Protocol Variants and Electronic Identification

Kristian Gjøsteen

May 29, 2013

Abstract

It is important to be able to evaluate information security systems involving humans. We propose an approach in which we consider the system as a cryptographic protocol, and users are modeled as ordinary players. To model the fact that users make mistakes that affect security, we introduce protocol variants that model mistakes or combinations of mistakes. By analysing the base protocol and its variants, and at the same time considering how likely each variant is, we get a reasonable estimate of the real security of the system.

Our work takes the form of a case study of four Norwegian federated identity systems, as well as two proposals for improved systems. The four systems span a good mix of various types of federated identity systems.

1 Introduction

Modern cryptographic theory is a success. The discipline of provable security allows skilled practitioners to design efficient protocols that achieve many important security goals under reasonable heuristics or well-studied mathematical conjectures.

Modern cryptographic protocol analysis has many tools that seem useful in today's information security world. For instance, just as modern home computers are taken over by malware and servers are broken into and used for further attacks, cryptographic researchers have always assumed that some players in a cryptographic protocol can be controlled by the adversary.

Cryptographic theory typically deals with a system of interactive Turing machines, where some machines cooperate to achieve some security goal, while the remaining machines try to frustrate that goal. Even though many cryptographic textbooks talk about Alice and Bob, humans are usually not considered as actually executing protocols. Indeed, most cryptographic protocols involve computations that are practically infeasible for humans.

The interactive Turing machine is a nice abstraction for a computer program, and as such it is very useful for analysing systems of computers employing cryptographic protocols. Frameworks such as universal composability [3] incorporate humans driving cryptographic protocols into an environment, which is modelled as a single interactive Turing machine.

One vital component for internet security is the TLS protocol, which is a descendant of the SSL protocol invented to facilitate electronic commerce on the internet. TLS is widely considered to be reasonably secure[1].

However, it is well-known that many information security systems using TLS fail. One reason is that TLS assumes a public key infrastructure. For various technical, historical and societal reasons, that public key infrastructure sometimes asks the human driving the computer whether or not to accept certain data. Deciding if that data should be accepted is conceptually simple, but in practice very difficult, even for cryptographers familiar with the technical issues. The end result is that a human mistake can essentially turn off security.

The example of TLS shows that humans are much more involved in running many cryptographic protocols, than merely providing input and using the output. The fact that cryptographic software was hard to use correctly was observed a long time ago and forcefully documented [11]. It has also been observed that adversaries can use this fact to cause users to make mistakes [9] that affect security.

We believe that if the cryptographic toolbox is to be successfully applied to real-world security, humans must be modelled as part of the system. Involving humans in cryptographic protocols is a hard problem. There are several examples [1, 8] where a voter following the instructions for an electronic voting system is modelled as executing a very simple program, but such analysis is not sufficient.

There is a need for new methods of analysis and new knowledge of user behaviour. The approach of Ellison [4] is in many ways a promising approach, and has lead to positive work [10] where the theory helps design better protocols.

Our contribution In this work, we are interested in tools for evaluating the real security of a cryptographic protocol involving humans. We consider the case of federated identity systems, where a shared electronic identity is used to login to many web sites.

Our approach begins by modelling a diligent user carefully following the instructions for correct use of the federated identity system. This gives us a base cryptographic protocol with one human player. Then we consider how the user may make a mistake. Note that we do not consider all possible mistakes, only those that under normal operation will not prevent logging in: regular users will quickly learn how to avoid other mistakes.

Once this analysis is complete, we create a number of variants of the user's program, one for each possible mistake or combination of mistakes. We analyse the security of each variant separately, determining exactly what security properties it has. Once this is done, we consider the likelihood of a random user choosing a given variant. Combining these pieces of data results in a clear idea of the effective security of a given federated identity system.

[1] TLS was not designed using modern cryptographic techniques. Consequently, improvements to old attacks and new corner cases are regularly discovered.

We illustrate this approach with a case study based on four Norwegian federated identity systems, as well as two conjectured systems that would be easy to deploy in Norway. We have chosen the Norwegian systems first of all because they are easy for us to study in practice. They also represent a nice mix of various approaches, and they all claim to provide proper security, and are used for real-world applications like internet banking, access to medical records and electronic voting. While we could have studied other widespread international systems deployed by various corporations, especially in conjunction with social networks, and other government systems, such systems would add little to the mix we already have.

For our protocols, it seems to be easy to determine the possible mistakes and their combinations, so finding a complete set of variants is easy. Analysing the security of the variants separately is fairly routine cryptography, and we do not include the detailed analysis (a brief analysis is included in the appendices).

The most problematic issue with our case study is estimating the likelihood of choosing variants, because we do not have reliable data. In this work, we have made certain estimates that we believe are reasonable, but these are conjectures only, they are not based on data.

To illustrate applications of our results, we consider two Norwegian applications relying on federated identity systems, internet banking and a web site for personal medical information. We show how the security analysis of the federated identity system can be used to estimate the risk posed by the application.

We stress that the owners of the systems and applications studied are or should be aware of our results. The main point of this paper is not the results themselves, but the techniques used to derive these results.

Future work The information security folklore contains a lot of information about what people do wrong, and there are even some studies about how people make mistakes. But there is little systematic work that can provide data for the various likelihoods our analysis needs. We need real experiments with real people.

Federated identity systems are an interesting case study, but other types of protocols should be studied. One example is electronic voting protocols, attacks against which have already been discussed above [9].

Our approach in this case study considers only honest users and honest infrastructures. Studying dishonest relying parties and infrastructures is also interesting, and for other types of protocols, dishonest users could be important.

We have also restricted our analysis to what is essentially passive attacks. The adversary is trying an attack strategy and hoping that the user makes the required mistakes. If the mistakes are common, this strategy works well. However, if some mistake is less common, the attacker may actively try to cause the mistake. One example is providing forged on-screen guidance. Many users will have learned that on-screen instructions are often more accurate than instructions recalled from memory or printed manuals. The likelihood that they follow forged on-screen guidance may therefore be large. Studying active attacks

is an important future topic.

We have not considered the fact that these likelihoods are not static values. In reality, they will vary with time and with user. For instance, if an attack is attempted and fails, that will typically make a user adhere more closely to the protocol for some time, increasing the likelihood of the user selecting the base variant or variants with few mistakes. If widespread attacks are detected, the same effect should occur.

The analysis done in this work is very informal. We will need to formalize these arguments. One approach is to adapt the ideas from universal composability: the security defined by the ideal functionalities would have to be adapted to take into account the various mistakes user might make. For instance, when told to establish a secure channel with some party, the ideal functionality may first flip a biased coin. If the result is heads, the user gets the usual security. If it is tails, the ideal adversary is allowed to compromise the channel. We can also allow different users to have different biases, and the bias may vary with time.

2 Federated Identity Systems

An *electronic identity* is a method for convincing a remote computer that you are indeed who you claim to be. One example of an electronic identity is a username and a secret shared by the remote computer and you. There are many cryptographic protocols that allow the user to convince the remote computer that he knows the password. If the password really is secret, it can (depending not only on the quality of the cryptographic protocol) logically follow that the user is who he claims to be.

Note that identification is usually used to establish a session with the remote computer, *logging in*. The main security goal of an electronic identity system is therefore that of a *establishing a secure channel*:

> *Suppose the remote computer accepts that a given user has established a session. If the user's computer is honest, the session established is between the remote computer and the user's honest computer.*

Note that we assume that if the user's computer is compromised, the attacker can get access to the remote computer through the established session, so the question of security is moot.

A *federated identity system* is a system that enables a single electronic identity to be used to login with many separate remote computers, or *relying parties*. Now, however, it is not enough to establish a secure channel, the system must also ensure that the secure channel is established with the intended relying party, *correct intention*:

> *Suppose the relying party accepts that a given user has established a session. Then either the user intended to establish a session with*

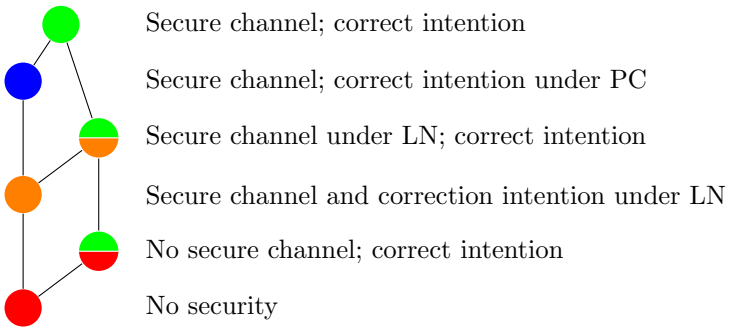

Figure 1: Colour coding of security claims: No security claim against stronger attackers holds for the protocol. A natural partial order amongst the security claims is indicated. Assumptions: PC – the user's computer is honest, but the local network may be corrupt; LN – the local network is honest, but relying parties may be compromised.

> *that relying party, or the user's computer is honest and a secure channel has been established.*

It is certainly possible to ensure correct intention, while not ensuring a secure channel. Vice versa, it may also be possible for a user to be tricked into connecting his honest computer to a relying party he did not intend to connect to. This will be a problem if the user later injects sensitive information into the secure channel, information intended for some other relying party. While this is a real problem, we shall not consider it in this paper, and therefore this shall not be considered a breach of security.

There are many forms of attackers against identification systems. The weakest attack is that of impersonating or compromising some relying party in the hope of stealing sufficient information to be able to establish a session with some other relying party or the impersonated relying party. Impersonating a relying party is usually known as a *phishing attack*. These attacks are technically very easy to execute, and we may assume that these are always available to an attacker, even against well-protected users.

A stronger attacker is one that has access to the user's local network or otherwise gain access to the computer's communications[2]. Communications that are not cryptographically protected may be interfered with. This is technically a more difficult attack to pull off.

The strongest attacker we consider is one that has compromised the user's computer. As discussed above, this renders the goal of establishing a secure channel moot, but correct intention is still possible to ensure, if a trusted display is available.

Note that a user, in addition to his computer, may have several other devices available during login. Examples include a mobile phone, smart cards and one-

[2]One possibility would be DNS poisoning attacks.

time code generators. Some devices may be connected to the user's computer (e.g. smart cards), while others will not be (e.g. one-time code generators).

We shall assume that, while a computer may be compromised, the user's other devices may not be compromised. For many devices, this is a reasonable assumption. Indeed, smart cards are designed to resist attacks.

Unfortunately, for mobile phones the assumption is debatable at best. For many simple mobile phones, the assumption is probably valid. So-called smart phones, on the other hand, are often connected to the user's computer, and it seems unreasonable to believe that a compromised computer cannot also compromise the phone. Since mobile phones are increasingly being used for authentication purposes, such attacks will become more likely in the near future. When this happens, one of the assumptions underlying the case studies in this paper will be at least partially incorrect.

3 Case studies

We begin with a case study of four federated electronic identity systems widely deployed in Norway. We should first remark that three of the four systems are secret, in the sense that the owners of the systems do not want independent researchers looking at them. The documentation for the fourth system may in some sense be public, but only in a form that is too hard to extract a protocol from.

This means that we do not know (or cannot publish) the exact cryptographic protocol in use in these systems. However, what is publicly available (or inferrable from public sources or by interacting with the systems themselves) is the interaction between the user and the system. This is the part of the system that we are interested in studying, and we have sufficient information for our analysis to apply as qualified below.

For the technical analysis of the protocols, we have invented reasonable and fairly obvious approximations for the non-public parts. These protocols are given in the appendices. Since every attack we use apply to the real protocols (they depend only on public information about the protocols), the protocols we analyse are at least as secure as the real protocols.

However, there is reason to suspect that not every protocol is as secure as our corresponding invented protocol. Therefore, our results should only be considered upper bounds on the security of the systems, not correct estimates.

We visualize our results as a collection of coloured circles, each circle representing a variant. A circle's colour describes the security achieved by the variant, coded according to Figure 1. The area of each circle is an estimate of how likely the variant each. We believe that this visual presentation is both easy to understand and gives an intuitive way to compare different systems.

For our passive analysis, there is a natural partial order on the variants: a "smaller" variant involves more mistakes than a "larger" variant. This partial order satisfies a natural requirement, namely that a "smaller" variant cannot achieve a stronger security goal than a "larger" variant, because making more

mistakes should not improve security. The edges between the circles indicate this partial order.

Finally, as we noted in the introduction, TLS is considered reasonably secure until you consider the PKI issues. Some modern browsers have made it much more difficult for users to make mistakes, but incorrectly issued certificates remain a problem. This means that all of the systems we discuss are in some sense vulnerable to flaws in the TLS PKI. We shall ignore these problems and assume that our users deal correctly with any TLS issues.

3.1 BankID

The BankID system is an electronic identification system jointly developed by Norwegian banks and used as the primary login method for Norwegian internet banks. The BankID system has also been offered to other web sites to serve as a login system, and is used by a government identity portal. It is therefore a proper federated identity system.

The system owners consider the system secret. They do not want independent researchers to look at it. However, since the client part of the system is implemented as a Java applet, it is fairly easy to decipher this part of the system, which reveals most of the system architecture. Some years ago, two independent studies of the system were done, and a number of issues with the system were found [5, 6, 7]. Since then, the cryptographic protocol has been modified[3], but the interaction with the user is essentially unchanged.

The system is based upon passwords and one-time codes. The one-time codes are usually generated by a special device or by software running on a mobile phone. The relying party places a special Java applet on its login page. The user enter his credentials into the applet, which contacts a central infrastructure to verify the credentials. If they verify correctly, the user's web browser is redirected to the relying party's landing page.

The user role of the BankID protocol and its variants are given in Figure 2. The user side of the protocol is derived from the instructions given by the official BankID web site.

Except for the Java applet, this system is basically about entering your internet banking password on any web page. The Java applet does complicate the analysis slightly, but as we shall see, it does not significantly affect the security of the system.

Analysis The BankID Java applet is signed and requires full privileges on the computer it is running on. This means that the Java browser plugin will display a security dialog asking the user to grant or deny access.

The BankID instructions imply that this dialog authenticates the Java applet appearing on the web page, saying that unless the security dialog appears and the applet is signed by the banks, the user should not enter his credentials.

[3]The implementation may change again in the near future, replacing the Java applet by JavaScript code. The analysis in this paper will be essentially unchanged.

Base

1. If there is no Java security dialog, stop.
2. If the Java applet is not the BankID applet, stop.
3. Enter username.
4. Generate one-time code.
5. Enter one-time code.
6. Enter password.

Variants It is hard to verify that the correct Java applet is running. We model mistakes in this step as if Step 2 is ignored.
Likewise, it is hard to notice that the Java security dialog did not appear.
In sum, we get three variants, the base protocol and two variants

Variant	Step 2	Step 1
Base	×	×
1	–	×
2	–	–

Figure 2: The BankID protocol variants.

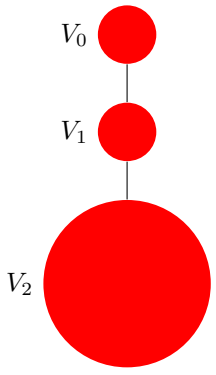

Figure 3: BankID variant security summary.

The Java security dialog allows a user to decide if an applet should be given access to the system or not. It is not intended to authenticate any part of a web page before entering sensitive information into it. And indeed, the Java security dialog is easily forged by a phishing site.

This means that even if the user is verifying the content of the dialog, the information he verifies may be forged, and therefore, the verification may be useless.

We conclude that even if both checks are performed as correctly as is reasonably possible, this protocol does not provide a defence against phishing attacks. It has no security.

We note that even if the Java security dialog was not easily forged, it is hard to decide if the password entry field on the screen belongs to the Java applet that caused the Java security dialog.

The two variants merely make phishing attacks slightly easier. Though we note that Variant 1 is somewhat dangerous, as the user grants an unverified Java applet full access to his computer. If a suitable applet can be found, this is a nice way to compromise the user's computer.

We expect that the majority of users will actually use Variant 2. If there is no Java security dialog, they will proceed, but if there is a Java security dialog, the user will allow any Java applet access.

This analysis is summarized in Figure 3.

3.2 BankID på Mobil

BankID på Mobil is an electronic identification system jointly developed by Norwegian banks. It is distinct from the above mentioned BankID system. The system is not public, and there is no reason to believe that the cryptographic protocol in use is as secure as the one given in the appendices.

The system uses software and keys installed on the SIM card inside the user's phone. The software communicates with a central infrastructure through special SMS messages, and with the user through the SIM toolkit system.

The relying party asks the user for his username and forwards this to the banks' central infrastructure. The infrastructure sends a nonce to the relying party, which displays it on the web page. The infrastructure also sends the same nonce and the relying party's name to the phone, which shows it to the user.

The user checks that phone displays the name of the relying party, and that the relying party and the phone both display the same nonce. If this holds, the user accepts by pressing a button. The phone informs the infrastructure that the user has accepted. The infrastructure then informs the relying party, which accepts the user as logged in.

Strictly speaking, after the user has accepted by pressing a button on the phone, the user must enter a password. In our model, this password has no effect. We shall therefore ignore this password.

Analysis The user interface presents the two pieces of information to be verified above an easy-to-press ok button. The nonce is displayed (at least on some

Base

1. Enter username.
2. Verify that the relying party's name displayed by the phone is correct.
3. Verify that the nonce displayed on the phone matches the nonce displayed on the computer.
4. Click ok on the phone.

Variants It is easy to forget to verify the relying party's name displayed by the phone, and it may be hard to correctly verify the name (Step 2). It is easy to forget to verify that the nonces match (Step 3).

Variant	Step 2	Step 3
Base	×	×
1	–	×
2	×	–
3	–	–

Figure 4: The BankID på Mobil protocol variants.

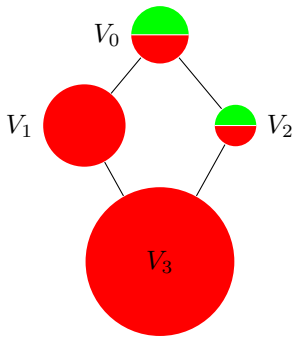

Figure 5: BankID på Mobil variant security summary.

phones) in a larger font than the relying party's name.

It is very easy for the user's thumb to press the ok button before the user's mind has even considered the information displayed, thereby making both mistakes.

The visual presentation will tend to emphasize verification of the nonce (which is also displayed prominently on the computer screen). It is reasonable to assume that this will tend to reduce cognitive effort spent on verifying the relying party's name.

Note that even if the user correctly verifies both the relying party's name and the nonce, there is no guarantee that the user's web browser is showing the relying party's web site. Therefore, this system does not establish a secure channel, not even when the local network is secure.

If the user correctly verifies the relying party's name, we know that the user intended to login with the relying party. Therefore, we have established correct intention.

The above analysis may suggest that the nonce verification is completely useless. This is not the case. There are certain specific attacks, especially when the local network is compromised, that nonce verification will stop. Unfortunately, as described above, there are also attacks that nonce verification does not stop.

This analysis is summarized in Figure 5.

3.3 Smart Card System

There are two widely deployed smart card[4] systems in Norway, and only one is directed at consumers. While the relevant software governing the system is widely distributed both to users and relying parties, the functioning of the system is still claimed to be secret. We know that the system is not more secure than the generic smart card system that we describe and analyse below. In other words, the following analysis is only an upper bound on the security of the deployed system.

The generic smart card system does not use client side authentication in TLS to prove possession of the client secret key. Instead, a secure TLS channel is established with the relying party, after which a signed Java applet is responsible for facilitating a conversation between the smart card and the relying party. This conversation essentially authenticates the already established TLS channel.

When the user wants to login, he inserts the smart card into a smart card reader, and the computer software does the rest. Usually, a smart card requires a password before it will do anything with its secret key. In our model, this password has no security effect, since we assume that the user has full control over the smart card. We shall therefore ignore the password.

Analysis In principle the user must verify that the Java applet is the correct applet. Otherwise, potentially, a phishing attack using a suitable Java applet

[4]One system is not using smart cards, but USB tokens. This is not important.

Base

1. When asked to insert smart card, insert smart card.
2. Click to log in.

Variants While smart card support costs suggest that users make mistakes when using smart cards, these mistakes do not have any effect on security. Instead, their effect is on functionality and is user-observable.

Figure 6: A generic smart card system.

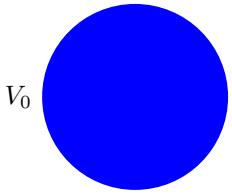

Figure 7: Smart card security summary.

could compromise the user's computer and subsequently jeopardise the protocol. However, every browser allows the user to mark certain sources as trusted for Java applets. The intention is that the user does this once, then never again. This might give the user some protection against malicious Java applets. Therefore, we ignore such attacks.

If the computer is honest, the analysis in the appendices shows that the protocol establishes a secure channel, even if the local network is compromised. This also establishes correct intention by default.

We note that if the user's computer is compromised, there is no security.

This brief analysis is summarized in Figure 7.

We note that under a stronger notion of correct intention, a generic smart card system may or may not provide some protection when the user's computer is honest.

3.4 MinID

MinID is the Norwegian government's light-weight login system. Its origins are in a system used to authenticate online tax returns, and certain parts of the system are still run by the tax authorities. The system is based on a password strengthened by one-time codes sent via SMS to the user's mobile phone[5].

The user clicks a MinID login button on the relying party's web site. This

[5]It is also possible to use the MinID system with a list of one-time codes printed on paper. We do not consider this system

directs the user's web browser to a dedicated MinID login page. There the user enters his username and password. The MinID system sends a one-time code along with the relying party's name to the user's phone via SMS. The user enters the one-time code into the MinID login page, after which the user's web browser is redirected to the relying party's landing page.

The MinID protocol and its variants are given in Figure 8. We note that none of the three security mechanisms detailed are mentioned in the system's user's guide.

Analysis Browsers usually display some indication that TLS is being used. Research suggests that users may misinterpret these browser security indicators. It seems reasonable to assume that it is easy to forget to check for TLS or mistakenly believe that TLS is turned on when it is not turned on.

Even though modern browsers have made address verification much easier by emphasizing the host name part of the URL, it is still reasonable to believe that users may either forget to verify the address or mistakenly accept an incorrect address.

Finally, while the wording in the SMS message is suboptimal, it is fairly easy to notice if the relying party's name is incorrect, since the user has to look at the phone to read the one-time code. Unfortunately, since the relying party's name is nearly always correct, experienced users of the system will be able to read only the required part of the SMS message, namely the one-time code, without reading the rest of the message. We must therefore expect that a rather large fraction of users will not notice if the relying party's name is incorrect.

Note that a more dynamic modelling would be more realistic. It is not the case that a fraction of the user population always get this check wrong, while the remaining users always get it right. You must expect that some users will usually get it right, while others will usually fail. However, the aggregate result should still be reasonable.

If the user correctly verifies the relying party's name, the protocol ensures correct intention against any attacker.

If the user correctly verifies the address and the local network is not compromised, we know that the user gives his username, password and one-time code to the correct web site. Therefore, the protocol establishes a secure channel as long as the local network is honest.

If the user correctly verifies the address and that TLS is turned on, and the user's computer is honest, then we know that the user gives his username, password and one-time code to the correct web site. Therefore, the protocol establishes a secure channel even if the local network is compromised.

If the user does not verify that TLS is turned on and the local network is compromised, then the adversary could impersonate the MinID login page. In this case, no secure channel would be established, but if the user verified the relying party's name, correct intention would still be established.

This analysis is summarized in Figure 9.

Base protocol

1. Click the MinID login button.
2. If the next page is not secure, stop.
3. If the next page does not have the address `minid.difi.no`, stop.
4. Enter username and password.
5. Wait for SMS with one-time code and the name of the relying party.
6. If the name of the relying party is incorrect, stop.
7. Enter one-time code.

Variants It is easy to forget to verify that TLS is used (Step 2). Likewise, it is easy to forget to verify the address, and it is hard to verify the address correctly (Step 3). It is easy to forget to verify the relying party's name in the SMS message, and it may be hard to correctly verify the name (Step 6).

Variant	Step 2	Step 3	Step 6
Base	×	×	×
1	–	×	×
2	×	–	×
3	×	×	–
4	–	–	×
5	–	×	–
6	×	–	–
7	–	–	–

Figure 8: The MinID protocol variants.

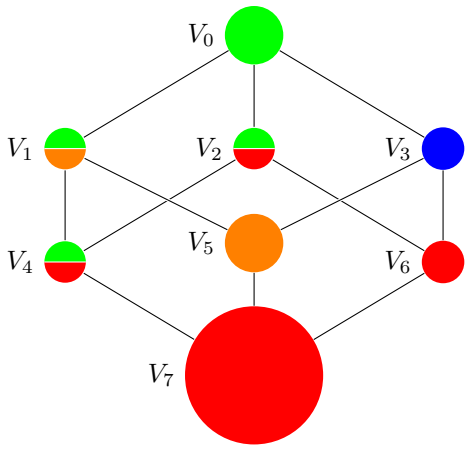

Figure 9: MinID variant security summary.

4 Theory studies

We now consider the following question:

How to maximise security given the already deployed infrastructure?

Our goal is to give as many users as possible as much security as possible. We also want to give users willing to make an extra effort more security.

Since a smart card infrastructure is already deployed, we may consider systems based on smart cards. However, we cannot only consider such systems, since many modern devices cannot handle smart cards.

These schemes are meant for the general population. We cannot assume that users in general are especially skilled, that they undergo training or that they install software.

However, some fraction of users will be willing to do extra work or change their habits to get extra security. One thing that many people are able to do is to have their web browser remember web site addresses, so-called bookmarks, and support for bookmarks is universal. Many web browsers now also have support for easy-to-install add-on software, so-called browser extensions. Unfortunately, an extension written for one browser will not work on a different browser, which means that deploying extensions may be expensive and complicated.

To summarize, any proposed system must work with the software installed by default on modern computers, that is, a reasonably modern web browser. Motivated users will certainly be able to add bookmarks to their browser, and some will also be willing to add browser extensions. We may also use an existing smart card infrastructure.

4.1 Improved MinID

We first consider a protocol that is fairly close to the MinID system, but tries to allow users to protect themselves against phishing and local network attacks. The idea used is not new, see for instance Adida's BeamAuth [2] or `bookmarkid.net`.

First, we insert one extra step into the MinID protocol:

3b. Click the "Start login" button.

This extra step is needed for communication between the relying party and the infrastructure[6].

The improved protocol departs from the MinID protocol when the "Start login" button appears. Instead of clicking the button, the user invokes a previously stored bookmark that takes the web browser to the username/password entry page. Invoking a bookmark will tend to destroy session information, but this can be preserved using so-called cookies.

The resulting protocol is shown in Figure 10.

[6]To communication across the bookmark invocation, we use cookies. This extra step is needed because some browsers prohibit so-called third-party cookies. While cookies could be set while the username/password entry form is displayed, that will encourage the user into making mistakes.

Base

1. Click the MinID login button.
2. Use the "Improved MinID" bookmark.
3. Enter username and password.
4. Wait for SMS with one-time code and the name of the relying party name.
5. If the name of the relying party is incorrect, stop.
6. Enter one-time code.

Variants Unless carefully trained, it is easy to click the "Start login" button instead of using the bookmark, turning Step 2 into

2. Click the "Start login" button.

It seems unlikely that a bookmark user will notice if TLS is not used or if the address is incorrect.
It is easy to forget to verify the relying party's name in the SMS message, and it may be hard to correctly verify the name (Step 5).

Variant	Step 2	Step 5
Base	×	×
1	×	–
2	–	×
3	–	–

Figure 10: Improved MinID protocol.

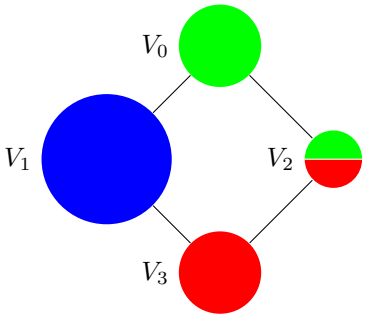

Figure 11: Improved MinID summary.

Analysis If the user correctly verifies the relying party's name, the protocol ensures correct intention against any attacker.

If the user correctly uses the bookmark and the computer is honest, we know that the user gives his username, password and one-time code to the correct web site. Therefore, the protocol ensures a secure channel in the presence of a compromised local network.

If the user does not invoke the bookmark, the protocol does not ensure a secure channel. If the user also fails to verify the relying party's name, the protocol does not ensure correct intention.

These results are summarized in Figure 11.

Discussion Using the bookmark ensures that TLS is turned on and that the address of the username and password entry page is correct. Compared to the MinID protocol, we have replaced two verifications by one action. Unfortunately, it is still easy to make a mistake and click the "Start login" instead of using the bookmark.

It is possible to allow the user to train himself to avoid clicking the "Start login" button. By checking referrer headers, the identification system can detect if the user arrived at the login page by using a bookmark or clicking the "Start login" button. The user can ask the identification system either to delay login if the "Start login" button was clicked, or even to refuse login, forcing the user to go back and use the bookmark. We conjecture that a user exposed to such "pain" will quickly learn to avoid the "Start login" button. Note that such measures will have no effect in the event of an attack.

A somewhat more complicated option (which may not be available on all web browsers on all devices) is to install a browser extension that recognizes the web page with the "Start login" button and simply removes it. The user will quickly forget that the "Start login" button was ever there. It is not clear which approach will best train the user to defend against phishing attacks.

Of course, if we consider browser extensions, we may as well construct an extension that recognizes the MinID login button on the relying party's web page, and offers a login button as part of the browser chrome. This could significantly increase security, since the scope for mistakes will be even smaller, but unfortunately, not every browser supports browser extensions, and they are typically not portable between browsers. However, relatively modest effort could provide a significant increase in security.

4.2 Improved Smart Card System

The generic smart card system discussed in Section 3.3 can be improved by adding better verification of the relying party's name. This can be done by passing a one-time code to the user's phone along with the relying party's name.

The protocol first proceeds as the generic smart card protocol in Section 3.3. After the smart card authentication is complete, the relying party passes its challenge together with the user's signature to an infrastructure. The infrastructure sends an SMS message to the user's phone with the relying party's name and

a one-time code. The user gives the one-time code to the relying party, which passes it on to the infrastructure. The infrastructure then replies with its own signature on the user's signature, after which the relying party accepts.

The user's part of this protocol is detailed in Figure 12.

Analysis If the computer is honest, the analysis in Section 3.3 still applies. This means that the protocol establishes a secure channel, and therefore also correct intention by default.

If the user correctly verifies the relying party's name given, the protocol ensures correct intention against any attacker.

The result is summarized in Figure 13.

5 Application To Risk Analysis

In the following, we use a simple and naive model of risk as the sum over all possible attacks of the consequence of an attack multiplied by the probability of that attack happening (where sum and multiply may not be ordinary addition and multiplication).

Naively and as a rough approximation, the probability of a given attack happening can be taken as the product of the probability that an attempted attack succeeds and the probability that someone at some point in time wants to attempt the attack. Unfortunately, the latter term is often impossible to estimate, but 1 may be a reasonable approximation. In other words, unless we have good reason not to, we shall usually approximate the probability of an attack happening by how likely it is that an attempted attack succeeds.

How do we translate the results of the previous sections into likelihoods? It seems reasonable that an attempted attack against a user of a system that is vulnerable to phishing attacks is highly likely to succeed, while an attempt against a system that ensures correct intention against even compromised computers is much less likely to succeed.

5.1 Internet banking

The obvious security goal of any internet banking application should be the following:

> For any transaction that is completed on behalf of some user, the user intended that transaction to happen. If the user's computer is honest, the user's account transcripts remain private.

First consider privacy. Note that the moment an adversary manages to log into an internet bank as a given user, quite a lot of private information will leak. However, one may consider this as of little consequence[7].

[7] Some people would certainly consider their account transcripts very private and be embarrassed by any leak. However, when you consider the degree of protection given to account transcripts by banks, they clearly consider privacy to be of little importance.

Base

1. Insert the smart card.
2. Click to log in.
3. Wait for SMS with relying party name and one-time code.
4. If the relying party name in the SMS message is not correct, stop.
5. Enter the one-time code.

Variants It is easy to forget to verify the relying party's name in the SMS message. It may be hard to correctly verify the name (Step 4).

Variant	Step 4
Base	×
1	–

Figure 12: An enhanced smart card system.

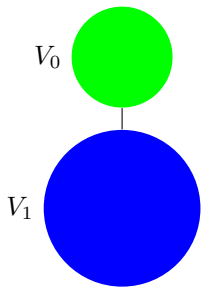

Figure 13: Enhanced smart card system summary.

Second, consider transactions. The main point is that many transactions are reversible, and even if transactions cannot be reversed (e.g. because the money has left the banking system), they can be effectively reversed by reimbursing the user. This leads to a loss for the bank and some inconvenience for the user. We may consider this the cost of internet banking.

We therefore arrive at the following (more modest) security goal for the internet banking application:

> The total cost of internet banking should be small.

A typical internet bank has roughly speaking two security systems. The first line of defence is the login system. Many internet banks also use some form of transaction authorization, but in Norway, anyone who can breach the login system can usually also breach the transaction authorization system.

The second system is not visible to the user, and is based on monitoring the application. There are a number of signals that can be monitored. Some examples are:

- When accepting a given user as logged in via a given computer, the internet banking application can check if the computer has been used by the user for internet banking before. The obvious technique is to check for a previously set cookie. If the application has sufficient access to the user's computer, more complex methods can be used.
- If the internet banking application has sufficient access to the user's computer, it can inspect the computer, looking for signs of compromise.
- Network surveillance can reveal if a user's computer is in communication with so-called command and control servers used to control networks of compromised computers. Internet banks obviously cannot do this, but other organizations may share data with the banks.
- The internet banking application can measure the time between user actions, as well as other patterns in user interaction. Sufficiently crude malware could have recognizable signatures.
- The banks can monitor patterns in transactions, not just for one user, but for all users, and not just for one bank, but for many banks.

Few of these signals will be sufficient to prove that a transaction is fraudulent (with high probability), but combinations of these signals may be sufficient to detect almost all fraudulent transactions with a sufficiently low false alarm rate. When the monitoring system detects a possibly fraudulent transaction, it is usually easy to contact the user out-of-band and either cancel or reverse the transaction.

Even if the system does not detect the fraudulent transaction, there is significant likelihood that the user will detect it eventually. In this case, the transaction can often be reversed, and only if it cannot be reversed must the user be reimbursed for his loss.

Note that such a system is not sufficient to reach our modest security goal. It may be unable to prevent old-fashioned vandalism, for instance when the adversary inserts fraudulent transactions similar to the user's real transactions. If this affected sufficiently many users and transactions, the cost to the banks could be significant.

The probability of someone wanting to steal money from internet banks is obviously high. Likewise, we see that the probability of failure in the login system is high (see Section 3.1 and Section 3.2 for a discussion of the login systems used; the probability of failure could be significantly lower if the internet banking login systems were not used as federated identity systems). However, there are other security systems limiting the consequences of these failures.

And while there are vandalism-type attacks that may have more serious consequences if they affect many users, one could fairly argue that the probability of such an attack is lower, reducing risk.

Obviously, there are many more contributions to total risk in an internet banking application, but the risk related to these two issues should dominate the total risk.

To summarize, it seems reasonable to claim that the overall risk associated with internet banking is low, and certainly low relative to the benefit derived from internet banking.

5.2 Medical data

My Prescriptions, `mineresepter.no`, is a web site run by Norwegian health authorities. It is part of a larger system for managing prescriptions, but we only consider the web site `mineresepter.no`. Its only function is to show people what prescriptions they have received in the past year. It is quite simple: log in, and it shows you a list of prescriptions. Relatively few people use the web site.

The security goal for such a web site is clear:

> If `mineresepter.no` releases medical information about a user into a channel, then the user intended to contact `mineresepter.no` and the channel is the secure channel established by the user.

Unlike for internet banking, gentle probing suggests that there are few, if any, secondary lines of defence. The login system seems to be the only significant security system. That is, if the login system fails, application security also fails. Obviously, application security can fail even if the login system does not fail, but it seems reasonable that the probability that the login system fails will dominate in any risk analysis.

Unlike account transcripts, there seems to be general agreement that medical information should be kept private, and that any compromise is therefore very serious.

Prior to 2012, login to the system required the use of a smart card system. If the analysis in Section 3.3 applies, the smart card system is sufficiently secure whenever the user's computer is honest. Compromise will then only happen if

the user's computer is compromised. This probability might therefore be only modest (except for skilled attackers with specific targets). However, because the consequences of compromise are severe, the risk should probably be classified as high.

If a system such as the one in Section 4.2 was used, we see that this would probably reduce the probability of a successfull attack even further. While this will reduce the overall risk, we might still want to classify the risk as high because of the severe consequences.

Towards the end of 2012, the BankID system from Section 3.1 was added as an alternative login system. Since BankID is highly insecure, the probability of compromise of `mineresepter.no` is now high. The risk must then be classified as very high.

We found no effective method for opting out of the web site, so non-users of the system have no sensible way to mitigate the risk they are exposed to.

While some people might derive some benefit from `mineresepter.no`, it seems that a significant proportion of the population will not derive any real benefits from this web site. In sum, `mineresepter.no` seems to be a web site that exposes the Norwegian population to a significant risk without any corresponding benefit.

We should also note that the Norwegian authorities have used a single-sign-on solution to login to many public web sites, `mineresepter.no` included. This means that using better login systems may not reduce the overall risk.

References

[1] Ben Adida. *Advances in cryptographic voting systems*. PhD thesis, Massachusetts Institute of Technology, Cambridge, MA, USA, 2006. AAI0810143.

[2] Ben Adida. BeamAuth: two-factor web authentication with a bookmark. In Peng Ning, Sabrina De Capitani di Vimercati, and Paul F. Syverson, editors, *ACM Conference on Computer and Communications Security*, pages 48–57. ACM, 2007.

[3] R. Canetti. Universally composable security: A new paradigm for cryptographic protocols. Report 2000/067, IACR ePrint Archive, 2005.

[4] Carl Ellison. Ceremony design and analysis. Cryptology ePrint Archive, Report 2007/399, 2007. `http://eprint.iacr.org/`.

[5] Yngve Espelid, Lars-Helge Netland, André N. Klingsheim, and Kjell Jørgen Hole. A proof of concept attack against norwegian internet banking systems. In Gene Tsudik, editor, *Financial Cryptography*, volume 5143 of *Lecture Notes in Computer Science*, pages 197–201. Springer, 2008.

[6] Yngve Espelid, Lars-Helge Netland, André N. Klingsheim, and Kjell Jørgen Hole. Robbing banks with their own software—an exploit against norwegian

online banks. In Sushil Jajodia, Pierangela Samarati, and Stelvio Cimato, editors, *SEC*, volume 278 of *IFIP*, pages 63–77. Springer, 2008.

[7] Kristian Gjøsteen. Weaknesses in BankID, a PKI-substitute deployed by Norwegian banks. In Stig Fr. Mjølsnes, Sjouke Mauw, and Sokratis K. Katsikas, editors, *Proceedings of EuroPKI 2008*, volume 5057 of *Lecture Notes in Computer Science*, pages 196–206. Springer, 2008.

[8] Kristian Gjøsteen. Analysis of an internet voting protocol. Cryptology ePrint Archive, Report 2010/380, 2010. `http://eprint.iacr.org/`.

[9] Chris Karlof, Naveen Sastry, and David Wagner. Cryptographic voting protocols: a systems perspective. In *USENIX Security Symposium*, pages 33–50. USENIX, 2005.

[10] Chris Karlof, J. D. Tygar, and David Wagner. Conditioned-safe ceremonies and a user study of an application to web authentication. In Lorrie Faith Cranor, editor, *SOUPS*, ACM International Conference Proceeding Series. ACM, 2009.

[11] Alma Whitten and J. D. Tygar. Why Johnny can't encrypt: A usability evaluation of PGP 5.0. In *8th Usenix Security Symposium (Security99)*, August 1999.

A Full Protocols

In this section, we give reasonable interpretations of the protocols involved in the case studies from Section 3. While we cannot guarantee that the secret protocols actually used in the case studies are as good as these protocols, it is at least possible that the deployed systems can achieve the security claimed.

We do not provide protocols and analysis for the theory studies from Section 4. For straight-forward implementations, these are very similar to the protocols from Section 3. But in reality, any future password-based protocols should have much stronger security. In particular, there should be some security against corrupt infrastructures.

The protocols are described in terms of message sequence charts. Solid arrows denote communication through TLS channels[8], while double arrows denote communication through channels that are assumed[9] secure.

[8] Note that it is not possible to equate TLS channels with TLS sessions, since TLS sessions may not be sufficiently persistent. Establishing and maintaining TLS channels is therefore non-trivial. Typically, the server stores a random value in a cookie. This value (or rather, information tied to the random value) must often be repeated in the requests sent to the web server to ensure that the request originated with the correct channel.

[9] When considering the channel from infrastructures to mobile phones, this assumption is clearly debatable. But still, for the time being, attacks against the mobile channel are uncommon and unrealistic for many adversaries.

Note that these protocols are abstractions of the real-world interaction. Between computers, the abstractions are close to the real-world action, but between humans and computers, the abstractions only indicate the corresponding real-world interaction.

For instance: When the user's computer displays a relying party's web page, the computer is giving the user a lot of information, and only some of it is relevant to the protocol. One piece of information that is relevant for the protocol is the name of the relying party. Typically, this is communicated to the user in two ways: through the browser's address bar and through the visual layout of the web page itself.

Obviously, any web page can adopt any visual layout. Only the address bar is a reliable signal. However, users often rely more on the visual layout than the address bar, and this is one source of mistakes in common web protocols.

When describing protocol messages from a computer to a user, we adopt the convention that in $m \mid r$, m denotes the (unreliable) information conveyed by the web page visual layout, and r denotes the (reliable) information displayed by the browser. Note that we only describe such information when it is required by the protocol.

We should also note that our protocols will sometimes mention some party's name both in the reliable and the unreliable parts of the message. The reliable signal will typically refer to an URL, which is only sometimes human-readable, and often only vaguely related to the human-readable name from the unreliable signal. This implies among other things that humans cannot reliably verify URLs, unless they know the URL very well.

A.1 BankID

A plausible protocol for BankID is given in Figure 14. We note that in the real protocol, the infrastructure stores one signing key per user and uses the user's key to sign the message, not its own. This does not have any real effect on security, but it significantly complicates the implementation and its analysis.

As the attack in Figure 15 shows, the BankID protocol does not have any security. We stress that this attack does not depend on the computer–relying party–infrastructure part of the protocol. The attack exploits the user–computer interaction only, which we know we have modeled correctly.

Weak Positive Results It is possible to find weak positive security results for this protocol if we strengthen our assumptions. Suppose every relying party is honest, and the user correctly verifies that his computer is communicating with a relying party. Suppose also that the user's computer is honest.

If the relying party accepts that the user has logged in, it knows that the infrastructure has generated a signature of the required form. This means that there is an infrastructure session agreeing with the relying party about the challenge, the user's identity and the relying party's identity.

Since the infrastructure session will only generate the signature if it has seen the correct password and one-time code, we know that the user has requested a

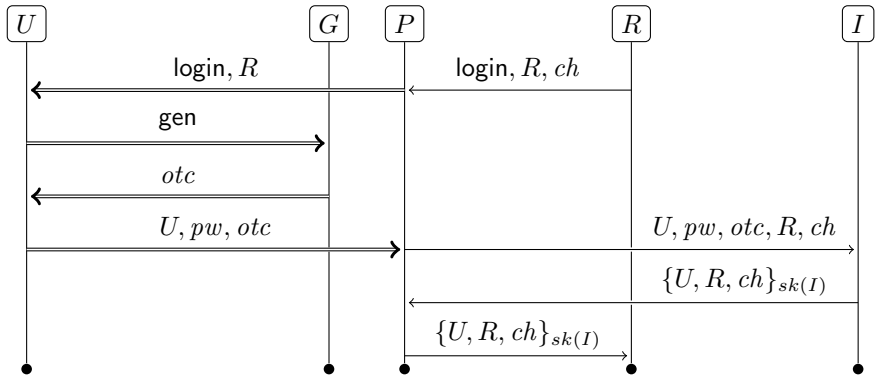

Figure 14: A protocol of equivalent functionality to the BankID protocol. The participants are: a user U, a one-time code generator G, a computer P, a relying party R and a central infrastructure I.

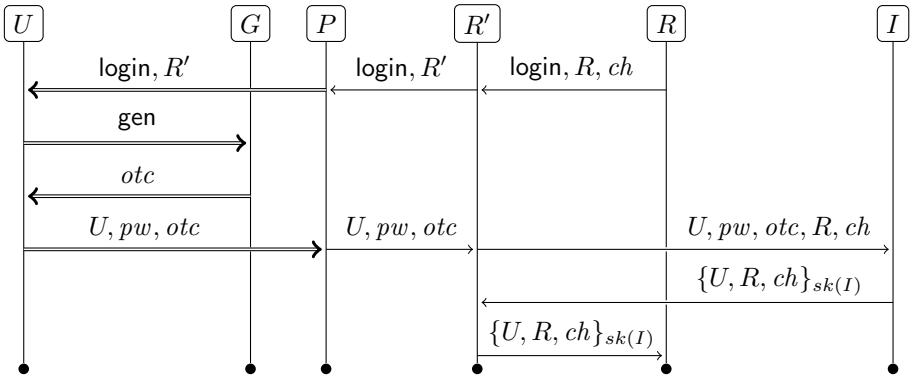

Figure 15: An attack by a corrupt relying party R' (or some web site merely impersonating a relying party). The attacker uses the stolen information to log in with an honest relying party R.

one-time code from the one-time code generator and given the one-time code and his password to a computer. Since the user is only talking to honest computers, and verifies that the computer is talking to an honest relying party, we know that the infrastructure is talking to an honest computer, and that this honest computer agrees with everyone about who the relying party is.

We conclude that under these strong assumptions, this protocol establishes a secure channel and ensures correct intention. Obviously, the stronger assumptions are completely unrealistic for a federated identity system, which is what BankID is today. However, BankID has its origins in the banking system. As a federated identity system for the financial system only, it might have been possible to justify the stronger assumptions. Of course, our variant analysis would still show that the system is very brittle.

We should also stress that the exact BankID protocol is not public, so we cannot guarantee that the real protocol achieves the same weak security as our example protocol given in Figure 14.

A.2 BankID på Mobil

A plausible protocol for BankID på mobil is given in Figure 16. It is worth noting that the user's signing key is stored inside the mobile phone SIM card, and that communication between the central infrastructure and the SIM card happens via SMS messages that are given extra protection.

As the attack in Figure 17 shows, the BankID på mobil protocol does not establish a secure channel. Again, this attack does not in any way depend on the computer–relying party–infrastructure–mobile phone part of the protocol. The attack exploits the user–computer–mobile phone interaction only, which we know we have modeled correctly.

Weak Positive Results It is possible to find weak positive security results for this protocol if we strengthen our assumptions. Suppose the user verifies that the URL belongs to the relying party and that TLS is turned on.

Suppose the user only talks to honest computers. If the relying party accepts that the user has logged in, it knows that the user's phone has generated the correct signature. Therefore, the phone has displayed the relying party's name and a nonce.

The user accepted the name and the nonce, which means that the user's computer has shown the user the nonce, and the user has verified that TLS is turned on and the URL shown by the computer belongs to the relying party.

Since the nonce included in the phone's signature matches the nonce seen by the user on the computer, the computer's session is indeed the relying party's session. This means that a secure channel has been established.

For a general federated identity system, such URL verification is unrealistic. But for a very restricted set of relying parties, such as a small number of financial web sites, it might be possible to argue that URL verification is possible. However, our variant analysis would still show that the system is very brittle.

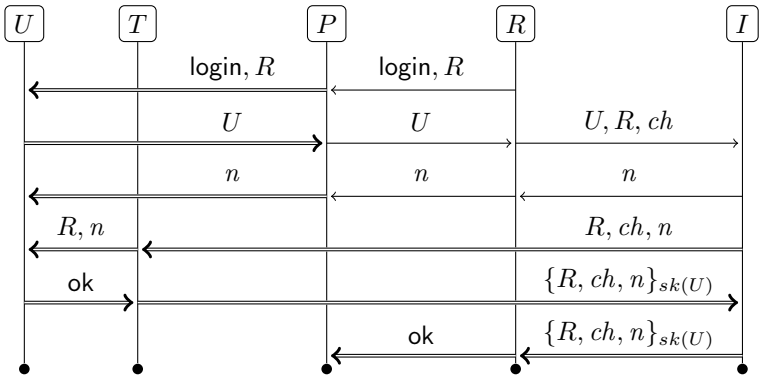

Figure 16: A protocol of equivalent functionality to the BankID på Mobil protocol. The participants are: a user U, a mobile phone T, a computer P, a relying party R and a central infrastructure I.

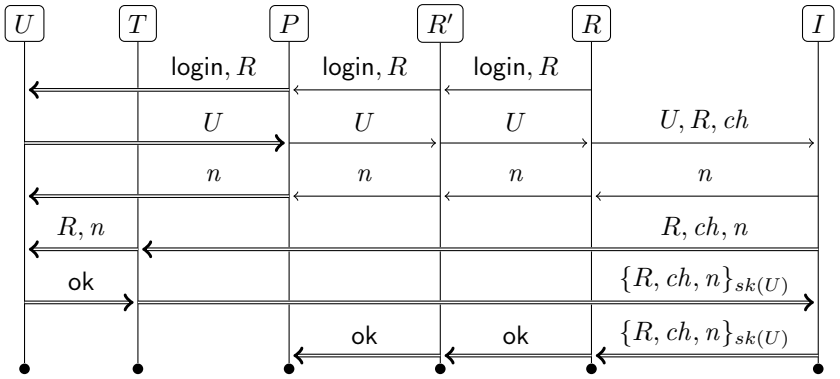

Figure 17: An attack by a web site R' impersonating an honest relying party R. Correct intention is satisfied, but a secure channel has not been established.

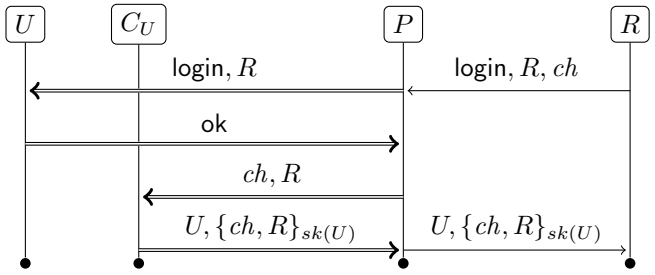

Figure 18: Smart card protocol. The players are: a user U, a smart card C_U belonging to U, a computer P and a relying party R.

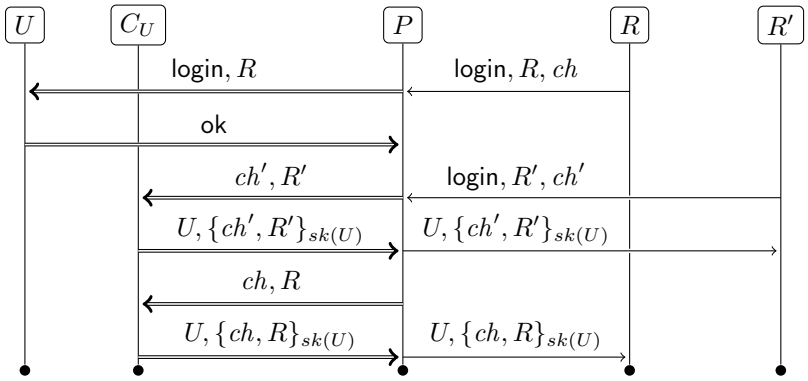

Figure 19: A corrupt computer attacking the smart card protocol and surreptitiously logging in with a second relying party R' while the user is logging in with the intended relying party R.

A.3 Smart Card Protocol

The protocol is given in Figure 18. The general assumption is that a user's smart card is only allowed to talk to a computer if the user is talking to the computer. Suppose further that the user only talks to honest computers.

If the relying party accepts that a user U has logged in, we know that the user's smart card has received the challenge and the relying party's name. Since the smart card must have talked to a computer, we know that it received the challenge and the relying party's name from an honest computer. Since the computer is honest, it must have established a secure channel to the relying party and received the challenge through that channel. Since the relying party does not reuse challenges, the honest computer's session agrees with the relying party's session.

This means that we have established a secure channel, and since the computer is honest, correct intention by default.

A.4 MinID Protocol

The protocol is given in Figure 20. Note that for the purposes of this analysis we assume that one-time codes are unpredictable and do not repeat, that is, they are equivalent to nonces. This is reasonable. Suppose the relying party accepts that the user has logged in. First of all, there is a computer talking to the relying party over a TLS channel.

Next, since the relying party verifies the signature, there must also be an infrastructure session agreeing with the relying party about the challenge, the user and the relying party. There is also a computer talking to the infrastructure.

Since the infrastructure receives the one-time code it sent to the user's phone, there must be a user executing the base protocol that has seen the SMS message with the one-time code, which also contains the relying party's name. This establishes correct intention.

The user must be talking to a computer. If that computer is corrupt, we are done. Assume therefore that the voter is talking to an honest computer. Since the voter verifies the URL and that TLS is turned on, we know that the voter's honest computer has a session shared with the infrastructure.

We first argue that the computer session talking to the user is the same as the computer session talking to the infrastructure. We know that the two sessions agree on the one-time code. The infrastructure sends the one-time code to the user's phone, which only reveals it to the user, which in turn only passes it on to the computer talking to the user. This computer learns the one-time code from the user and immediately sends it on to the infrastructure. In sum, the one-time code can only be known by one computer, and this honest computer is talking to both the user and the infrastructure.

Next we argue that the computer session talking to the infrastructure is the same as the computer session talking to the relying party. Observe that the two computer sessions share the same signature. This means that they also agree on the relying party's name, the user's name and the relying party's challenge. The properties of TLS channels ensures that for the computer session talking to the relying party, the message specifying the challenge really came from the relying party. The only way the challenge can leave that computer session is if that session sends it to the infrastructure. The infrastructure will in turn never disclose the challenge (except possibly as part of a signature sent through the same TLS channel). This means that since the two computer sessions agree on the challenge, they must be identical.

In conclusion, the honest computer has established a secure channel to the relying party.

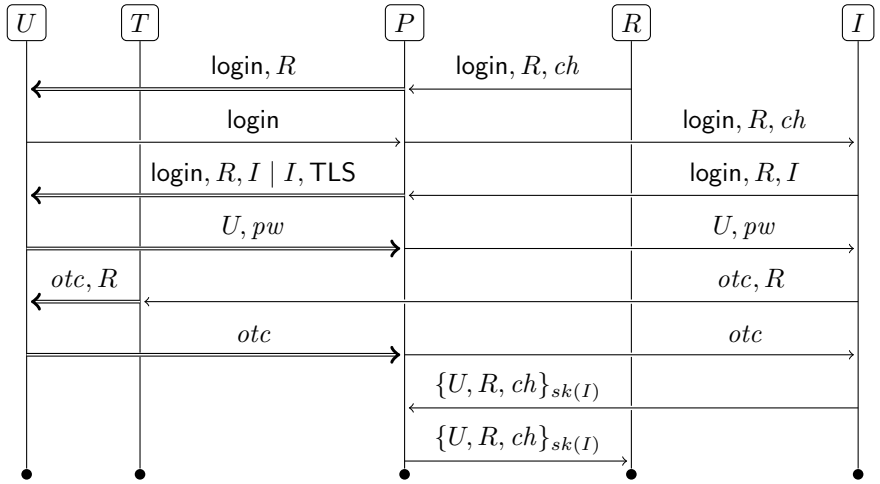

Figure 20: A protocol of equivalent functionality to the MinID protocol. The participants are: the user U, his mobile phone T, his computer P, the relying party R and the infrastructure I.

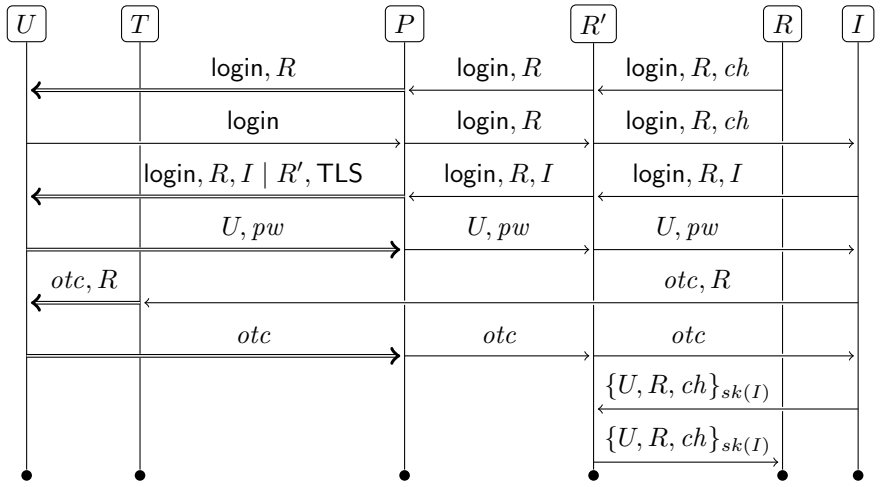

Figure 21: A phishing attack by a corrupt relying party R' impersonating an honest relying party R against a user of MinID, where the user forgets to verify the URL (Variant 2).