Paper II: Case Study: Online Banking Security

Case Study: Online Banking Security

Kjell Jørgen Hole, Vebjørn Moen, and Thomas Tjøstheim

Abstract

This article argues that Norwegian Internet banking was vulnerable to computer attack in the period 2003–2004 by describing scenarios for potential attacks.

1 Introduction

Many Norwegians turned to the Internet for personal banking services during 2003 and 2004. Because of the Norwegian banks' security-by-obscurity policy, the online customers knew very little about the true level of security offered by the Internet banking systems; the banks naturally maintained that their systems were very secure.

In this article, we discuss simple—but powerful—strategies for attack on selected Norwegian online banks, and discuss why these attacks were theoretically possible during much of 2003 and 2004. The scenarios are based only on publicly available information found on the Internet. None of the attacks were carried out in practice. Instead we presented our findings to the Norwegian government agency overseeing the national banking industry. We also made a sustained effort to directly inform the banks most vulnerable to the attacks.

Our main reason for making public this account is to contribute to the development of more secure Internet banking systems. For the same reason, we also speculate why the banks developed Internet banking solutions with bad security in the first place. Finally, we suggest ways in which universities can teach students to design more secure alternatives.

Many Norwegian Internet banks have long required their customers to log into online accounts using a Social Security Number (SSN) or an account number, as well as a Personal Identification Number (PIN). While the SSN and the account number aren't considered secrets, the PIN is only known to the customer. If a customer tries to log into an account with the correct SSN (or account number) and a wrong PIN enough times (usually between 3 and 5), the bank will temporarily deny the customer any further access to the account.

As we'll see, it was possible for a cracker to launch an attack against an Internet bank by combining a simple brute-force attack with a Distributed-Denial-of-Service (DDoS) attack [1] that exploits the bank's login procedure. A successful combined attack gains access to a small number of accounts and—at the same time—prevents a large number of legitimate customers from accessing their accounts. We'll show that the combined attack is also effective when the bank's customers use PIN calculators, also called PIN cards or (hardware) tokens, to generate dynamic PINs.

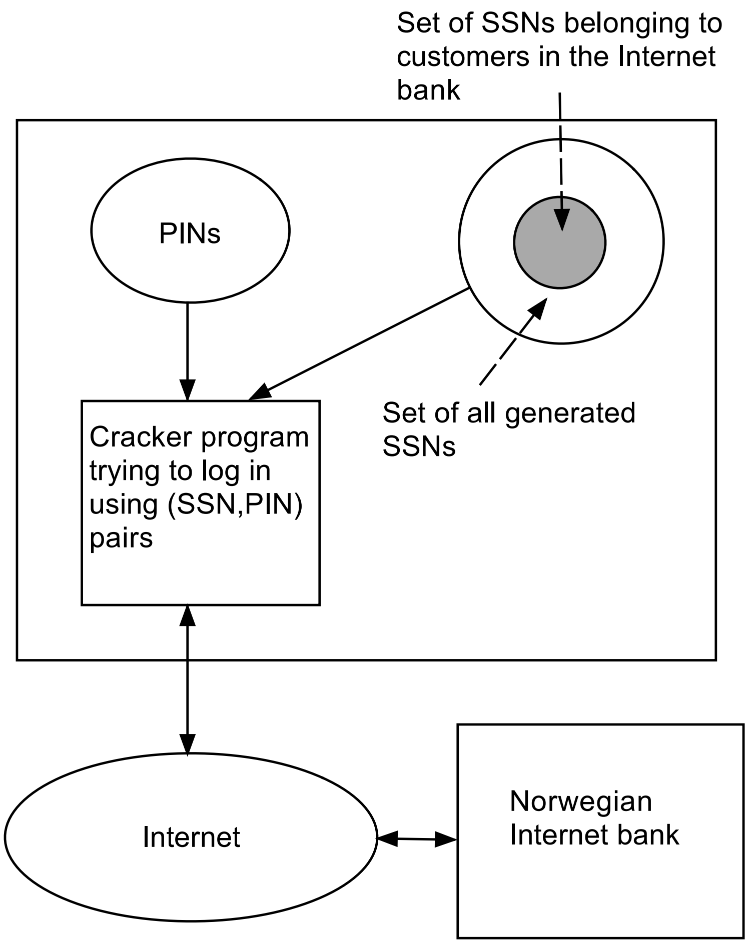
Set of SSNs belonging to customers in the Internet bank
PINs
Cracker program trying to log in using (SSN,PIN) pairs
Set of all generated SSNs
Internet
Norwegian Internet bank

Figure 1: Model of a simple brute-force attack on a Norwegian Internet bank.

2 A simple attack

A possible brute-force attack against a Norwegian Internet bank in 2003 and 2004 is illustrated in Fig. 1. For simplicity, we consider only an attack utilizing SSNs. The single computer utilizes a large set of SSNs containing all the SSNs of the bank's online customers. (We'll explain how to determine this set shortly.) Note that the set will contain some SSNs not belonging to customers. The computer also needs the set of all possible PINs. If each PIN has n digits, then the set contains 10^n values.

When the attack starts, the computer picks an SSN from the set of SSNs and tries to log into the Internet bank using a randomly chosen PIN from the set of PIN values. Assuming that the SSN belongs to a customer, the probability of success is only 10^{-n} where $n \geq 4$ for the Internet banks we have studied. If the computer fails to log in, it tries again. Using the same SSN, the computer chooses a new PIN uniformly at random and repeats the login procedure. Since the bank closes access to the account after $T\,(> 1)$ trials with correct SSN and wrong PIN, the probability of success is $p = T/10^n$.

The computer repeats the above procedure, once for each SSN in the determined set. Since this set contains the SSNs of all the bank's customers, a cracker is able to access at least one account with probability

$$\begin{aligned} \text{P(access at least one account)} &= 1 - \text{P(access no accounts)} \\ &= 1 - (1-p)^Q, \end{aligned}$$

where Q is the number of customers in the Internet bank. The expected number of accounts a cracker gets into is $Q \cdot p$.

Note that the probability p was determined under the reasonable assumption that a bank generates customer PINs with uniform distribution. If the distribution is skewed, such that some PIN values are significantly more probable than others, then the described attack can be improved. As an example, the PINs may have a skewed distribution when the customers are allowed to choose their own PINs. According to Ross Anderson, about a third of the customers will use a birth date as a PIN in this case [2].

2.1 Norwegian SSNs

For a real attack to succeed, the cracker would've had to write computer code to generate a set of SSNs containing the SSNs associated with the Q online accounts. (As indicated earlier, the size of this set may be larger than Q, i.e., the set may contain SSNs not corresponding to any real accounts.)

The Norwegian population was about 4.5 million during 2003 and 2004. Each citizen is identified by a unique SSN consisting of 11 digits divided into three groups:

$$x_1x_2x_3x_4x_5x_6i_1i_2i_3c_1c_2.$$

Here, $x_1x_2x_3x_4x_5x_6$ is the date of birth (*ddmmyy*), $i_1i_2i_3$ is an individual identification number, and c_1c_2 are control digits:

$$\begin{aligned} c_1 &= 11- \\ &\quad ((3x_1 + 7x_2 + 6x_3 + x_4 + 8x_5 + 9x_6 + 4i_1 + 5i_2 + 2i_3) \bmod 11), \\ c_2 &= 11- \\ &\quad ((5x_1 + 4x_2 + 3x_3 + 2x_4 + 7x_5 + 6x_6 + 5i_1 + 4i_2 + 3i_3 + 2c_1) \bmod 11). \end{aligned}$$

All children born on the same day are assigned a unique $i_1i_2i_3$ number. The individual numbers are assigned chronologically and in such a way that i_3 is even for a girl and odd for a boy. If c_1 or c_2 is equal to 11, then the value is changed to 0. When c_1 or c_2 is equal to 10, the resulting 12-digit number isn't used because an SSN can only have 11 digits. To avoid confusion, we remark that a Norwegian SSN is called a "birth number" in Norway.

It should be clear that it is possible to generate efficiently a set of SSNs containing the SSNs of all Norwegian people between, say, the ages of 16 and 75. This set contains (nearly) all the SSNs of the customers in *any* Norwegian Internet bank. The cracker could've used this set during an attack. It was also possible for the cracker to reduce the size of the set by utilizing available statistical information on both the distribution of births in Norway and the usage of Norwegian Internet banks.

Norwegian account numbers also have a well-defined structure. A cracker could therefore run the same type of brute-force attack when a bank customer used an account number to log on instead of his or her SSN. More information on the structure of Norwegian account numbers is available in [3].

3 A distributed attack

Most likely, a bank's Intrusion-Detection System (IDS) will discover brute-force attacks similar to the attack described in the previous section, because no attempt is made by the cracker to hide the attack by e.g. spreading it over many days. Since the attack is run from a single computer, the bank can simply deny the computer access to its network to stop the attack. A cracker is of course well aware of this fact, and uses a so-called *botnet* to run the attack.

A botnet is a large network of PCs, called zombie PCs, that's controlled by a server. Worms such as MyDoom and Bagle are used to take over vulnerable PCs and add them to the expanding botnet. Often, the compromised PCs are controlled across Internet Relay Chat (IRC) channels [1].

Botnets can be large. A network of more than 10,000 zombie PCs was dismantled after security staff at the Norwegian telco Telenor located and shut down its controlling server [4]. It's also known that there are many botnets on the Internet. On September 20, 2004, The New York Times reported that the number of botnets monitored by Symantec increased from less than 2,000 to more than 30,000 during the six first months of that year. Symantec also saw a dramatic increase in electronic commerce attacks during the same period.

Let us once more consider the situation in Norway in 2003 and 2004. A cracker controlling a large botnet could divide a set of SSNs over all the zombie PCs in the network. Each PC could then try to log into an Internet bank using only the SSNs it was assigned. When the bank's IDS discovers the attack, the bank is not able to stop all the traffic from the zombie PCs because there are so many of them. Furthermore, since the attack is on the application layer of the network, it's difficult for the bank to discriminate between legitimate customers and zombie PCs trying to log on.

Note that the described attack is a combined brute-force/DDoS attack since the cracker could expect to get access to a small number of accounts $Q \cdot p$ while all the remaining $Q \cdot (1 - p)$ accounts would be closed to the customers.

3.1 Example attack

During 2003, new customers in a particular Norwegian Internet bank downloaded a certificate after providing an SSN and a fixed PIN with $n = 4$ digits. Unfortunately, the certificate could also be downloaded by a cracker knowing the SSN and PIN. Hence, the introduction of the certificate didn't really enhance the security.

The access to an account was closed after $T = 3$ unsuccessful login attempts. The probability that a cracker was able to log into at least one account was $\mathrm{P}(\text{access at least one account}) \approx 1$ for $Q = 220,000$ customers. The expected number of cracked accounts was 66.

A cracker could possibly run the attack several times. Since the attack is stochastic in nature, he could expect to gain access to 66 new accounts each

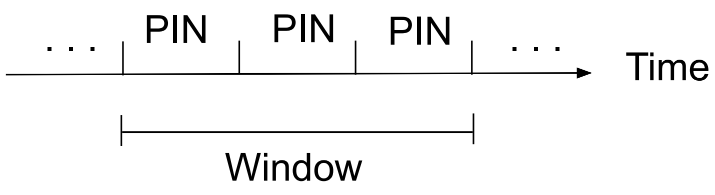

Figure 2: Window containing 3 PINs.

time. Fortunately, this Internet bank changed its login procedure in 2004.

4 Exploiting PIN calculators

A PIN calculator is supposed to provide two-factor authentication by requiring that the customer has something (the PIN calculator) and that he or she knows something (the secret PIN needed to activate the calculator). Often, the customer enters a *fixed* 4-digit PIN into the calculator to get a *dynamic* 6-digit PIN. This dynamic PIN is also called a one-time password.

The manufacturer initiates a new PIN calculator with data needed to generate a sequence of PINs. The same data is also stored in a database controlled by the Internet bank. The customer uses the calculator to generate a new PIN each time he wants to log into his account. The PIN is entered into the online bank's Graphical User Interface (GUI) and sent to the bank's security server. The server accesses the database to get the data for the given PIN calculator. It then calculates the current PIN and determines whether or not it's equal to the PIN generated by the calculator.

Many PIN calculators are programmed to generate a new (dynamic) PIN after a certain time period. It's therefore possible to divide a time line into equally long intervals and associate one PIN with each interval (see Fig. 2). Observe that both the calculator and the security server must have a clock to synchronize generation of the PINs in this case.

A problem occurs because of *clock drift*, i.e., the calculator and the server have different estimates of the time. To correct for the inevitable clock drift, a security server not only calculates the current PIN, but also the PIN in the previous time slot, as well as the PIN in the next time slot. The security server then compares the received PIN from the calculator with all three locally calculated PINs. In other words, the security server has a window of PINs that it will accept. Fig. 2 depicts a window of three PINs.

Several manufacturers allow larger windows of PINs. Provided a customer logs on regularly, an authentication server from RSA security will keep an estimate of the PIN calculator's time so that the dynamic PIN always falls within a window of three PINs. However, if a customer doesn't log on for an extended period, the PIN calculator's time could drift outside the window. In this case, the server tests against PINs in the 20 previous and the 20 next time slots, giving a window of 41 PINs [5].

Experimentation with two Vasco PIN calculators in 2004 indicated that the security server at a Norwegian Internet bank had a window of 19 PINs. The large size was maintained even when the customer logged on regularly. The existence of a window was confirmed in [6].

Let L denote the size of a security server's window of acceptable PINs. The window makes it easier to access an account. The attacker now has to guess an arbitrary PIN in a set of L PINs instead of the fixed PIN a customer enters into the bank's GUI when a PIN calculator isn't used. The fact that the calculator generates dynamic PINs only means that the cracker must empty an account the first time he gets access because he needs a new (unknown) PIN to log on a second time.

The probability that the cracker is able to access a particular account is $q = L \cdot p$ for an SSN belonging to a bank customer. The probability that the cracker is able to access at least one account is

$$\mathrm{P}(\text{access at least one account}) \quad = 1 - (1-q)^Q,$$

where Q again is the number of customers. The expected number of cracked accounts is $Q \cdot q$.

4.1 Example attack

Another Norwegian Internet bank had about $Q = 400,000$ customers during 2004, all using PIN calculators to generate dynamic PINs with $n = 6$ digits (in addition the calculators produced 2 control digits used to authenticate the bank's web site). The customers logged into their accounts using SSNs and dynamic PINs. Experimentation indicated that the bank closed the access to an account after $T = 5$ failed attempts to log on.

Assuming a minimum window of $L = 3$ PINs, the probability of getting access to at least one account is ≈ 0.998, and the expected number of cracked accounts is 6. Increasing the window size to its maximum $L = 19$, determined by experimenting with two real PIN calculators received from the bank, gives an average of 38 cracked accounts. It was possible to repeat the attack to gain access to new accounts each time.

The bank received a written report from us outlining our concerns. We also met with representatives from the bank's top-level management. They asked us to withhold the name of the bank, but had no objection to us publishing our findings. At the time of writing, the bank has not changed the login procedure for its Internet banking system, however, we understand that the bank has developed an alternative login procedure.

5 Attack generalizations

In the following we discuss a few possible attack variations during 2003 and 2004, introduce an SSN-filtering technique, and explain why our results apply to many current client-server systems.

5.1 Simple attack variations

During 2003 and 2004 it was also possible for a cracker to launch a pure DDoS attack on the application layer against many Norwegian Internet banks. Since a bank closed down the access to an account after T login trials with correct SSN and wrong PIN, all the customers' accounts could be closed down after $T \cdot S$ trials where S is the size of the set of SSNs. Note that while it was necessary to transmit a huge number of packets for a long time to launch a successful DDoS attack at a lower layer in the network, a DDoS attack utilizing an Internet bank's login procedure at the application layer could be carried out much more quickly. This made it hard to stop a real attack.

Assume that one bank was actually attacked. Once the bank had reopened the access to all accounts after the attack, it was possible for the cracker to start the DDoS attack again. Hence, the cracker could prevent (nearly) all bank customers from accessing their online accounts for a long time.

It was also possible for a cracker controlling a large botnet to launch DDoS attacks against several of the major Internet banks in Norway at the same time. Close to two million customers would be denied access to their online accounts in this case.

Several stealthy versions of both the simple attack launched from a single PC and the distributed attack were possible during 2003 and 2004. If a cracker had knowledge about a bank's IDS, he could try to design an attack to avoid detection. It was of course important to make the set of SSNs as small as possible. As an example, the cracker could reduce the set of SSNs by concentrating the attack on a particular age group, say, people between 35 and 50. This group was likely to have more money in the bank than younger people. The cracker could also spread the attack over many days to avoid too many unsuccessful logins on a single day.

One of the anonymous reviewers pointed out that a cracker could write an alternative attack program which held the PIN fixed as it ran through all the SSNs. This alternative attack was advantageous to the cracker when some PIN values were more likely than others.

The cracker could first run this attack using one of the most likely PINs. He could then repeat the attack several times using other PINs that occurred with high probabilities. Depending on the actual distribution of the PINs, the cracker could expect to get access to a significantly larger number of accounts than in the case of uniform PIN distribution.

5.2 SSN filtering

We have discovered that it's sometimes possible to determine SSNs belonging to customers in an online bank by exploiting the bank's own web site. A script can be written that tries to log into the site using different SSNs. For each SSN, the script receives a message from the site. Often, there are several types of messages depending on whether an SSN is valid and whether it belongs to a customer. The script can use these messages to determine customer SSNs. The described "filtering" of SSNs can be very helpful when constructing a small set of SSNs.

Other web sites—not belonging to Norwegian banks—may also be exploited to determine SSNs in use. In the following we report our experience with a site created by a Norwegian Public Service Pension Fund (NPSPF). A large percentage of the Norwegian population are members of this pension fund. In particular, government employees, including university employees, in Norway are members.

Any member of NPSPF could apply for a housing loan on the web during 2004. To apply for a loan, the applicant first entered his Norwegian SSN. A small script could also carry out the same operation. If the SSN was valid and it belonged to a member of the pension fund, then the script received a message containing the person's name and address, including the zip code; else the script received an error message.

Since the Norwegian SSNs have a well-defined structure, a script can generate a large number of SSNs that may or may not be in use. Using the NPSPF's web site during 2004, such a script could build a database containing the name, address, and SSN of a large number of Norwegian citizens. Because the database contained a zip code for each person, a cracker could easily create a set of SSNs belonging to people in a particular geographical area. This again enabled an attacker to go after the customers in smaller local banks.

One of our colleagues informed both NPSPF and the *Data Inspectorate*, an independent administrative body under the Norwegian Ministry of Labour and Government Administration, that it was possible to determine valid SSNs using NPSPF's web page for loan applications. In a letter to the Data Inspectorate, NPSPF promised to change the web page for loan applications within 14 days. However, when we checked after 2 weeks it was still possible to filter out a large number of SSNs. Our test script was able to access NPSPF's web site for nearly 24 hours. NPSPF and the Data Inspectorate were contacted once again, this time we included the SSNs and addresses belonging to the Norwegian Prime Minister and the Director of the Data Inspectorate. We also explained that an attacker could create chaos by applying for a large number of loans using other people's SSNs obtained from NPSPF's own site.

During 2003 and 2004 it was also possible to filter out SSNs from other Norwegian web sites. In fact, by combining SSNs from several sites, an attacker could build a database with the names, addresses, and SSNs of nearly everybody in Norway.

5.3 Generalization to other systems

It's important to realize that our insights apply to any client-server system on the Internet as long as the authentication of each client is based on a structured user ID and a short PIN (or password). Examples of IDs with a well-defined structure are SSNs, account numbers, patient IDs, and e-mail addresses. Whether a customer's PIN is static or dynamic makes little difference in our case.

A combined DDoS/brute-force attack represents a potential serious problem for a client-server system owned by a business enterprise. The attack is hard to stop, it closes down many user accounts, and it allows a cracker to access some of the accounts. An attack can also result in other problems for the enterprise. The venture can lose income if it closes down its servers to stop the cracker from accessing accounts. Bad press can result in loss of current and future customers, as well as reduce the level of trust afforded by the remaining customers. The value of the brand can be reduced, especially if it takes the enterprise a long time to reactivate all the accounts closed down by the attack. Consequently, designers of commercial client-server systems using structured IDs and fixed or dynamic PINs should verify that their systems are not vulnerable to an attack.

6 Countermeasures

This section discusses how a system can fend off combined DDoS/brute-force attacks at the application layer of the network.

6.1 Brute-force countermeasures

The degree of exposure to brute-force attacks depends on the number of digits n in the PINs. Consider one client-server system using fixed PINs and another system using dynamic PINs, and assume that all PINs in both systems have n digits. Let us compare the two systems' exposure to brute-force attacks. If the dynamic PINs are generated by PIN calculators and the security server has a window of length $L = 10$, then it is 10 times more likely that an attack program guesses the correct PIN for a given customer compared to the case when the PINs are fixed.

The system using dynamic PINs of length n has the same level of security as a system using fixed PINs of length $n - 1$. In other words, the use of PIN calculators can cause a system to become significantly more vulnerable to brute-force attacks. Hence, it may be necessary to use longer PINs in systems utilizing PIN calculators than in systems using fixed PINs.

As the number of users grows, a client-server system using PINs of a set length n becomes more vulnerable to brute-force attacks. A designer of a new system must therefore estimate the number of future users before she can determine the number of digits n needed to be safe from brute-force attacks. The simple calculations introduced earlier in the paper show how to determine whether or not the PINs have enough digits.

Brute-force attacks can be further hampered if large random numbers are used as user IDs, making it difficult for an attack program to generate these IDs.

6.2 DDoS countermeasures

There exist various DDoS attacks targeted at different network layers. No general defense against these attacks exists. We refer the interested reader to [1] for a comprehensive introduction to DDoS attacks and the different techniques used to limit the negative consequences of the attacks.

We've seen that if a client-server system closes down a customer's access to the server after a few login trials with correct user ID and wrong PIN, then a simple and efficient application-layer DDoS attack can prevent all customers from getting access to the server. Clearly, well-designed client-server systems should not utilize ID/PIN authentication techniques that reduce the amount of resources a cracker needs to carry out a DDoS attack at the application layer.

One possible solution to this particular DDoS attack is to base a client-server system on a Public-Key Infrastructure (PKI) that requires new users to show up in person at the registration authority before getting access to the system [7]. In this case it is no longer necessary to transmit IDs and PINs from the clients to the server. Instead, the server can verify that a client has the (long random) private key corresponding to the public key in the client's certificate.

A PKI-based solution may still need a PIN to unlock a client's private key. However, this PIN need not be transmitted to the server. On the other hand, there exist PKI solutions where user IDs and PINs are sent from the clients to the server. These solutions may be vulnerable to DDoS attacks.

7 Discussion

To help foster development and maintenance of distributed systems with better security, we first discuss why the combined DDoS/brute-force attack was possible during much of 2003 and 2004. Next, we consider the dangers associated with the Norwegian banks' security-by-obscurity policy, before finally discussing how universities can teach computer science students to develop more secure systems.

7.1 Bad security

A high level of expertise isn't needed to carry out the presented attacks; they're only based on well-known brute forcing and DDoS techniques. A well-designed system should of course not be vulnerable to brute forcing in practice. In fact, this is one of the first things a designer of a new system should verify. Why then were many Internet banking systems in Norway with a total of more than one million customers vulnerable to the combined DDoS/brute-force attack during 2003 and 2004? Our answer is based on discussions with representatives from Norwegian banks and a report [8] from *Kredittilsynet*, the Norwegian government agency overseeing the banks.

Many of the banks' security experts and software developers had prior experience with Automatic Teller Machine (ATM) systems. The (mental) models developed during the ATM work have—to a large degree—influenced the design of the Internet banking systems. Since it's difficult to access customer accounts in an ATM system by brute-force attack, it seems that the banks paid insufficient attention to the comparative ease with which a cracker can assail an Internet banking system by brute force.

Because no Norwegian bank was instructed by Kredittilsynet to develop a separate risk analysis for its IT systems before August 2003, few banks had in place mechanisms to determine an acceptable level of risk for these systems [8]. Consequently, the security of a system could very well deteriorate over time without a bank discovering the fact. Of course a successful brute-force attack requires that an Internet bank has a large number of customers. Initially, the number of customers was relatively small in Norwegian Internet banks. As the number of customers grew, not all banks realized that it was necessary to increase the number of digits in the PINs to avoid being vulnerable to brute-force attacks.

Many banks had outsourced the daily operation of their Internet banking systems during 2003 and 2004. Kredittilsynet [8] has pointed out that it was difficult for these banks to maintain the needed security expertise when no longer they could learn from their own systems. A single corporation operated most of the Internet banking systems during the discussed time period. Since this company had nearly a monopoly, and many banks no longer had the ability to evaluate the security of their outsourced systems, there were no external mechanisms outside the corporation to ensure that the Internet banking systems maintained a high level of security over time.

Taken together, these reasons led to much of the bad security in Norwegian Internet banking systems. In the following, we argue that the banks' security-by-obscurity policy was also a contributing factor.

7.2 Security by obscurity

Quoting Bruce Schneier [9], "there is a considerable confusion between the concept of secrecy and the concept of security, and it is causing a lot of bad security." Like many foreign banks [10], Norwegian banks have long practiced security by obscurity. No technical information about the banking systems' security protocols and use of cryptographic primitives is made available to independent security researchers or customers complaining about debits on their accounts for which they were not responsible. The banks have for many years simply stated that their systems are very secure.

Our analysis shows that the security in Norwegian Internet banking systems may not be very high after all. We believe that the banks' security-by-obscurity policy has led to a false feeling of security instead of real security, making the Internet banking systems vulnerable to rather trivial attacks during 2003 and 2004. It's well known that it's much more difficult to design a new secure system than to find vulnerabilities in an existing system. After a designer has invested much time, effort, and prestige on a new design, he or she may not be motivated to find weaknesses in the same design. It's therefore important to hire outside experts to evaluate all new security designs. It's equally important to evaluate the implementation on a regular basis. As an example, we have seen how a Norwegian Internet bank has several times been vulnerable to cross site scripting and, hence, "phishing" e-mail scams after it made changes to its web site.

In Norway, the banks' upper-level management is much to blame for the current practice of security by obscurity. Management typically has little understanding of real security, and has a tendency to assume that a system is secure if all information about it is kept secret. Consequently, all employees responsible for the security must sign nondisclosure agreements making them ill-prepared to discuss security problems with anybody outside the banking industry. This was amply demonstrated to us when we tried to inform selected banks about our findings. In our opinion the security-by-obscurity policy creates unnecessary friction, prevents us from learning, and, hence, causes the same mistakes to be made over and over again.

7.3 Teaching security

It's desirable to develop better security courses for tomorrow's computer science students. Initially, such a course should evaluate the security in some existing systems, preferably systems that the students already use. An 'holistic' approach should be taken, covering the main security techniques implemented in the evaluated systems; difficult technical details should be avoided because they will only cloud the important issues at this early stage.

A course should cover banking systems, the next biggest application of cryptology after government systems. We believe that future Internet banking systems must be based on PKIs with client certificates [7] to strengthen the customer authentication. Consequently, it's important to analyze at least one real

PKI system during a course.

The new Norwegian BankID standard for Internet banking has a PKI. Unfortunately, the Norwegian banks have (once again) decided to keep the complete standard a secret, and not allow an independent evaluation of the standard. Clearly, it's important to teach the students the difference between what information needs to be kept secret and what information may be shared.

We're of the opinion that it's necessary to study real attacks in a security course. A good understanding of attacks helps students analyze the security through the eyes of a cracker; exposing weaknesses and determining the most serious risks. Furthermore, demonstrations of "real" attacks make wonders when it comes to motivating students to incorporate security in the initial system design.

In particular, DDoS attacks should be studied. As we have seen, it isn't a good idea to design a login procedure that simplifies a DDoS attack by closing down access to an account after a few login attempts. The discussed security course should contain an introduction to different types of DDoS attacks and techniques to mitigate such attacks.

The decision to teach attack techniques should not be taken without some serious thought to the potential consequences. It's irresponsible to study attacks without also including some discussion about what constitutes ethical behavior. It may be a good idea to style this part of the course as an introduction to penetration testing to emphasize that the attacks are taught to discover vulnerabilities in systems, not to attack systems for personal financial reasons.

An understanding of vulnerabilities and attacks alone isn't enough to develop secure systems that let us escape from the endless cycle of penetration and patch. We recommend [11] for a discussion on how to teach constructive security. Information on how to build secure software may be found in [12].

8 Final remarks

Internet banking is increasingly popular both in Norway and in many other countries. The banks have actively encouraged this trend by persuading customers to sign up as a cost-saving measure. The online banking customers appear to be driven by convenience and aren't much concerned about identity theft and "phishing" e-mail scams. In fact, most customers seem to believe that Internet banking is very safe, simply because their banks told them so. In reality, the customers may well have a false sense of security. We believe more online banking systems must be evaluated by independent security researchers to determine the true level of security. Our own investigation shows that the authentication of many Norwegian online customers was too weak in 2003 and 2004.

Like many other security researchers, we have experienced how hard it is to alert banks (and other large institutions) to security weaknesses. The press on the other hand is very interested in the results of our research. In one instance we informed the press about our findings after the bank in question had made changes to its system. We got a lot of media attention, but also some very negative comments from mostly anonymous sources. In a few cases it was even suggested to us that we had personal financial motives for our investigation of the banks' security procedures. Personal attacks like these only show how important it is to partake in the public debate about security issues in an open and straight forward manner—and how vital it is to create more and better security courses in our universities.

References

[1] J. Mirkovic, S. Dietrich, D. Dittrich, and P. Reiher, *Internet Denial of Service Attack and Defense Mechanisms.* Prentice Hall, 2005.

[2] R. Anderson, *Security Engineering.* Wiley, 2001.

[3] IBAN, "Domestic account number." [Online]. Available: http://www.ecbs.org/Download/TR201/norway.pdf

[4] The Register, "Telenor takes down massive botnet." [Online]. Available: http://www.theregister.co.uk/2004/09/09/telenor_botnet_dismantled/

[5] RSA Security, "The power behind RSA SecurID two-factor user authentication: RSA ACE/server," 2001, white paper.

[6] Vasco, "VACMAN controller integration," technical white paper.

[7] C. Adams and S. Lloyd, *Understanding PKI*, 2nd ed. Addison-Wesley, 2003.

[8] The Financial Supervisory Authority of Norway, "Risk and vulnerability analysis 2003," 2003, (in Norwegian).

[9] B. Schneier, "Keeping network outages secret." [Online]. Available: http://www.schneier.com/crypto-gram-0410.html#2

[10] R. Anderson, "Why cryptosystems fail, from communications of the ACM, november, 1994," in *William Stallings, Practical Cryptography for Data Internetworks.* IEEE Computer Society Press, 1996. [Online]. Available: http://citeseer.ist.psu.edu/anderson94why.html

[11] C. E. Irvine, "Teaching constructive security." *IEEE Security & Privacy*, vol. 1, no. 6, pp. 59–61, 2003.

[12] J. Viega and G. McGraw, *Building Secure Software.* Addison-Wesley, 2002.