* (or *alarmed* users) are very wary against exposing too much privacy and using new services at the cost of sharing personal information. They often employ active measures to shield their identity (e.g. entering incomplete or false information). In the US, this group is estimated to constitute 25% of internet users.
- *The unconcerned* are at the opposite end of the scale, having high trust in their information being handled properly. These individuals are happy to sign up and use new programmes and services, if they can benefit from doing so. This group also constitutes around 25% of the internet population.
- In the middle we have the *pragmatists*. This roughly 50% share of the population continually weighs the benefits of using services against the information required to do so. They weigh the risks and benefits and adapt their behaviour accordingly.

The fear of exposing too much privacy is closely related to the concept of *risk*, defined for instance by the Oxford Online Dictionary as *"a situation involving exposure to danger"*. The danger here being the overexposure or even disclosure of information that may be regarded as sensitive to the user, in close relation to the notion of *privacy*, which we shall investigate in more detail later on.

Sheehan explores five different privacy influences: *awareness of data collection* (is my data being stored by this entity?), *information usage* (what is my personal data used for and by whom?), *information sensivity* (what information is sensitive to me?), *familiarity with entity* (do I know this entity?), and *compensation* (what do I get in return for sharing this information?). Using 15 situations in these influences, a *total concern* figure is calculated. The results show a need to segment the pragmatists into two separate groups: the *circumspect* and the *wary* users. The survey shows a differently-sized segmentation than the above one, with fundamentalists only constituting 3%, the pragmatists 81%, and the unconcerned 16%. This naturally suggests that internet users are gaining more trust in the online applications they use. More, albeit somewhat vague, observations regarding the demographics of the users are summarised in Figure 1. Not surprisingly, a final conclusion of the study is that the user's concern varies according to context – corresponding well with the high percentage of pragmatists.

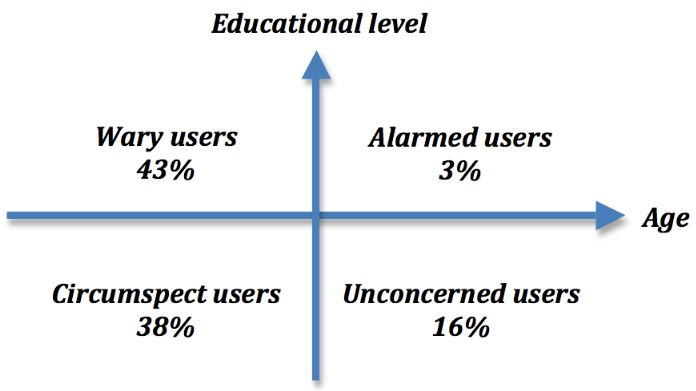


Figure 1: Online user typology of privacy concerns, adapted from Fig. 1 in Sheehan [30].

Although Sheehan focuses on the broad internet population, it is fairly safe to assume that this population is also comparable to the one using authentication methods on a regular basis. The segmentation in four different groups makes good sense and should be on the mind of every software developer at an early stage – especially when developing publicly available authentication systems for accessing highly sensitive information such as tax papers, medical journals, or public administration systems. These systems are meant to relieve government administration and make it easier for the user to interact with public sectors, but they fail miserably if they do not succeed in being trusted and widespread across all of the internet population. Focusing on the circumspect and unconcerned users is not enough – good usability is required to gain trust and minimise concern in the remaining population.

The aspect of educational background is also interesting, due to the common agreement that *good security is invisible*. One might argue that the higher the education, the more details of the authentication process can be comprehended. As this may have some truth to it, Balfanz et al. [3] showed that even highly-educated people with PhDs in computer science can have immense problems applying security methods. We shall return to this in section 2.2. So, in order to accomplish usable security that is widely accepted, developers have to consider the typology of the users planning to use the system, including factors such as concern, age, and educational level. With respect to the latter, the lowest common divisor could be used as a measure for usability (if *they* can't use it, no one should).

1.2 Users' perceptions of security

Simply knowing *who* the users are is not enough to be able to develop usable authentication. There is also a need to investigate users' perceptions of the *idea* of security itself. In the previous section we have established a rough estimate of how concerned users are in general, however, we also need to investigate how users perceive the applications they are presented to: are they trustworthy, what is their function, is the connection secure, is the current security level adequate for my task?, etc.

In an interesting study by Friedman et al. [14], the authors investigate users' abilities to describe, recognise, and visualise secure connections on the web. They conduct their survey on subjects from three different communities (high-technology, suburban, and rural) and observe that not surprisingly the high-technology community was better at defining a secure connection, although not much better at actually recognising one on a webpage. They were quite a lot better at recognising non-secure connections, however, and these judgments were primarily based on six different types of evidence on the actual web pages. They conclude that *"...the high-technology participants did not always have more accurate or sophisticated conceptions of Web security than did their rural and suburban counterparts"*.

A similar and more recent study by Whalen and Inkpen [36] finds that when spoofing a well-known home-banking site, only 1 out of 16 participants correctly observe that the site is not secure. The study itself is focused on browser measures for visualising security of different sites; something we shall also have a closer look at in section 4.

The latter two surveys are interesting for the fact that they are in good harmony with the findings from section 1.1. Although we are aiming certain security features at certain population groups, e.g. professionals, they might just as well misjudge the security concepts, allowing hackers and eavesdroppers to target a wider segment of the internet population.

Dourish et al. [10] agree in a preliminary paper that security and usability have to go hand in hand. However, they see it as *"an achievement of people interacting with technology"*, and current problems are seen to be caused by the software in question not providing enough information to allow users to make informed decisions. This is closely related to what the above two surveys conclude, although the authors disagree that invisible or *transparent* security is the best form of security. Rather, they argue that the technologies should be highly visible, immediately available, and expressed in terms that fit the user's normal tasks and needs. (Dourish et al. [11], p21) This is of course not in complete opposition with transparent security, as visibility of security concerns both assuring the user that his or her transactions are secure by showing some form of confirmation of the secure state, and the immediate hiding of unnecessary and complex information about protocols, certificates and the like from the user in order to avoid confusion.

The principle of *delegation* is also introduced: as users are incapable of constantly keeping themselves updated on new security software and issues, it can be fruitful for the individual user to delegate this responsibility to other people, companies, or organisations – effectively making it ”someone else's problem”. They observe four types of delegation (more thoroughly defined in Dourish et al. [11]):

- *Delegating to technology* implies transferring trust to some technological means of securing data and communication. This could for instance mean that if users see the small padlock in the bottom of the browser, they are assured that some technology (such as SSL or HTTPS) is protecting them from harm.
- *Delegating to another individual* is perhaps best illustrated by an example: I have always maintained most of my parents' computers, and having their son (who happens to be a computer science student) set up home-banking or firewalls has always been a completely trusted delegation of security.
- *Delegation to an organisation* could for instance be to the IT department at work. They as a group are trusted to know what they are dealing with, but they are often not known on a personal level as opposed to delegation to another individual.
- *Delegation to institutions* refers to many users' firm belief that when dealing with for instance their bank, the concept of vaults and security guards transfers on to the online business of the bank – regardless of the fact that the technologies used are outside of the bank's control.

Delegation is quite an interesting factor when designing new security software: who do we want the user to delegate trust to? Preferably, we would want the user to delegate it to the institution or company employing the software – the ultimate combination of reputation, technology, design, and trustworthiness. But what does it take for an entity to be trusted? With the advent of phishing and spoofing, the users are constantly faced with decisions on whether or not to trust certain sites or services. Also, as security gets more and more complex, users are presented with certificate approval dialogues, spam and spyware filters, phishing attempts, and many other decisions from an increasing amount of sources: the operating system (user authentication, setup of firewalls, downloaded files), web browsers (phishing attempts, web site security), email clients (spam filters), and perhaps other third-party applications. Most of these entities allow delegation of such decisions.

Dourish et al. [11] attempt to understand how users see security in daily usage. Initially they observe that security is often seen as a barrier for completing the tasks set out to do. If some applications deal with only certain security issues and not others, the are perceived as being inadequate or even insecure by some users. On the other hand, some users falsely believe that a virus scanner is protecting them from for instance phishing.

They also observe three different attitudes towards security; *frustration* that security is something to be dealt with but doesn't really contribute to anything and sometimes even hinders the task at hand; *pragmatism* (which resembles the definitions by Sheehan [30]); and *futility*, the feeling of never being on top of things and constantly having to keep up with hackers and phishers.

Many users have been observed applying special methods to enforce security, effectively circumventing the technical measures instated to do so. Actions like appending legal notices to emails and thereby outlining the consequences of misusing the contents instead of encrypting them, as well as switching to e.g. telephone for the most secure conversations (although the telephone is far from secure, to put it mildly). This, in our opinion, indicates a great distrust in and inefficiency of many security applications. After all, encryption of emails was meant to be a faster and more secure way of communicating sensitive material than switching to a telephone or having to write emails in some "encoded" office dialect. These are examples of poorly-designed or hard-to-use security systems that the users would rather avoid or even circumvent than spend time to learn using – sometimes creating even worse security than before the systems were introduced.

1.3 Mental models and trust

A previously much-acclaimed term in HCI is the notion of *mental models* – an image of how the world around oneself works, how to interact with it, and how to predict the consequences of these interactions. Or, as Dourish and Redmiles put it: *"The conceptual understandings that users hold of the domains in which they operate. Actions are planned and interpreted with respect to these models."* ([9], p78) Although to many researchers this term is considered slightly "fluffy", it is still of some relevance to relate it to the use of security applications, since these utilise extremely complex technologies underneath the image presented to the user. The notion of *objects* and *actions* play an important role in these models, and they can be closely related to the language constructs nouns and verbs, respectively.

Yee [40] uses this form of mental model to better understand the user's perception of security.[1] An important factor here is the notion of different types of *stances*: a user's mental model can adopt a certain stance towards an object or action based on the context it is in and the expectations of the user. For instance, the author gives the example of a cup being pushed off the side of a table – we naturally *expect* it to fall down, hereby adopting a *physical stance* (applying physical laws and rationale to deduct consequences of actions). In the context of computing, when we open an image file, we expect the computer to open and display it; not carry out some other action.

Some objects also perform actions and are therefore seen as *actors*. Whether it being other human beings or computer programs, we can no longer model their behaviour simply by adopting the physical stance. On a computer program we normally adopt the *design stance* (understanding what the program was designed to do), and on other users we apply the *intentional stance* (trying to figure out their actions based on their motivation and intention). Relying on evidence from previous studies showing that users also see computers as social actors, the author concludes that

> *"A system is secure from a given user's perspective if the set of actions that each actor can do are bounded by what the user believes it can do."*
>
> Yee [40], section 3.2.

So as a simplified example, if a user is signing up for the annual boy scout convention via a secure site using authentication and believes that every other boy scout using the site has only the possibility of doing so in their own name, the user sees this system as

[1] In our opinion, the word "perception" is actually a better term for these kinds of "models".

secure.[2] It is an interesting definition, as it does not say anything about *how* secure the system has to be, what measures need to be taken, and what actions are available to the user – it is indeed very general.

The perception of security as expressed in stances is also interesting in connection with authentication: which other actors have access to the same information, and how does this influence me as a user? Indeed, one could propose that the ultimate goal of authentication would be to offer users such transparency and usability that nothing other than the physical stance would be needed in order to interpret the consequences of actions committed to the system. If the stance taken is not in harmony with the actual behaviour of the system, the user is prone to make errors. This is in good harmony with the notion of *psychological acceptability*, which is broadly considered as the first mention of usability in conjunction with security:

> *"It is essential that the human interface be designed for ease of use, so that users routinely and automatically apply the protection mechanisms correctly. Also, to the extent that the user's mental image of his protection goals matches the mechanisms he must use, mistakes will be minimized. If he must translate his image of his protection needs into a radically different specification language, he will make errors."*
>
> Saltzer and Schroeder [28], p5

This definition is from 1975, a time in which research on usability and security was not even thought of. As HCI researchers have developed standards for browsing the web,[3] users have adopted these standards and, more importantly, expectations towards what these standards supply and how.

In an interesting survey of literature on trust in internet applications, Grandison and Sloman [15] cite various definitions of the concept of *trust* and then create their own in relation to internet applications, stating that trust is *"...the firm belief in the competence of an entity to act dependably, securely and reliably within a specified context"* [15, p4]. So, in the context of package delivery, we have a firm belief that companies such as DHL or UPS deliver our packages (they *act*) on time, to the correct receiver, and in such a manner that no-one but the sender or receiver knows about the contents of the package. The authors continue on to define trust relationships as being for instance one-to-one (I trust my best friend in not stealing my girlfriend) or many-to-one (the customers of a bank all put their trust in their savings being well protected by that particular bank).

Also, various levels and types of trust are identified. Of particular interest is the *provision of service by the trustee* and the *certification of trustees* by third-parties. The former deals with putting trust in some service offered on the web, be it email services or home-banking; the latter concerns trusting auxiliary entities to verify that the entity providing a service is trustworthy. This can be seen in most modern web browsers when clicking on the padlock to see the verification of a certain site by for instance VeriSign.

[2]Leaving out the massive amount of other actors: the site administrator, potential hackers and eavesdroppers, etc.

[3]Actually, that specific achievement (manifested by the development of Mosaic – one of the first graphical browsers for the web which is credited for leading to the Internet boom of the 1990's) is widely seen as one of the greatest in HCI research so far, only topped by the invention of the Graphical User Interface (GUI).

Also, delegation (see page 7) is naturally considered a form of trust.

The concept of trust is perhaps the most important factor in designing usable secure systems. A lot of people today are still hesitant of using home-banking services; they would still rather put their trust in the cashier at the bank than paying their bills electronically from the comfort of their home. As we saw in section 1.1, though, the alarmed users are more present in the older population, thereby inferring a reasonable assumption that they may require longer to lay aside old habits and put their trust in new services online. However, usability still plays an important role in online authentication systems gaining the trust of the broad population of internet users. An unprofessionally designed and complex site does not emit enough trustworthiness to be accepted and used. Trust also depends on the entity at hand. A well-known bank is clearly trusted to handle financial transactions on behalf of the user and is therefore also trusted to handle the personal information required to perform such transactions online. On the other hand, the local sports club's initiative of enabling subscription payments online from a site designed by one of its little league players may not obtain the same level of trust.

With respect to transparency it should be possible in some easy way to see what information the user is allowing access to when authenticating him- or herself to a service. A very recent example is Facebook. In the media, Facebook has been heavily criticised for making it difficult for users to manage their privacy and for possibly giving away or selling personal information entered by the users. Considering the amount and nature of information submitted by the users, Facebook should be considered *highly* trustworthy by its almost half a *billion* users. It is noteworthy, in relation to our discussion in section 2.1, that Facebook only requires an email and a simple password to gain full access to this often very private information.

Another example of transparency and trust is the Danish auction and buy/sell site Den Blå Avis (DBA), recently acquired by eBay. DBA allows its users to place sales ads on all kinds of used goods. As an additional feature they allow sellers to have their personal information verified by means of entering their social security number. This certification is then displayed on all the seller's ads and puts DBA in a special position of being both a service provider that is to be trusted to handle financial transactions (sellers can pay for several extras in their ads) as well as being a form of certification authority for sellers, effectively increasing their trustworthiness. However, contrary to Facebook, DBA is very clear on its handling of a user's social security number, which to many people in Denmark is considered the pinnacle of sensitive information.

An interesting example of another, albeit slightly more mathematical and general, view on trust can be found in Jøsang and Presti [21]. Here the authors propose mathematical/economical models of the relation between risk and trust – models for evaluating the expected gain of certain transactions, the probability of success, and other factors. Although not completely relevant for our topic or even applicable for regular users, it still makes for an interesting solution to the problem: *"Is this service offering me enough in return for investing my trust and information?"*

Also, an easy-readable and thorough chapter on trustworthy design including an overview of lots of trust models by Patrick et.al. can be found in Cranor and Garfinkle [6] (Chapter 5).

1.4 A few notes on privacy

We haven't yet directly handled the notion of *privacy*, as can be defined for instance by Wikipedia.org as *"the ability of an individual or group to seclude themselves or information about themselves and thereby reveal themselves selectively."*[4] Naturally, though, privacy is tied closely together with trust and authentication: when we authenticate ourselves on the web, we reveal some form of identity towards the entity in mention, which then presents us of some form of personalised service or even allows us access to stored and sometimes personal information about ourselves. In each case the user has to decide whether to give information and to what extent – a decision process that relies heavily on trust in the entity and an evaluation of the risks associated with supplying the information required. Privacy on the web is therefore highly associated with keeping information and transactions secret and restricted to certain users and entities – a process that naturally requires authentication. For an indeed *very* thorough survey of literature on privacy in HCI (including a massive 315 references), we refer to Iachello and Hong [18].

Although the discussion of privacy is of course relevant in any software context, we shall not delve much deeper into it here. However, we find it worth mentioning an initiative by the World Wide Web Consortium (W3C) called P3P – Platform for Privacy Preferences.[5] A promising protocol implementing privacy certificates issued to websites that require information about the user, containing details about what types of information and for what reasons and possible use it is collected (also referred to as a *statement of privacy policy*). A similar reverse policy is set up by each user deciding which information to share and to what extent. When entering a site that requires more information than the user's policy allows, a warning is issued to the user. Sadly, this protocol has never really gained momentum and has lived a quiet life since 2007, being held back by software vendors' unwillingness to support the protocol. In our opinion, however, this is a clever way of implementing reusable privacy settings on the web.

As a final note on privacy, I myself recently discovered that I was now apparently running my own business. My personal website was listed on an otherwise well-renowned business-finder website, even though I had never requested anybody to do so. This was an obvious abuse of the above definition of privacy, as I was not able to isolate the information available about me or the use of it, and a clear warning to sometimes monitor that the information you make available in one place (or publicly for that matter) is not misused in places you had not foreseen. Privacy in a nutshell.

2 Authentication methods

In this section we briefly outline different authentication methods currently employed on the Web. We present three common methods of authentication and discuss their advantages and disadvantages in terms of both security and various points of interest concerning usabillity factors. Specifically, we have a look at passwords and how to create and memorise these, public key cryptography (with focus on Diffie-Hellman and RSA), and lastly two-factor authentication. We conclude this section by outlining other authentication methods that are not commonly in use on the internet. For a thorough yet fairly

[4]See: http://en.wikipedia.org/wiki/Privacy
[5]See: http://www.w3.org/P3P/

simple introduction to many authentication methods (including the below mentioned ones), techniques, and means of attacking them, we refer to Smith [32].

2.1 Passwords

Passwords should require no thorough introduction to most readers. They are used in a multitude of situations and are by far the most widely-employed method of authentication. Be it logging on to one's own PC, using various web-based services (webmail, home-banking, various profiles for blogs, forums, etc.), or even using your credit card in the local grocery store all utilise passwords for unlocking features and authenticating yourself to a machine or service.

Although passwords are in such common and daily use, there are a number of factors influencing the usability and security of passwords. Who hasn't been given the usual lecture on how to create good passwords and a bunch of advice on not to share or write down passwords, not reuse them for different services and so on. We shall have a look at some of this advice, and see how whether users are following it or not can have a serious effect on both security and usability.

The first obvious factor to look at is the length of the password. Most people can agree on the fact that the longer the password, the better the protection of the service it grants access to. So why don't all people just not employ passwords of, say, 512 characters? Three apparent reasons are obvious: the time it takes to input the password naturally grows as the length grows, there is a substantially larger margin for typing errors, and very few people are capable of memorising passwords of such lengths. In fact, it is interesting to investigate the "optimal password length" – how many characters or numbers can a normal person keep in memory?

Miller [24] investigates this more generally. He collects results of various experiments measuring test subjects' abilities to memorise different situations, numbers and even sensory experiences. For instance, in one experiment a test person is shown a pattern of dots for a very short time (merely a flash) and is afterwards asked how many dots were in the image. Interestingly, all images containing up to six dots were counted correctly by all test subjects, whereas going to seven and past made them estimate and therefore often miscalculate the number of dots. Even more interesting, other completely different experiments show a similar "sweet spot" somewhere around "the magical number seven". Miller's experiments are primarily focused on short-term memory, though, and he recognises that the ability to remember a far greater number of characters or entities in general dramatically increases as we for instance increase the number of attributes attached to the objects to remember (face recognition, for instance, involves both the eyes, nose, hair colour, etc.). So, for our immediate interest in passwords which have very few unique characteristics the optimal length of a password that it memorisable by the user is somewhere around seven. Actually, many web-based services require that new passwords be between six and eight characters in length.

Simply constructing random passwords of length seven is not the best solution for the user, though. Yan et al. [38] investigate the tradeoff between using an easy-to-remember password that is often weak against common brute-force or dictionary attacks, and a complex randomly-generated password with the opposite properties. They argue that complex passwords may compromise security because users are more likely to write them down or even put them on a piece of paper on their screen. They investigate passwords generated from different advice given and segment their subjects into three groups. Group A is asked to generate a password of at least seven characters that contains at least one number. Group B is given a matrix containing random characters and numbers and is asked to randomly select eight characters from the matrix, write them down and memorise them. Group C is asked to create s simple sentence of eight words and choose for instance the initial letter of each word as well as inserting a number or a special character somewhere in the password.

The results of the experiment show that the average length of the generated passwords was between seven and eight characters, and that attacks on group A's passwords successfully cracked around 30% of them while groups B and C both were well below 10%. Also, the difficulty and time consumption of memorisation in group B was significantly higher than groups A and C. The authors therefore conclude that mnemonic-based passwords are both more secure and easier to remember, that length matters, that special characters should be enforced, and that compliance with the password advice given should be enforced by the system. Otherwise users will ignore that advice and select passwords more susceptible to attacks.

In another study, Adams and Sasse [2] conduct a questionnaire on users' behaviours and perceptions with respect to password-based systems. Like Yan et al., they also find that users often come up with their own procedures and methods for generating and memorising passwords. Also, users are generally not well enough informed on how to generate secure passwords, but introducing restrictions is likely to lead to a greater degree of password disclosure. The lack of security information on security issues is also apparent and causes users to misconceive the importance of for instance password generation and disclosure. If a user cannot see a threat, it is perceived as not being present. Another misunderstood burden on the user is the fact that not only the password is seen as something that is to be kept safe and memorised but also the user's id (e.g. username or other), effectively causing the user to spread the complexity of memorisation on both instead of focusing on making the password more secure. The authors argue that more and better security information needs to be given to the user, and that design of the interface plays an important role in this task. In general, developers of security systems need to take the user more into account when designing systems in order to make them reflect the user's daily work practises. A similar study can be found in Zviran and Haga [43].

So, to sum up, the road to successfully generating and using passwords is more curved than simply writing two lines of advice next to a login prompt. Considering both the user and the security of the system at hand, mnemonic-based passwords of length seven or eight containing some special characters that are enforced by the system and whose guidelines and security implications are meaningfully presented to the user, are optimal in most respects.

2.2 Public key infrastructures (PKI)

Back in the days when digital communication was being broadly adopted to also handle more sensitive business transactions and communication, there was no better solution for encrypting information and authenticating than physically transporting a *key* used for encryption and decryption to the receiver of the message, either by courier or even via a dedicated, secure line directly between the parties involved. However, couriers are expensive and dedicated lines were slow and costly to build. So in order to achieve secure communication without having to wait a week for a courier to arrive, there was an imminent need for another solution. Diffie and Hellman [8] suggested the use of a so-called *public key cryptosystem*, where the keys used for encryption were publicly available.

The proposed scheme is actually not secure; it is well-known how to crack these types of encryption. Cryptographers, though, make a distinction between *unconditionally secure* systems, which can be proven to be secure against any form of attack, and *computationally secure* systems. The latter type is insecure only if infinite computational power is available. The reason why public key cryptosystems are adequately secure is that they rely on mathematical properties of *one-way functions*. A bijective one-way function f is easy to apply to some argument x, and thereby obtaining a result $y = f(x)$, but finding x when you only know y (e.g. computing $x = f^{-1}(y)$) is computationally infeasible. Making the domain of x equal to a binary representation of some alphanumeric character set allows us to apply these functions to for instance emails. Diffie and Hellman suggest various such functions and note that promising results are expected using algebraic exponentiation. In fact, two years later Rivest et al. [27] proposed using these mathematical properties for designing a specific scheme, named after the surnames of the authors: RSA. This is now almost the de facto standard for secure digital communication, and is based on the fact that factoring the product of two very large prime numbers is computationally infeasible. An overview and reasons for needing newer more secure cryptosystems at the time can be found in Hellman [16].

The general and most widely adopted method for using public key cryptosystems is sketched in Figure 2. Both Alice and Bob each have a unique private key (the function f^{-1}) and a public key (the function f), and have made their public keys available, for instance on a public key server or simply by sending them to each other. Bob wants to send the message M encrypted to Alice via email. He applies Alice's public key to the message, hereby obtaining the encrypted version $C(M)$ which he sends to Alice. Alice now uses her secret private key to decrypt $C(M)$, obtaining the original message M. A similar approach is used when authenticating using public key infrastructures.

Although the method in Figure 2 seems fairly simple, in practice it often is not. Another PKI method used for authentication is the X.509[6] standard. Balfanz et al. [3] used X.509 to set up a wireless network at Palo Alto Research Center and decided to investigate users' ability to set up and use the system. Contrary to expectations, the results were shocking: setup took 140 minutes per user on average, and despite utilising various GUI wizards the setup process still required a massive 38 steps, each requiring a decision or action from the user who typically had a PhD in computer science. The authors conclude from their investigations that retrofitting a GUI to a complex process is not the way to go – usability and security must be thought of simultaneously and at an early stage in the development process. Also, there is a need for high-level tools for designers to improve usability, and they conclude (as Adams and Sasse did) that the design must match the user's perception of the system and its features and not try to

[6]See for instance: http://www.infosecurity.org.cn/content/pki_pmi/x509v4.pdf

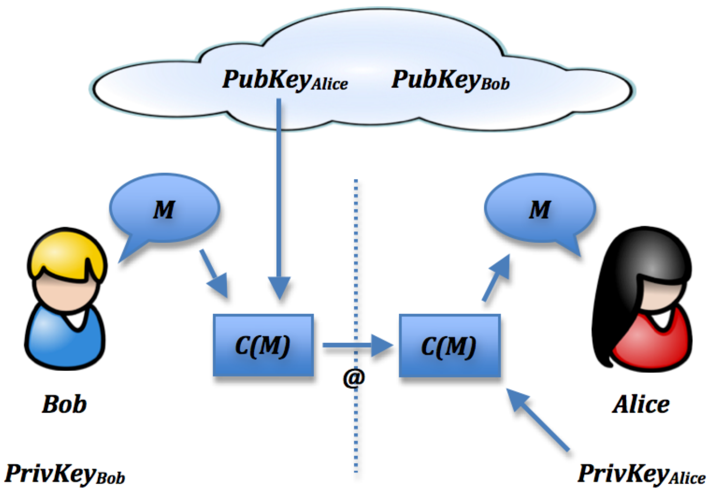


Figure 2: Sketch of most common PKI method for encrypting messages.

solve security issues in the low layers of the system. They need to be brought up front. A classical study of a PKI implementation can be found in Whitten and Tygar [37], which we shall return to in section 5.

In section 5 we shall have a look at some possible definitions of criteria for developing and usability testing public key infrastructures. As a note on security, a PKI is of course only secure if the private key is kept secret. Exposing it enables hackers and the like to impersonate and act on behalf of the owner of the key.

2.3 Two-factor authentication

Despite public key cryptosystems being highly secure, they are not always suitable when considering usability or portability – installation is cumbersome and ties the security features to certain computers. Standard PKI methods are usually referred to as *single-factor authentication*[7]: you need only the private key (and perhaps a password to unlock it) to authenticate yourself. In an everlasting effort to develop more secure systems, the notion of *two-factor authentication* has come up (see [1]). It builds primarily on the concept of *something you have* and *something you know*. The thing to know could be a password, and the thing to have could be a PIN generator or a *smart card*. A smart card is a common name given to some device or object that embeds a certain simple functionality, allowing it to for instance generate codes based on the time of day or on some simple input. A common method is to issue each user a small keyring number generator which is in perfect sync with a precise clock on the server and paired with the id of the user. When authenticating with the server, the user inputs both a username, a password, and the code displayed on the keyring at that time (a form of one-time password that usually changes once a minute or so). Using 6-7 digits or more ensures that guessing is not a viable option.[8] The thing to have could also be physical, e.g. a fingerprint or an iris scan, or a small card containing some number of one-time keys.

[7]Actually, they lie somewhere between single and double (or two-factor) authentication.
[8]Another version using a plastic card and a 4-digit password is in widespread use today in every ATM on the planet.

In general, *multi-factor authentication* is considered more secure than the above mentioned methods, although many of them are in fact vulnerable to man-in-the-middle attacks. However, once issued they are much easier to use and are completely portable. A wealth of methods and techniques exist for using two-factor authentication and one-time passwords. One of the factors can also be a PKI, and some different approaches (albeit quite technical) can be found in Paterson and Stebila [26], Yang et al. [39], and Shoup and Rubin [31], just to name a few.

2.4 Other authentication methods

The above mentioned authentication methods are only a small selection of the many methods available. They are, however, the primary methods in use today when authenticating and using secure means of communication over the internet. There are of course a wealth of protocols and tools underlying and powering these methods (SSL, IPsec, SSH, just to name a few), but these are not of relevance to the subject covered in this review. The primary reason for looking at the above methods is the fact that they do not require physical presence or devices at the point of the resource wanted access to. In other words they allow access to resources from anywhere in the world.

Another popular usage of authentication is granting access to physically different located objects or sites, for instance accessing certain facilities at a plant or authenticating yourself in airports around the world. Here methods like identity cards, passports, RFID-enabled mobile phones, or biometric authentication (like fingerprints, voice identification, or retina scans) become handy. They require either a unique physical object in possession of the person wanting access or some form of contact with the person himself. As neither of these measures are easily practicable when communicating over the internet, we shall not investigate them further here. We refer to Table 3 in Braz and Robert [4] for a schematic overview of authentication methods and their applications, security and usability.

3 Usability factors

Usability factors (also sometimes referred to as *principles*, *rules*, *guidelines*, or *heuristics*) are the primary manuals in an HCI designer's toolbox. They often comprise of a set of guidelines or general goals that a certain software product or design should comply with. They could be compared to a buyer approaching a car dealer wanting a car that has four doors, a low mileage, air conditioning, and which is comfortable and easy to drive. The manufacturer of the car has most likely investigated on beforehand what the users demand and like about a car, and uses this knowledge to produce cars that sell as well as possible. He also knows about certain things not to do; for instance placing the driver in the right side of the car (at least not in Denmark) or omitting some form of storage compartment. Based on user response as well as damage reports from workshops they refine future models by correcting errors and increasing comfort and features – very much like versioning in a software product.

The ultimate goal of usability guidelines is to produce software that in is correspondence with what the user expects and needs to be able to achieve. In most cases, though, the developer or designer has no direct contact with the users of the system (as depicted in Figure 3) and therefore needs some ”advice” on how to model the system and its features and components in order to ensure good usability.

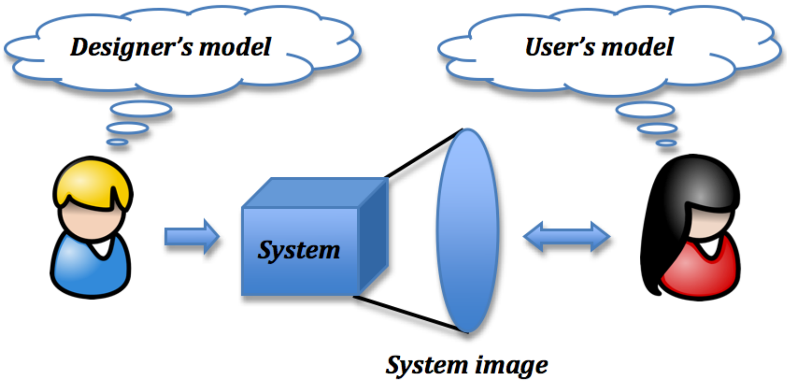


Figure 3: Sketch of the encounter between the designer's model of the system and the user's (adapted from Yee [40]). Notice the two-way communication between the user and the system image, and the one-way communication between the designer and the system.

3.1 Definitions of usability and guidelines

Now, why would we choose to dedicate a whole section to revisiting notions that are well-known to most people interested in or working with HCI? Couldn't we just reuse those usability factors that are already in widespread use today? First of all a multitude of different sets of factors exist, and various fractions in the research community swear to different such sets. Before going into the factors themselves, it is relevant to revisit the various definitions of usability itself. van Welie et al. [35] try to create an overview of these and compare them to each other. A widely accepted segmentation of usability is the ISO 9241-11 standard; usability is discussed in terms of *efficiency*, *effectiveness*, and *satisfaction*. They present two additional definitions by Schneiderman [29] and Nielsen [25] and relate the three to each other (see Table 1, adapted from van Welie et al. [35]). As the observant reader may notice, the three different sets of criteria are very much

ISO 9241-11	Schneiderman	Nielsen
Efficiency	Speed of performance	Efficiency
	Time to learn	Learnability
Effectiveness	Retention over time	Memorability
	Rate of errors by users	Errors/Safety
Satisfaction	Subjective satisfaction	Satisfaction

Table 1: Overview of usability definitions.

alike, and are indeed in many research communities seen as different aspects of the same thing. The reason for people wanting to redefine usability factors and criteria is most likely that certain definitions and names are better suited and more directly applicable in some specialised contexts than others. For those same reasons we continue on to look at the definition by Schneiderman and the "eight golden rules" he has set up for user interface design. These "rules" are more concrete than the usability definitions in Table 1 and can be seen as goals for the designer to be used during development:

1. *Strive for consistency:* dialogs and windows follow the same design guidelines, and similar actions have similar results (e.g. clicking on the little cross in the upper corner of a window closes it – no matter which window).

2. *Enable frequent users to use shortcuts:* allow users to use e.g. keyboard shortcuts or scripting to perform frequent tasks.[9]

3. *Offer informative feedback:* tell the user what went wrong, why, and how to proceed.

4. *Design dialogs to yield closure:* sequences of actions are shown to be completed, e.g. by means of confirmation dialogs or pages, or visible changes in the interface. Allow the user to put the finished set of actions behind him and carry on to the next.

5. *Offer error prevention and simple error handling:* the interface should prevent user mistakes by disabling unwanted options and provide enough information to handle errors effectively by the user himself.

6. *Permit easy reversal of actions:* transparent undo functions (in the sense that the result of performing them is obvious) are available for the most critical actions.

7. *Support internal locus on control:* do not let the software take over control of things unnecessarily during use. The user must always be able to take control.

8. *Reduce short-term memory load:* avoid having the user memorise unnecessary amounts of information, e.g. between different screens (recall Miller [24] from section 2.1).

Most people would probably agree that fulfilling these criteria would certainly be required for applications categorised as having good usability. However, with respect to security applications and authentication, Braz and Robert [4] argue that six of them are broken. The authors focus primarily on passwords but in the context of web authentication these are very close to being unavoidable. Only rule 1 and 4 survive. For instance, rule 6 is generally broken by the system locking an account after the user has entered three incorrect passwords. This makes perfect sense in the security community but the user is completely crippled in his or her tasks having to wait for a system's administrator to unlock the account again. Also, rule 3 is violated by concealing what the user is typing in password fields.

The above eight rules apply well to almost all types of software design today, but it may seem that security software has some problems conforming to them. And the more complex the underlying security concepts, the less informative and intuitive the design is prone to be. Therefore, there is a need for new or redesigned usability criteria for designing security software.

[9]In Schneiderman's latest edition from 2009 this rule is denounced *Cater to universal usability* and also deals with novice users getting enough help from the interface, for instance by means of explanatory texts and tooltips. In other words, design for the diversity of users.

3.2 User-centered security and design

The need for new criteria for designing and developing usable security is also recognised by Zurko and Simon:

> *"Since user-interface technology is constantly evolving and user needs are so diverse, no particular technology or architecture is always 'user-friendly'. While some usability researchers work on producing theoretical results that can be successfully used to guide initial UI design, very few principles are robust enough to be generally applicable and specific enough to directly influence engineering decisions."*
>
> Zurko and Simon [42], p28

They define the notion of *user-centered security* as *"security models, mechanisms, systems, and software that have usability as a primary motivation or goal"* (Zurko and Simon [42], p27). In other words, adding new technical security measures to the SSL protocol is not considered user-centered security, whereas e.g. designing login pages for websites is. They identify three directions that have been tried in the research community for combining security and usability: applying usability to secure systems, applying security to usable systems, and designing security features desired by users directly. The first is surprisingly poorly represented in the literature, the second has been extensively tried and seen to have only moderate success, and the third is sparsely tried with little success. At the time, challenges were indeed present.

Almost 10 years later, Zurko [41] picks up the subject again and looks at the progress made so far. She recognises that the problem has only become more pressing and identifies three groups of obstacles for designing usable security: human and social relations, technical challenges, and further difficulties with implementation and deployment. The first category comprises observations such as that non-understandable security mechanisms are ineffective, visualising security does not necessarily support decision making, users are unlikely to want further insight into the technological mechanisms behind the security systems (it just has to work and only the immediate use is interesting), and errors or breaches in security software can never be blamed on the user (it is always the responsibility of the designer to avoid errors – see golden rule number 5 above). The second contains observations that there does not yet exist an abstract model for human security behaviour, that security models are still somewhat unclear when supporting the user's decisions by making it clear which entities to trust and to what extent, and that this trust is based on only what is immediately observable or available to the user. Finally, Zurko advertises for criteria or checklists for evaluating the usability of security applications, as well as observing that contrary to 20 years ago most people now manage a wealth of e.g. passwords making most of them insecure due to users for instance writing them down (as analysed in section 2.1). Also, the reuse of security protocols and APIs pose a risk as they are employed in various new contexts and usage situations.

Research and collaboration is, however, increasing in amount, and Zurko rounds up a few possible principles for design, including differentiating *integrated security* (aligning security with common tasks and actions to make them secure per default) and *transparent security* (showing the relevant security information to the user in connection with actions). There is still no clear set of rules or guidelines with respect to usable security, though.

Yee [40] tries to remedy this by setting up ten design principles for secure interaction design. These principles are not meant to replace for instance the eight golden rules of the previous section, but are to be seen as a supplement to existing design principles with specific focus on development of applications utilising security. He initially bases the principles on Saltzer and Schroeder's *principle of least privilege*: every actor and object should perform actions under the least possible amount of privileges. E.g. when a regular user of a home-banking service wishes to transfer funds to another account, the program must ensure that in that specific context the user is only allowed exactly that and nothing else. We now briefly take a look at these ten principles.

Path of least resistance: relies on the notion of *fail-safe defaults* – the easiest or most natural way of performing actions should also be the most secure. This makes perfect sense with respect to authentication, in that actually just typing the password or authenticating with the means provided is naturally the most secure way of gaining access to the resource in question. Yee also argues that the easiest path should be made visible and convenient, and in cases where extra inconvenience is necessary, provide enough benefits to justify it. Authentication is of course a means of providing personalised information and documents to the user, thereby somewhat justifying the need for authentication in itself. Path of least resistance can be easily compared to Zurko's integrated security.

Appropriate boundaries: the system must enforce and provide visible and meaningful boundaries between objects, actors, and actions in order to clearly distinguish different rights and possibilities. In other words, a user of a government tax system does not care about the rights and privileges of administrators of the system, but cares whether other users have access in some form to the same information or actions as the user himself.

Explicit authorisation: any changes to the privileges of actors in the system affecting the current user (Yee calls this the *actor-ability state*) must be explicitly granted by that user. This occurs for instance in the Danish e-Boks system, where the system specifically asks permission to access the user's social security number, even though the user has already authenticated himself to the system via PKI authentication. Explicit authorisation can be compared to Schneiderman's seventh rule.

Visibility: the system must display enough information about actors and objects to enable the user to keep a mental overview of the rights and privileges of each entity connected to the system – albeit only the relevant ones, but especially in conjunction with changes to the actor-ability state. This compares well to Zurko's notion of transparent security.

Revocability: the user must at all times be able to revoke privileges or authorities granted to the system or other users. This is not to be confused with an actual undo function, but is to be seen as a prevention of further (mis-)use of the granted privileges. So, if a user has given some company the right to sell information about himself to third-parties, it must be possible to revoke this right again by the user. This principle resembles Schneiderman's rule number 6.

Expected ability: the user interface must clearly show what the user himself as well as others can and cannot do. A horror example of breaking this principle is that of Facebook not so long ago – deleting your uploaded images or even your entire profile did not mean that they disappeared from Facebook's servers. Rather, looking deep into Facebook's EULA revealed that they had already gained exclusive rights to everything uploaded or posted to their servers. Also, Facebook still has issues regarding the amount of information available about its users to total strangers.

Trusted path: the system must provide a secure and unspoofable path from the user to each other entity designated to manage authorities on request of the user. Yee gives the example of the well-known Control-Alt-Delete keyboard combination in Windows that guarantees unconditional access to the login window from anywhere inside the system or running programs.

Identifiability: actions and objects must be uniquely identifiable in the system, and the user must be able to perceive different objects and actors as actually being so. Much like Schneiderman's rule number one, only the other way around.

Expressiveness: the language used by the system to express security policies, authorities, and the like must be able to do so and in such a way that it is in harmony with the user's intention and perception of the system in general. This principle is also mentioned in Dourish and Redmiles [9] (section 4.1).

Clarity: simply states that is must be clear what the consequences of actions are, and that enough information must be provided for the user to make informed decisions.

Yee provides excellent user confidence statements accompanying the above principles in his summary as well as detailed examples of violations of the principles in current software in his appendix.

Although Yee provides an excellent set of interface design criteria as well as outlines their importance through examples, it is still not quite obvious how to implement these principles in practice. There are no specific guidelines for the developers of security systems and they are therefore forced into defaulting to well-known development methods not immediately suitable for the task at hand. Only in the testing phase a small possibility persists for correcting small mistakes and adding slight amounts of enhanced usability. More on this in section 5. At hand, though, we have now got an extensive set of (albeit somewhat abstract) principles that can be used as pointers to developing usable security. We are not aware of any software products or techniques actively enforcing these principles, but this may very well be due to the lack of the before-mentioned developer guidelines. Yee's principles are more suitable for further academic research than for actual development.

Whitten and Tygar [37] also set up five famous properties that developers need to keep in mind when developing security applications. They are in good harmony with the above principles from Yee:

Barn door property: Costly security mistakes must be ruled out in the design process, as it is often not possible to discover or recover from security breaches once they have occurred.

Weakest link property: Users (seen as the weakest link) should be motivated to handle security settings in order to avoid being compromised.

Unmotivated user property: Users do not care about security – it is a secondary goal and is just supposed to work. Therefore designers must make important security decisions clear and visible.

Abstraction property: Designers need to take into account the fact that complex security mechanisms and policies may seem like jibberish to the user.

Lack of feedback property: Systems must provide adequate feedback in order to prevent errors and support the user's tasks, which are at times somewhat unclear for developers.

Another final take on a set of design principles is given by tom Markotten [34]. They are based on the ISO 9241-10 dialogue principles (see Table 3 in van Welie et al. [35]) and consist of *error avoidance* (similar to Schneiderman's fifth golden rule but widened with respect to the "error tolerance" principle of the original ISO standard), *user guidance in critical situations* (identify critical situations and supply sufficient information and help to guide the user – much like Schneiderman's rule number 3), *user-friendliness especially for new users* (which must be considered one of the goals of HCI in general), *abstraction of security mechanisms* (somewhat comparable to Yee's Clarity, Expected ability, and Visibility), and *avoidance of interaction in everyday use* (another formulation of transparent security). The author also agrees with Yee's principle of Explicit authorisation and Schneiderman's rule of internal locus of control, in that control must always remain at the user's hands when not conflicting with the safety of the system in general. Although these new principles resemble those of Yee and Schneiderman and are just as abstract for developers, this time, however, the author actually provides a new and interesting development and testing process, which we shall return to in sections 4 and 5.

4 Developing usable security

In the previous sections we have established a fairly good model of the users of online authentication as well as their perception towards security and its applications in general. We have also had a look at various usability factors and criteria to keep in mind when developing secure software or websites. In this section we will take a closer look at some methods for actually developing the applications. We do not make a direct distinction between *regular applications* (e.g. programs running on the user's own computer) and *web applications* (hosted on a server and executed remotely from the user's computer), because they present pretty much the same obstacles and challenges with regard to security. Furthermore, most regular applications dealing with security in some form do so because they are connected to the internet or remote servers anyway.

The literature is full of tips, new ideas and approaches to enhancing usability of existing applications. Many of these focus on additions to web browsers. For instance, Jøsang et al. [22] discuss how trusted sites are presented to the user in web browsers and how much information to display. They make a few small suggestions on how to improve this by displaying an entity's logo in the browser outside of the actual content when the site is secure and also discuss mobile phone security – definitely an upcoming and rapidly evolving area of interest to HCI researchers – and how to display secure sites on mobile devices (here the size of the display is a serious limitation). The article being of some age, the proposals have never really taken off.

Jøsang and Patton [20] discuss pretty much the same things but also contribute with a view on browsers' different renderings of the same HTML code, and *links* (or *URLs*, as is the correct denomination). When considering the guidelines from the last section, it is curious how many secure sites redirect the browser to other servers during authentication. A webmail provider such as Hotmail.com redirects its users six or seven times (!) during a simple login process. And none of the subsequent servers – not even the initial login page – are on the original hotmail.com domain. Google's similar GMail service, on the other hand, stays on the google.com domain all the way through the login process. Such redirections happen without the user's consent and sometimes even knowledge and undermine the user's ability to maintain an overview of the system currently being used. This becomes especially evident when dealing with entities that keep personal and sensitive information about the user – redirecting can seriously undermine the trust in this entity taking care of the user's privacy: in how many places is my information actually stored?

Another approach to displaying security risks and concepts in a browser is presented as the Omnivore system by Flinn and Stoyles [13]. This extension shows a number of security "status lights" at the top of the browser, indicating whether problems with regard to for instance privacy or security are present. Clicking the red indicators reveals further information on the webpage itself. While the general idea may seem okay, in our opinion this solution adds too much stress on the user, having to constantly evaluate at least four criteria for *each* webpage visited. Also, it is not clear how the decision to turn one of the indicators red is made. What makes a site insecure or even untrustworthy? Potential risks of insecure web forms may yield red lights in Omnivore, even though they were not meant to be secure at all in the first place. The authors try to solve this by using online reputation systems, but even they cannot index the entire web.

Among *many* others, the above three suggestions may very well contribute to increased usability and clarity when browsing the web safely but they are mostly extensions and add-ons to existing applications, not means of actually developing applications. We shall now have a look at two such proposals.

The first is presented in Flechais et al. [12] and is called *AEGIS* (Appropriate and Effective Guidance for Information Security). It is initially designed to help developers identify security challenges early on in the development process and supply methods for dealing with them systematically. The most important factors in AEGIS are those of *asset modelling*, *risk analysis*, and *context of use*. Asset modelling is the process of defining and identifying the assets that the system is supposed to protect and manipulate, with respect to both data and honest as well as malicious users, also known as *operatives*. Risk analysis consists of systematically evaluating risks, their countermeasures, the cost of such, and a final evaluation of the cost versus benefits of the selected countermeasures. This process is iterative and sketched in Figure 4. AEGIS in general follows the spiral model for software development (see Figure 1 in Flechais et al. [12]) and an initial important task is to put together a group consisting of *facilitators* (who have the overall responsibility of applying the method as well as various documentation tasks), *stakeholders* (a diverse group consisting of both owners, developers, and users representing all parts of the design and usage population), and *security experts* (only needed if none other in the group has the necessary knowledge).

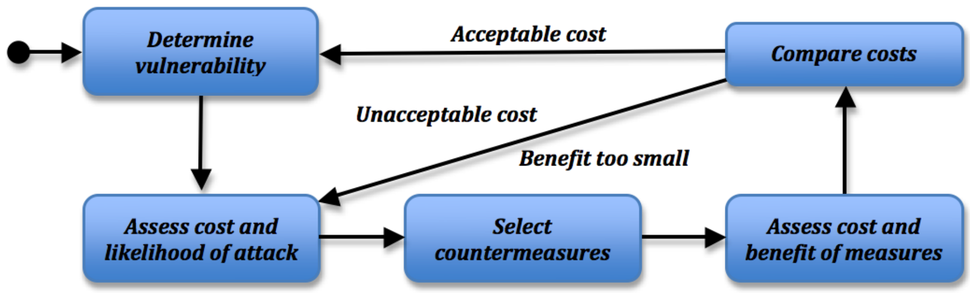


Figure 4: The iterative risk analysis and secure design process of AEGIS (adapted from Flechais et al. [12]). The important thing to take note of is the before-mentioned composition of the group carrying out this process – the users of the system are especially important in assessing the cost of countermeasures.

The composition of the participants in AEGIS is in our eyes not the standard way of developing software today, and it represents a different way of thinking about development as a joint effort between management, developers, and users; in good harmony with user-centred design. It does not, however, compromise the use of standard development methods, such as the spiral model, but merely integrates security aspects into them – increasing acceptance from developers and minimising security errors and lack of usability.

These things considered, though, AEGIS does not seem to be the ”ultimate solution” with respect to secure interaction design. Yes, users are integrated into the development process and security experts are consulted to make the systems secure enough for their purpose, but it seems like an incomplete solution when looking at the discussion of usability factors from the previous section. The AEGIS spiral model includes iterative testing, prototypes and all the bells and whistles, but initially there is no in-depth user classification or explicit evaluation of the specific usability requirements of the product being developed, other than the usual requirements specification (at least not as described by Flechais et al. [12]). It also does not provide means of measuring and/or testing the usability of security. It should be credited, however, for the fact that in preliminary testing of the model's capabilities it succeeded in bringing users and developers closer together and gaining insight into each other's domains, resulting in new-found problems to be solved and features to be implemented. Let alone that is quite an achievement.

A somewhat similar but more general approach is presented in tom Markotten [34]. It is based on known models; Nielsen's iterative analysis-design-testing model for general usability engineering, and standard security engineering development models (see Figure 1 in tom Markotten [34]). Although these models are well known to developers in the HCI as well as the security community, they usually live separate lives in the development process. The authors propose a new, combined engineering method with two branches: one for security engineering and one for usability engineering. The process can be seen in Figure 5.

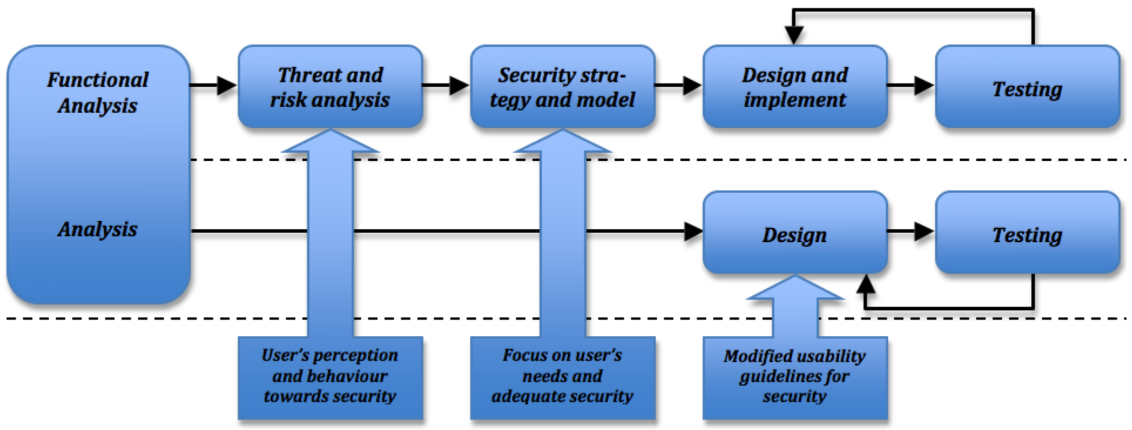

Functional Analysis
Threat and risk analysis
Security strategy and model
Design and implement
Testing
Analysis
Design
Testing
User's perception and behaviour towards security
Focus on user's needs and adequate security
Modified usability guidelines for security

Figure 5: Modified sketch of the engineering method given in tom Markotten [34]. On top is security engineering, and in the middle we have usability engineering. In the bottom are the contributions from research specifically concerning usability and security. After separate testing, the process ends in a conjoint testing phase (not shown).

The authors propose to use knowledge from usability research during important phases in the security development process as well as merging the analysis process to get a much more precise image of the requirements by management and in particular the users of the system. Albeit being very briefly mentioned, they apply their own set of security design guidelines (discussed in section 3.2) and give short mentions of methods for the analysis and testing phases of the process.

We like the model given by tom Markotten as it relies on common design methods and recognises the need for revised principles to build the design upon. As the author herself notes, though, the model is far from complete, and its description in the article must be considered superficial. Whereas Flechais et al. spend a lot of energy on risk analysis and security design (which resembles step 2 and 3 in the model in Figure 5), tom Markotten only spends a few lines. It is, however, a very good starting point for developing a complete model for designing and implementing usability in security software. None of the two models presented give some decent insight or advice on how to test and/or measure the developed products with respect to usability, though. We investigate this further in the next section.

5 Testing usability of security

The final, but very important, missing link in obtaining usability in security applications is actually measuring and testing the results of our efforts. In other words, we are looking for methods and principles for evaluating and testing how well our developed software fares in relation to our usability factors and principles. Some of the development models presented in the previous section have focused only lightly on actual testing, both during development and end-user testing. Although a wealth of common testing techniques are already available (see for instance Jeffries et al. [19] for a comparison of four of them), we shall not spend time describing them. Instead, we shall look at concrete measures and taxonomies to apply to these existing techniques, as well as outlining some previous case studies to see how testing has been applied with regards to usability in security applications.

5.1 Evaluation methods

Our first step is somehow figuring out what to look for when measuring and testing usability. Clearly, the user not being able to find a certain key on the keyboard while entering a password can not be considered security-critical, perhaps not even related to usability at all. Kaiser and Reichenbach [23] propose a *taxonomy* for classifying usability errors with respect to security. The first level of the taxonomy is dividing errors into usability-related and security-related problems. The intersection of these two sets is the interesting one: the *security critical usability problems.* This group of problems is then again divided into problems that occur *despite* the user having sufficient competences within security, and the ones that occur *because* of lack of such knowledge. The last level is dividing each of these two categories into *slips* (the correct sequence of actions did not yield the expected results), and *mistakes* (a doable sequence of actions originating from a faulty plan). The authors continue to set up inequalities for measuring the criticalness of the different types of errors, and recommend using either cognitive walkthrough or heuristic evaluation for actually discovering the errors.

One could argue whether error categorisation is a sufficient means for measuring usability. Users might hate the way a system works, but nevertheless they succeed in completing their tasks. Such scenarios are not caught by categorising errors alone. On the other hand, a system that never provokes errors *could* be seen as highly usable. However, Kaiser and Reichenbach provide a fairly good taxonomy for classifying the errors once they have been discovered. Our question still remains: how do we best discover the usability errors and design flaws in security applications?

Braz et al. [5] introduce a new evaluation method that tries to answer the above question. The method is called *Security Usability Symmetry* (or just SUS)[10] and builds on the hierarchical *Quality in Use Integrated Measurement Model* (QUIM), an industry-standard evaluation model for usability in general that adds factors such as *safety* and *trustfulness* to the more standard ones. QUIM consists of 10 usability factors (think of them as counterparts to Schneiderman's golden rules), which are again broken down into 26 measurable criteria, which are again divided into 127 specific metrics. The authors recognise the fact that in the research community it is a common belief that a tradeoff is needed between usability and security and they therefore investigate how to produce this tradeoff. During this process they give a set of short characteristics for designers and developers to keep in mind – not exactly principles, but merely observations – as well as extending the usual task scenarios with *security scenarios*: simply a task scenario where the user has to interact with some form of security architecture in some way.

This is a very specific way of including security directly into regular testing, allowing

[10] To the confusion of everyone, another usability evaluation method by Brooke exists, also called SUS (System Usability Scale).

it to be closely related to the well-known task descriptions. The main contribution of the method is to put up security evaluation methods next to usability evaluation methods to be able to quickly decide the tradeoff between the two. Although the authors leave a link for a detailed description of the model,[11] it still seems to us that the model is too briefly described to be of immediate use from the article alone.

When trying to relate more closely to web-based authentication, we find the framework given by Straub and Baier [33] very useful. Here the authors present a framework for evaluating PKI-based applications consisting of 15 evaluation categories with accompanying example questions. They apply a numerical scale to be able to get quantitative measurements of usability and use a scale ranging from -2 (below standard) to +2 (outstanding). Also, they propose to assign different weights to the results according to the context within which the application is to be used. The 15 evaluation criteria are split into three groups: *deployment issues*, *ergonomics*[12], and *security features* and the weights on the criteria are set according to how important these three groups are. For instance, in a high-risk environment, security features are assigned 60% importance, whereas in private use they are only assigned a 20% importance. In this latter scenario, however, deployment and ergonomics take up 40% each. The evaluation criteria as well as the example questions are based on Whitten and Tygar's five properties, redefined to fit a PKI context. Unfortunately, due to space restrictions the authors don't actually present an example of the framework in use.

We like the framework by Straub and Baier for actually basing on and actively using properties and principles derived in previous studies. Although the use may seem limited due to its target audience in a specific corner of security software, it should be fairly straightforward to modify the framework to work with other types of authentication. The framework is meant to be used by an evaluator, though, and is not directly useful for end-user testing. Rather, it is able to support design decisions or serve as a supplement to requirement specifications etc – and serves as a valuable tool for conducting cognitive walkthroughs. However, this work feels incomplete as well. The framework presented would probably have been of great benefit to Balfanz et al. [3] when implementing their PKI-based wireless network at PARC.

5.2 Usability studies

Not that many usability studies of security software exist today. We will now have a look at some of them, and for a very thorough review of 180 usability studies and a comparison of the ways and on which basis they measure usability, we refer to Hornbæk [17]. Although the studies presented are not specifically targeted at web authentication, they investigate techniques applied here as well, such as PKIs.

The first known study of this kind is credited to Whitten and Tygar [37]. This much-cited study looks at the usability of *PGP* (Pretty Good Privacy), a PKI-based encryption scheme meant for secure email and identification and self-proclaimed as having excellent usability. The PGP model relies on private and public keys as well as a *trust network*, allowing individual users to put some amount of trust in other people's public keys in

[11]A link which (at least on July 2) was broken, effectively preventing us from further investigating the method.

[12]Ergonomics here refer to general usability. The odd name is chosen to differentiate it from the other categories.

order to help verify their identity. The authors tested various simple tasks performed in the PGP system, such as generating keys, publishing them, and encrypting an email within reasonable time limits. They also set up five properties of usable security (which are, however, somewhat superficial) and from these properties define what usable security is and use this as a basis for their testing (Whitten and Tygar, p5):

1. Users are reliably made aware of of the security tasks they need to perform.
2. Users are able to figure out how to successfully perform those tasks.
3. Users don't make dangerous errors.
4. Users are sufficiently comfortable with the interface to continue using it.

The above definitions do not in our opinion add much to the ones from section 3 with respect to security. One must bear in mind, though, that this article is one of the first of its kind and the authors should therefore be credited for opening up the bag of usability studies in security.

They initially perform a cognitive walkthrough with elements from heuristic evaluation and continue on to perform a user test with 12 participants (in line with the recommendations from [23] and [34]) to assess usability. A single extensive task is presented to the users. Within 90 minutes, only one third of the test subjects were able to perform basic email encryption, and one in four accidentally sent the message unencrypted, thereby constituting a security risk. Based on these results and the cognitive walkthrough, among others, the authors conclude that PGP is not sufficiently usable for the broader population.

The main contributions by Whitten and Tygar are first and foremost their specific focus on usability in security applications. Secondly, the appendices to their article present very thorough descriptions of test methodology, participants, and setup; something of high use for subsequent studies. The main differences to regular usability studies are the initial considerations described above as well as the points of interest in the thorough cognitive walkthrough, specifically looking at the criteria set up in the beginning of the article. Moreover, they suggest that usability in security applications has more to it than just a good interface.

In a more recent study, DeWitt and Kuljis [7] investigate the usability of Polaris, a system developed for Windows used for shielding various applications from viruses and other threats by enforcing strict access rights on each application – a process known as *polarising* the application. Polaris builds on the *principle of least authority* by Yee[13], and Polaris was developed with high usability as one of its primary goals. This study also performs a user test very similar to Whitten and Tygar's and relates the results to the *efficiency-effectiveness-satisfaction* model (see also section 3.1). Their conclusions are also pretty much on par with Whitten and Tygar's: Polaris fails in being sufficiently usable for the broad population of users. The main problem seems to relate to users not knowing when or how to make security decisions – relating back to the perception study in section 1. Also, speed and ease of task completion was on many occasions prioritised over security, in line with Yee's path of least resistance. Another interesting observation

[13]Unfortunately, we have not been able to gain access to the more recent article by Yee describing this principle: *Aligning security and usability* (2004).

is that users may have been influenced by the experimental setup of the tests, inferring the argument that the best security decisions are made when users actually feel at risk – something that is hard to obtain during controlled experiments. The authors conclude that as of yet, no applications seem to have been developed with security and usability tied together from the beginning.

As the two studies above show, although applications may have been designed for ease of use and good usability, they fail in delivering so when being put to the test. Most of the research community agrees that a substantial factor causing this is the lack of integration between usability and security experts early on the design process. Also, a lack of knowledge about the end-users at hand influence bad design decisions that are too costly to fix at the late stage of end-user testing the product – often the first stage these problems are encountered.

An important factor to consider in general is that of users not being at risk when performing usability evaluations. Although this doesn't immediately affect regular usability evaluation methods, it certainly has an effect when measuring perceptions and behavioural patterns. Perhaps conducting group testing with a prize for task completion could be a way to go, as well as making test tasks as realistic and tied to user's needs as possible. Yet again, though, it seems that research is somewhat spread out in stead of coming together and developing more complete methods.

6 Conclusions and further directions

Through this review we have taken a broad view of the HCI literature dealing with usability and security, with special focus put on authentication on the web. We have investigated the users of such authentication and seen that they constitute a varied population of internet users and that these have different ways of navigating with respect to security. Most have, however, found a balance between risk and trust; a popular segment often mistakenly prioritised by developers. Many users are not able to perceive or navigate the security aspects correctly, posing a security threat due to misunderstanding, cumbersomeness, and circumventing measures meant to protect them.

We have reviewed and evaluated three of the most popular and current authentication methods on the web. No matter the method, passwords most likely play an important role and should therefore be made as usable and secure as possible – utilising techniques such as mnemonic-based or randomly chosen passwords. Although PKI-based security offers close to ultimate *theoretical* protection, the difficulty of implementing and using it poses a great risk. Multi-factor authentication has promising ways of relieving some of the technical difficulties of PKI solutions, while still retaining similar or even better security. We have also reviewed several proposed definitions of usability factors and criteria with respect to security. It seems to us that these factors are very much alike, though not in equally elaborate ways, and so far not many of them have been in use in active development. The need for new criteria has emerged from the observation that existing usability factors are not adequate for discovering every usability defect in security software. This has sprung the advent of user-centered design.

While discussing the actual development of secure software solutions we have looked at a few proposals for extending existing software (primarily web browsers) to enhance privacy control as well as usability. We have also outlined two models for incorporating usability into secure software design, although they both seem somewhat incomplete and in need of further development. Finally, we have reviewed different methods and frameworks for testing the usability of secure software systems as well as outlining some specific studies conducted on this type of software. They showed that although an almost sufficient amount of evaluation methods exist, the general principles for secure interaction design are still not followed in current development models.

Throughout the review we have outlined deficiencies in current research and the results obtained so far. Below we summarise a few proposals for future research in the HCISEC area:

- Although the need for usability in secure applications has been recognised since 1975, the research community has not paid it much attention until just a few years ago. As online authentication has become an everyday task for most people online, in our opinion researchers need to perform much more work in this interesting cross-disciplinary field. In the words of Zurko, they need to step up to the challenge.
- We know that the group of people utilising online authentication today is extremely vast but we do not to our knowledge possess up-to-date insight on the perceptions of security among its users. As people get more and more used to security measures their habits and perception are bound to undergo changes that may require redesigning or modifying existing development and design methods.
- The usability factors presented in section 3 are good examples of the need for the research community to come together and agree on factors and criteria that are (if not universally) usable for secure interaction design. In our opinion, Yee [40] would constitute a good starting point.
- We agree with the research community's desire for more usability studies in this area. They form a practical approach to discovering problems and building experience and supplement the more theoretical considerations which often do not evolve as expected.
- The lack of commonly agreed on principles as stated above poses an obstacle on the path towards producing a solid and complete model for software development and evaluation of secure software. The methods available are either incomplete or focusing on some certain phase in the development process. This makes it difficult for designers to adopt the different independent models, and we should like to see the development of a complete model that stretches from early development through testing to delivery. We are not asking researchers to reinvent the wheel; we are merely suggesting putting together the individual pieces to finally make the HCI car run securely.

References

[1] Two-factor authentication, 2005. URL http://en.wikipedia.org/wiki/Two-factor_authentication.

[2] A. Adams and M. A. Sasse. Users are not the enemy. *Communications of the ACM*, 42(12):40–46, 1999.

[3] D. Balfanz, G. Durfee, D. Smetters, and R. E. Grinter. In search of usable security: Five lessons from the field. *IEEE Security and Privacy*, 2(5):19–24, 2004.

[4] C. Braz and J.-M. Robert. Security and usability: The case of the user authentication methods. In *Proceedings of the 18th International Conference of the Association Francophone d'Interaction Homme-Machine*, pages 199–203, 2006.

[5] C. Braz, A. Seffah, and D. M'Raihi. Designing a trade-off between usability and security: A metrics based-model. In *Proceedings of the 11th IFIP TC 13 international conference on Human-computer interaction*, volume II, pages 114–126, 2007.

[6] L. F. Cranor and S. Garfinkle, editors. *Security and Usabillity: Designing Secure Systems that People Can Use.* O'Reilly, 2005.

[7] A. J. DeWitt and J. Kuljis. Aligning usability and security: A usability study of polaris. In *Proceedings of the second symposium on Usable privacy and security*, pages 1–7, 2006.

[8] W. Diffie and M. E. Hellman. New directions in cryptography. *IEEE Transactions on information Theory*, 22(6):644–654, 1976.

[9] P. Dourish and D. Redmiles. An approach to usable security based on event monitoring and visualization. In *Proceedings of the 2002 workshop on New security paradigms*, pages 75–81. ACM Press, 2002.

[10] P. Dourish, J. D. de la Flor, and M. Joseph. Security as a practical problem: Some preliminary observations of everyday mental models. *Workshop on HCI and Security Systems, CHI2003*, 2003.

[11] P. Dourish, R. E. Grinter, J. D. de la Flor, and M. Joseph. Security in the wild: User strategies for managing security as an everyday, practical problem. *Personal and Ubiquitous Computing*, 8:391–401, 2004.

[12] I. Flechais, M. A. Sasse, and S. M. V. Hailes. Bringing security home: A process for developing secure and usable systems. *Workshop on New Security Paradigms*, pages 49–57, 2003.

[13] S. Flinn and S. Stoyles. Omnivore: Risk management through bidirectional transparency. In *Proceedings of the 2004 Workshop on New Security Paradigms*, pages 97–105, 2004.

[14] B. Friedman, D. Hurley, D. C. Howe, E. Felten, and H. Nissenbaum. Users' conceptions of web security: A comparative study. *Conference Extended Abstracts on Human Factors in Computer Systems*, pages 746–747, 2002.

[15] T. Grandison and M. Sloman. A survey of trust in internet applications. *IEEE Communications Surveys*, 3(4):2–16, 2000.

[16] M. E. Hellman. An overview of public key cryptography. *IEEE Communications Society Magazine*, pages 24–32, November 1978.

[17] K. Hornbæk. Current practice in measuring usability: Challenges to usability studies and research. *Internation Journal of Human-Computer Studies*, 64(2):79–102, 2006.

[18] G. Iachello and J. Hong. End-user privacy in human–computer interaction. *Foundations and Trends in Human-Computer Interaction*, 1(1):1–137, 2001.

[19] R. Jeffries, J. R. Miller, C. Wharton, and K. M. Uyeda. User interface evaluation in the real world: A comparison of four techniques. In *Proceedings of the SIGCHI conference on Human factors in computing systems: Reaching through technology*, pages 119–124, 1991.

[20] A. Jøsang and M. A. Patton. User interface requirements for authentication of communication. In *Proceedings of the Fourth Australasian user interface conference on User interfaces*, volume 18, pages 75–80, 2003.

[21] A. Jøsang and S. L. Presti. Analysing the relationship between risk and trust. In T. Dimitrakos, editor, *Proceedings of the Second International Conference on Trust Management*, 2004.

[22] A. Jøsang, M. A. Patton, and A. Ho. Authentication for humans. In *Proceedings of the 9th International Conference on Telecommunication Systems (ICTS2001)*, 2001.

[23] J. Kaiser and M. Reichenbach. Evaluating security tools towards usable security. In *IFIP 17th World Computer Congress*, 2002.

[24] G. A. Miller. The magical number seven, plus or minus two: Some limits on our capacity for processing information. *The Psychological Review*, 63(2):81–97, 1956.

[25] J. Nielsen. *Usability Engineering.* Academic Press, London, 1993.

[26] K. G. Paterson and D. Stebila. One-time-password-authenticated key exchange (not yet published). In *15th Australasian Conference on Information Security and Privacy (ACISP 2010)*, 2010.

[27] R. Rivest, A. Shamir, and L. Adleman. A method for obtaining digital signatures and public-key cryptosystems. *Communications of the ACM*, 21:120–126, 1978.

[28] J. H. Saltzer and M. D. Schroeder. The protection of information in computer systems. *Fourth ACM Symposium on Operating System Principles*, 1973.

[29] B. Schneiderman. *Designing the User Interface.* Addison-Wesley Publishing Company, USA, 1998.

[30] K. B. Sheehan. Toward a typology of internet users and online privacy concerns. *The Information Society*, 18:21–32, 2002.

[31] V. Shoup and A. Rubin. Session key distribution using smart cards. pages 321–331. Springer-Verlag, 1996.

[32] R. E. Smith. *Authentication: From Passwords to Public Keys.* Addison-Wesley, 2001.

[33] T. Straub and H. Baier. A framework for evaluating the usability and the utility of pki-enabled applications. In *1st European PKI Workshop Research and Applications*, 2004.

[34] D. G. tom Markotten. User-centered security engineering. In *NordU2002 - The 4th EurOpen/USENIX Conference*, 2002.

[35] M. van Welie, G. C. van der Veer, and A. Eliëns. Breaking down usability. In *Proceedings of Interact '99*, 1999.

[36] T. Whalen and K. M. Inkpen. Gathering evidence: Use of visual security cues in web browsers. In *Proceedings of Graphics Interface*, 2005.

[37] A. Whitten and J. D. Tygar. Usability of security: A case study. Technical report, School of Computer Science EECS Carnegie Mellon University Pittsburgh and University of California SIMS, Berkeley, 1998.

[38] J. Yan, A. Blackwell, R. Anderson, and A. Grant. Password memorability and security: Empirical results. *IEEE Security and Privacy*, 2(5):25–30, 2005.

[39] G. Yang, D. S. Wong, H. Wang, and X. Deng. Two-factor mutual authentication based on smart cards and passwords. *Journal of Computer and System Sciences*, 74:1160–1172, 2008.

[40] K.-P. Yee. User interaction design for secure systems. In *4th International Conference on Information and Communications Security*, pages 278–290, 2002.

[41] M. E. Zurko. User-centered security: Stepping up to the grand challenge. In *Proceedings of the 21st Annual Computer Security Applications Conference*, pages 187–202, 2005.

[42] M. E. Zurko and R. T. Simon. User-centered security. In *Proceedings of the 1996 Workshop on New Security Paradigms*, pages 27–33, 1996.

[43] M. Zviran and W. J. Haga. A comparison of password techniques for multilevel authentication mechanisms. *Computer Journal*, 36(3):227–237, 1993.