

Case Study: Online Banking Security

A description of attack scenarios over a two-year period illustrates several key security issues with Internet banking systems in Norway. Given the banks' security-by-obscurity policy, online customers knew little about security levels and falsely believed their assets were safe.

KJELL J. HOLE,
VEBJØRN
MOEN, AND
THOMAS
TJØSTHEIM
*University of
Bergen,
Norway*

Internet banking is increasingly popular both in Norway and elsewhere. Banks have actively encouraged this cost-saving trend by persuading customers to sign up. Customers, attracted by online banking's convenience, seem largely unconcerned about identity theft and phishing email scams. In fact, most customers seem to believe that Internet banking is quite safe simply because their banks told them so. In reality, this sense of security might be false.

We studied customer authentication methods in several Norwegian Internet banks from 2003 through 2004. Our investigation shows that authentication was often weak, offering simple—but powerful—attack possibilities. (Fortunately, none of the attacks were actually carried out.) Here, we discuss the authentication methods and the attacks they made possible. Our scenarios are based solely on publicly available Internet information.

Upon concluding our study, we presented our findings to the Norwegian government agency overseeing the national banking industry. We also engaged in a sustained effort to directly inform the banks most vulnerable to attacks. Our main reason for making this account public is to contribute to the development of more secure Internet banking systems. To further that aim, we speculate on why banks have developed insecure Internet banking solutions in the first place. We also suggest how universities might teach students to design more secure alternatives.

Overview: Possible attacks

As the “Norwegian identification numbers” sidebar describes, many Norwegian Internet banks have long required their customers to log in to online accounts using a social security number (SSN) or account number, as

well as a personal identification number (PIN). While SSNs and account numbers aren't considered secrets, only the customers know their own PINs. If a customer attempts to log in to an account using the correct SSN (or account number) and the wrong PIN more than a certain number of times (usually three to five), the bank will temporarily deny that customer further access to that account.

As we describe here, during our study period it was possible for crackers to launch attacks against Internet banks by combining simple brute-force attacks with distributed denial-of-service (DDoS) attacks¹ that exploit a bank's login procedure. By successfully combining these attacks, attackers not only gain access to a few accounts, but also prevent numerous legitimate customers from accessing their accounts. Such a combined attack is also effective when the bank's customers use PIN calculators—also called PIN cards or (hardware) tokens—to generate dynamic PINs.

Brute-force attacks

Figure 1 shows an example brute-force attack against a Norwegian Internet bank; for simplicity's sake, we consider an attack that uses only SSNs. A single computer utilizes a large set of SSNs, which contains all of the SSNs of the bank's online customers (as well as 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.

To start the attack, the computer picks an SSN from the SSN set and tries to log in to the Internet bank using a randomly chosen PIN from the PIN values set. Assuming the SSN belongs to a customer, the success probabil-



Norwegian identification numbers

In Norway, customers typically log in to accounts using either their social security numbers (SSNs) or account numbers. For an attack to succeed, the cracker would have to write computer code to generate an SSN set that contains the SSNs associated with the Q online accounts. (The set's size might be larger than Q, containing SSNs not corresponding to real accounts.)

During 2003 to 2004, Norway's population was roughly 4.6 million. Each Norwegian citizen has a unique SSN—or birth number—consisting of 11 digits divided into three groups:

$$x_1 x_2 x_3 x_4 x_5 x_6 i_1 i_2 i_3 c_1 c_2.$$

Here, $x_1 x_2 x_3 x_4 x_5 x_6$ is the date of birth (*ddmmyy*), $i_1 i_2 i_3$ is an individual identification number, and $c_1 c_2$ are control digits:

$$c_1 = 11 - ((3x_1 + 7x_2 + 6x_3 + x_4 + 8x_5 + 9x_6 + 4i_1 + 5i_2 + 2i_3) \bmod 11),$$

$$c_2 = 11 - ((5x_1 + 4x_2 + 3x_3 + 2x_4 + 7x_5 + 6x_6 + 5i_1 + 4i_2 + 3i_3 + 2c_1) \bmod 11).$$

All children born on the same day are chronologically assigned a unique $i_1 i_2 i_3$ number; i_3 is even for girls and odd for boys. 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 have only 11 digits. Clearly, it's possible to efficiently generate an SSN set 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, and the attacker could use this set during an attack. It's also possible for a cracker to reduce the set's size using available statistical information on both Norway's birth distributions and Internet bank use. Norwegian account numbers also have a well-defined structure.¹ Given this, a cracker could run the same type of brute-force attack when bank customers log in using account numbers rather than SSNs.

Reference

1. European Committee for Banking Standards, "Domestic Account Number," Mar. 2003; www.ecbs.org/Download/TR201/norway.pdf.

ity is only 10^{-n} , where $n \geq 4$ for the Internet banks we've studied. If the login fails, the computer tries again, using the same SSN and a new PIN chosen uniformly at random. Because the bank closes access to an account after T (> 1) trials with correct SSN and incorrect PIN, the probability of success is $p = T/10^n$. The computer repeats the same procedure for each SSN in the determined set. Because this set contains all bank customers' SSNs, a cracker can access at least one account with probability

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

where Q is the number of bank customers. Thus, the expected number of accounts a cracker gains access to is $Q \times p$. We determined the p probability under the reasonable assumption that a bank generates customer PINs with uniform distribution. If this distribution is skewed, such that some PIN values are significantly more probable than others, then the attack's success odds improve. The PIN distribution is skewed, for example, when customers can choose their own PINs. In this case, according to noted security expert Ross Anderson, about one-third of customers will use a birth date as a PIN.²

Distributed attacks

Most likely, a bank's intrusion detection system (IDS) will discover brute-force attacks like the one we just described because the cracker makes no attempt to hide the attack (by spreading it over many days, for example). Because the cracker runs the attack from a single computer, the bank can easily stop the attack by denying

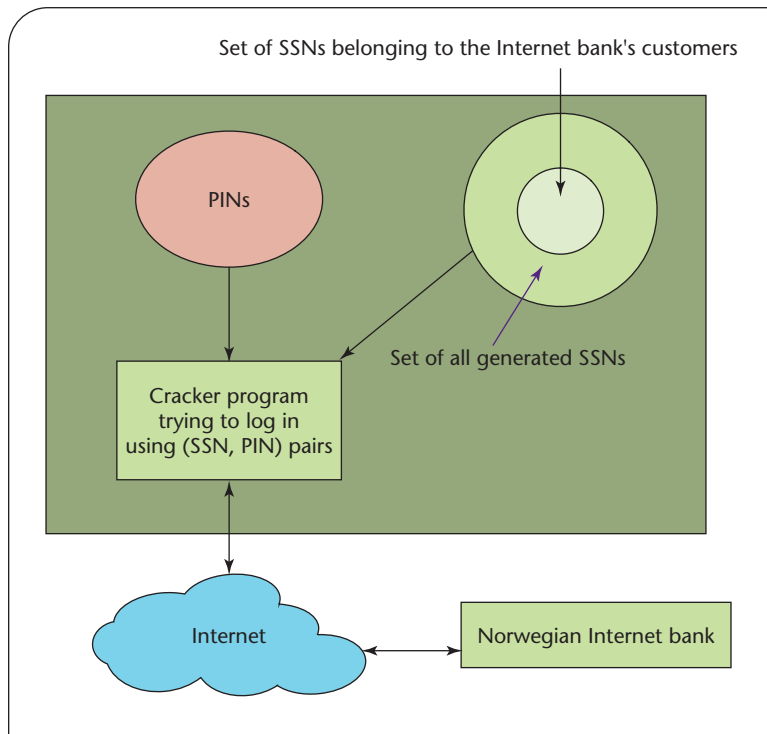


Figure 1. Model of a simple brute-force attack on a Norwegian Internet bank. A single computer picks a social security number (SSN) from a large set and then attempts to log in using a randomly chosen personal identification number (PIN).

network access to that computer. Crackers, of course, are well aware of this fact and therefore often use a *botnet* to run an attack.

Botnets. A botnet is a large network of “zombie” PCs controlled by a server. Crackers use worms—such as MyDoom and Bagle—to take over vulnerable PCs and add them to the expanding botnet. The compromised PCs are often controlled across Internet Relay Chat (IRC) channels.¹ Crackers can assemble enormous botnets. For example, security staff at Norway’s Telenor telecommunications company dismantled a network of more than 10,000 zombie PCs when they located and shut down its controlling server.³

Globally, the Internet has many botnets. On 20 September 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 first six months of that year.⁴ Symantec also saw a dramatic increase in e-commerce attacks during the same period.

Example attack. Let’s return to the situation in Norway during our study period. Say a cracker controls a large botnet and divides a set of SSNs across the network’s zombie PCs. Each PC could then try to log in to an Internet bank using only its assigned SSNs. Once the bank’s IDS discovered the attack, the bank couldn’t stop it because of the volume of zombie PC traffic. Furthermore, because the attack would target the network’s application layer, it would be difficult for the bank to distinguish between legitimate customers and zombie PCs trying to log on. Our attack thus combines brute-force and DDoS attacks: the cracker could expect to gain access to a few accounts $Q \times p$, while closing customer access to all remaining $Q(1 - p)$ accounts.

During 2003, a Norwegian Internet bank’s new customers downloaded a certificate after they provided their SSNs and fixed PINs with $n = 4$ digits. Unfortunately, crackers could also download the certificate if they knew the SSN and PIN; hence, introducing the certificate didn’t actually enhance security. The bank closed account access after $T = 3$ unsuccessful login attempts. The probability that a cracker could have logged in to at least one account was therefore $P(\text{access at least one account}) \approx 1$ for $Q = 220,000$ customers. Thus, the cracker could have cracked approximately 66 accounts. Because the attack is stochastic, if the cracker ran the attack several times, he or she could have expected to gain access to 66 new accounts each time. Fortunately, our example Internet bank changed its login procedure in 2004.

Exploiting PIN calculators

To enhance security, some banks use a PIN calculator, which aims to provide two-factor authentication by requiring customers to have something (the PIN calculator) and to know something (the secret PIN that activates the calculator). Often, customers enter a fixed four-digit PIN into the calculator to get a dynamic six-digit PIN, or a one-time password.

How they work. PIN calculators contain the data needed to generate a sequence of PINs. This data is also stored in a database that the Internet bank controls. Each time customers want to access their accounts, they use the calculator to generate a new PIN. They then enter the new PIN into the online bank’s GUI, which sends it to the bank’s security server. The server then accesses the database to get the data for the given PIN calculator, calculates the current PIN, and determines whether it’s equal to the calculator-generated PIN.

Many PIN calculators generate a new (dynamic) PIN after a certain time period. It’s therefore possible to divide a timeline into equally long intervals and associate one PIN with each interval. In this case, both the calculator and security server must have a clock to synchronize PIN generation. However, clock drift—a loss of synchronization—is inevitable. When it occurs, a security server not only calculates the current PIN, but also the PINs in the previous and subsequent time slots. The security server then compares the received PIN from the calculator with all three locally calculated PINs. That is, the security server has a window of PINs that it will accept.

Several PIN calculator manufacturers allow larger PIN windows. Provided a customer logs on regularly, an authentication server from RSA security will keep track of the PIN calculator’s time so that the dynamic PIN always falls within a three-PIN window. 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 20 previous and the 20 subsequent time slots, giving a window of 41 PINs.⁵ When the received PIN falls outside the three-PIN window, the RSA authentication server asks for a second PIN. However, our example bank didn’t ask customers for a second PIN when the clock drift was large.

PIN window experiment. Experimentation with two Vasco PIN calculators in 2004 indicated that a Norwegian Internet bank’s security server had a window of 19 PINs. The server maintained this large window even when a customer logged on regularly.

Let L denote the size of a security server’s window of acceptable PINs. A large window makes account access easier: attackers must guess an arbitrary PIN in a set of L PINs, rather than the single fixed PIN that customers use when they enter the bank’s GUI without a PIN calculator. The fact that the calculator generates dynamic PINs means only that crackers are likely to empty an account the first time they access it because a second access requires a new (unknown) PIN.

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

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

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

Example attack. One Norwegian Internet bank had roughly 400,000 customers during 2004, all of whom used PIN calculators to generate dynamic PINs with six digits (the calculators also produced two control digits to authenticate the bank's Web site). The customers logged into their accounts using SSNs and dynamic PINs. Our experiments indicated that the bank closed account access after $T = 5$ failed login attempts.

Assuming a minimum window of $L = 3$ PINs, the probability of an attacker accessing at least one account would be approximately 0.998, and the expected number of cracked accounts would be six. Increasing the window size to its maximum— $L = 19$, which we determined by experimenting with two of the bank's actual PIN calculators—produced an average of 38 cracked accounts. It was possible to repeatedly attack the bank and gain access to new accounts each time.

We sent the bank a written report, outlining our concerns. We also met with representatives from the bank's top-level management. Although they asked us to withhold the institution's name, bank officials had no objection to our publishing our findings. At the time this article went to press, the bank still hadn't changed the login procedure for its Internet banking system.

SSN filtering

Using SSN filtering, attackers can exploit a company's own Web site to determine customer SSNs. They do this by writing a script that tries to log in to the site using different SSNs. For each SSN, the script receives a message from the site. The messages vary depending on whether an SSN is valid and belongs to a customer or not. The script uses these messages to determine customer SSNs. This SSN filtering can be highly useful for constructing a small set of SSNs.

We experimented with a site created by a Norwegian Public Service Pension Fund (NPSPF), which many Norwegians belong to, including government and university employees. In 2004, any NPSPF member could apply online for a housing loan. To apply for a loan, applicants first entered their SSNs. A small script could also perform the same operation. If the SSN was valid and belonged to an NPSPF member, the script received a message containing the person's name and address (including the zip code); otherwise, the script received an error message. Because Norwegian SSNs have a well-defined structure, a script can generate numerous SSNs that might be in use. Given this, the script could build a database containing the name, address, and SSN of many Norwegian citizens. Furthermore, because the database contained a

zip code for each person, a cracker could easily create a set of SSNs for people in a particular geographical area. Attackers could thus go after customers in smaller local banks.

We 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 loan application Web page. In a subsequent letter to the Data Inspectorate, NPSPF promised to change the Web page for loan applications within 14 days. However, when we checked the site after two weeks, it was still possible to filter out numerous SSNs. Our test script was able to access NPSPF's Web site for nearly 24 hours. We contacted NPSPF and the Data Inspectorate once again, this time including the SSNs and addresses of the Norwegian Prime Minister and the Director of the Data Inspectorate. We also explained that an attacker could create chaos by applying for numerous loans using other people's SSNs obtained from NPSPF's own site. Unfortunately, we received no reply.

Attack variations and generalization

There were a few possible variations on the 2003–2004 attacks, including pure application-layer DDoS attacks and stealthy brute-force attacks. As with the basic attacks described earlier, these attacks are applicable to many current client-server systems.

Simple attack variations

During 2003 and 2004, crackers could have launched pure, application-layer DDoS attacks against many Norwegian Internet banks. Because banks closed access to an account after T login trials with the correct SSN and the wrong PIN, all customer accounts could be closed down after $T \times S$ trials, where S is the size of the SSNs set.

So, let's assume that a bank was actually attacked. Once the bank reopened access to all accounts after the attack, the cracker could have started the DDoS attack again. Hence, the cracker could have prevented nearly all bank customers from accessing their online accounts for many hours, or perhaps even several days. Also, while launching a successful DDoS attack at a lower network layer would have required attackers to transmit numerous packets over an extended period, they could have more quickly carried out a DDoS attack using an Internet bank's application-layer login procedure. This would make it hard to stop a real attack. In addition, crackers controlling a large botnet could have simultaneously launched DDoS attacks against several of Norway's major Internet banks during our study period. If such attacks had occurred, close to 2 million customers would have been denied access to their online accounts.

Several stealthy versions of both types of brute-force

attack—a simple attack launched from a single PC and a distributed attack—were possible during our study. If crackers had knowledge about a bank's IDS, they could have tried to design an attack to avoid detection. In any case, it was obviously important to make the SSN set as small as possible. Crackers can do this, for example, by concentrating the attack on a particular age group—say, people between 35 and 50, who are likely to have more money in the bank than younger people.

Crackers could also write an alternative attack program to hold a fixed PIN while it runs through all the SSNs. Such an alternative attack can be particularly advantageous when some PIN values are more likely than others. In our examples, the crackers could have first run this attack using one of the most likely PINs. They could then repeat the attack several times using other high-probability PINs. Depending on the actual PIN distributions, the crackers could expect to access significantly more accounts than are possible with a uniform PIN distribution.

Generalization to other systems

Our insights apply to any client-server system on the Internet that authenticates each client using 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 email addresses. Whether a customer's PIN is static or dynamic makes little difference in our case.

A combined DDoS/brute-force attack represents a potentially serious problem for a business enterprise's client-server system for several reasons: it's hard to stop, it closes down many user accounts, and it allows crackers access to some accounts. An attack can also create other problems for the target organization:

- The organization can lose income if it has to close down its servers to stop crackers from accessing accounts.
- Bad press can result in a loss of current and future customers, as well as reduce trust among remaining customers.
- The brand's value might decline, especially if the enterprise takes a long time to reactivate the accounts closed by the attack.

Consequently, designers of commercial client-server systems that use structured IDs and fixed or dynamic PINs should verify that their systems are invulnerable to such attacks.

Attack countermeasures

Organizations have several options for fending off combined DDoS/brute-force attacks at the network's application layer.

Brute-force countermeasures

The degree of exposure to brute-force attacks depends

on the PIN's number of digits n . Let's consider one client-server system using fixed PINs and another using dynamic PINs, and assume that all PINs in both systems have n digits. If the dynamic PINs are generated by PIN calculators and the security server has a window of length $L = 10$, an attack program is 10 times more likely to guess the correct PIN for a given customer than when the PINs are fixed. The system using dynamic PINs of length n has the same security level as a system using fixed PINs of length $n - 1$. In other words, when it uses a PIN calculator, a system is significantly more vulnerable to brute-force attacks. Hence, systems using PIN calculators might need longer PINs than 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. Designers of new systems must therefore estimate the number of future users before they can determine the number of digits n needed to be safe from brute-force attacks. The simple calculations we introduced earlier can show designers how to determine PIN digit length. Organizations can further hamper brute-force attacks by making user IDs large random numbers, which are difficult for an attack program to generate.

DDoS countermeasures

Various DDoS attacks are targeted at different network layers. There is no general defense against such attacks. Jelena Mirkovic and colleagues offer a comprehensive introduction to DDoS attacks and different techniques that can limit their negative consequences.¹ When client-server systems close down a customer's server access after a few login trials with the wrong PIN, a simple and efficient application-layer DDoS attack can prevent all customers from accessing the server. Clearly, well-designed client-server systems shouldn't utilize ID/PIN authentication techniques, as they reduce the resources crackers need to carry out an application-layer DDoS attack.

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 register in person before they can access the system.⁶ This alleviates the need 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.

Some PKI-based solutions still need a PIN to unlock a client's private key. However, that PIN need not be transmitted to the server. On the other hand, some PKI solutions send user IDs and PINs from the clients to the server and might thus be vulnerable to DDoS attacks.

Discussion

To help foster development and maintenance of more secure distributed systems, it's important to examine

why the combined DDoS/brute-force attack was possible during our study period.

Bad security

The attacks we discuss here use well-known brute-force and DDoS techniques and require no high-level expertise to perform. In practice, of course, a well-designed system should be invulnerable to brute forcing. In fact, this is one of the first things a new system's designer should verify. Why, then, were many of Norway's Internet banking systems—which had a total of more than 1 million customers—vulnerable to the combined DDoS/brute-force attack during 2003 and 2004? We base our answer to this question on discussions with Norwegian bank representatives and a report⁷ from Kredittilsynet, the Norwegian government agency that oversees banks.

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

Risk assessment. Before August 2003, Kredittilsynet didn't require Norwegian banks to develop a separate risk analysis for their IT systems. As a result, few banks had mechanisms to determine their IT systems' acceptable risk levels.⁷ A system's security could therefore deteriorate over time without the bank's knowledge. Of course, a successful brute-force attack requires that an Internet bank have many customers, and in Norway, the initial number of Internet banking customers was relatively small. As the number of customers grew, however, some banks failed to realize that they had to increase the number of PIN digits to avoid brute-force-attack vulnerability.

Outsourcing. In addition, during 2003 and 2004, many banks had outsourced their Internet banking systems' daily operations. Kredittilsynet pointed out that it was difficult for these banks to maintain the needed security expertise when they couldn't learn from their own systems.⁷ During the study period, a single corporation operated most of the Internet banking systems. This company had a near monopoly, and many banks no longer had the ability to evaluate their outsourced systems' security. As a result, no mechanisms external to the corporation operating the Internet banking systems could ensure the systems' high-level security over time.

Security by obscurity

These factors culminated in much of the bad security in

Norwegian Internet banking systems during our study. Another contributing factor was the banks' security-by-obscurity policy. As security expert Bruce Schneier put it, “there is a considerable confusion between the concept of secrecy and the concept of security, and it is causing a lot of bad security.”⁸ Like many foreign banks,⁹ Norwegian banks have long practiced security by obscurity. Technical information about the banking systems' security protocols and use of cryptographic primitives is unavailable to independent security researchers and customers, who might have account debits for which they weren't responsible. For many years, banks have simply stated that their systems are very secure.

Our analysis shows, however, that Norwegian Internet banking systems' security might not be very high after all. We believe that the banks' security-by-obscurity policy led to a false feeling of security instead of real security, making the systems vulnerable to rather trivial attacks during 2003 and 2004.

Obviously, it's much more difficult to design a new, secure system than to find vulnerabilities in an existing system. Once designers have invested much time, effort, and prestige on a new design, they might not be motivated to find its weaknesses. It's therefore important to hire outside experts to evaluate all new security designs. It's equally important to regularly evaluate a system's implementation. As an example, in our study we saw a Norwegian Internet bank experience repeated vulnerability to cross-site scripting—and, hence, “phishing” email scams—after it made changes to its Web site.

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

Teaching security

We must develop better security courses for tomorrow's computer science students. Initially, such a course should evaluate existing systems' security—preferably systems that the students already use. Professors should take a holistic approach, covering the target systems' main security techniques. They should avoid difficult technical details, which will only cloud important issues at this early stage. Security courses should also cover banking systems,

which are the next biggest application of cryptology after government systems.

In our view, future Internet banking systems must be based on PKIs with client certificates⁶ to strengthen customer authentication. Consequently, a course should analyze at least one real PKI system. The new Norwegian BankID Internet banking standard has a PKI. Unfortunately, Norwegian banks have (once again) decided to keep the complete standard a secret, preventing its independent evaluation. Clearly, it's also important to teach the students which information needs to be kept secret, and which must be shared to improve security.

In addition, a security course should focus on real attacks. A good understanding of attacks helps students analyze the security through a cracker's eyes, exposing weaknesses and determining the most serious risks. Furthermore, demonstrating "real" attacks works wonders when it comes to motivating students to incorporate security in their initial system designs. In particular, courses should study DDoS attacks. As we've seen, it's not a good idea to design a login procedure that simplifies a DDoS attack by closing account access once a few login attempts fail. A security course should introduce different types of DDoS attacks and techniques to mitigate them.

Of course, the decision to teach attack techniques should not be made without seriously considering the potential consequences. It's irresponsible to study attacks without also discussing ethical behavior. It might be wise to style this part of the course as an introduction to penetration testing, to emphasize that attacks are taught so that students can discover system vulnerabilities, not so they can attack systems for financial gain.

Understanding vulnerabilities and attacks on its own won't help us develop secure systems and thus escape the endless cycle of penetration and patch. We must also learn how to teach constructive security¹⁰ and, of course, how to build secure software.¹¹

As our study shows, the authentication of many Norwegian online banking customers was too weak in 2003 and 2004. Since we completed our study period, several banks have improved their customer authentication. Unfortunately, application-layer DDoS attacks are still a serious problem. The security-by-obscurity practice continues unabated, hampering the sharing of important technical information. To achieve and maintain adequate security in the future, Norwegian banks must share security information and let independent security researchers evaluate their online banking systems.

Like many other security researchers, we've experienced the difficulty entailed in alerting banks and other large institutions about security weaknesses. In contrast, the press is quite interested in such research. In one case, for example, we informed the press about our findings

after the bank in question had changed its system. We got much media attention, but with it came some very negative comments (mostly from anonymous sources). A few people even suggested that we had personal financial motives for investigating the banks' security procedures. Such personal attacks only illustrate the importance of holding open and straightforward public debate about security issues. □

References

1. J. Mirkovic et al., *Internet Denial of Service*, Prentice Hall, 2005.
2. R. Anderson, *Security Engineering*, John Wiley & Sons, 2001.
3. J. Leyden, "Telenor Takes Down Massive Botnet," *The Register*, 9 Sept. 2004; www.theregister.co.uk/2004/09/09/telenor_botnet_dismantled.
4. J. Markoff, "Attacks on Windows PC's Grew in First Half of 2004," *New York Times*, 20 Sept. 2004; <http://select.nytimes.com/gst/abstract.html?res=F70F1EF93F5D0C738EDDA00894DC404482>.
5. RSA Security, *The Power Behind RSA SecurID Two-Factor User Authentication: RSA ACE/Server*, solution white paper, 2003; www.internet-security.ag/gen/rsa04/f/power.pdf.
6. C. Adams and S. Lloyd, *Understanding PKI*, 2nd ed., Addison-Wesley, 2004.
7. The Financial Supervisory Authority of Norway, *Risk and Vulnerability Analysis 2003*, Nov. 2003 (in Norwegian).
8. B. Schneier, "Keeping Network Outages Secret," 15 Oct. 2004; www.schneier.com/crypto-gram-0410.html#2.
9. R. Anderson, "Why Cryptosystems Fail," *Proc. 1st ACM Conf. Computer and Communications Security*, ACM Press, 1992, pp. 215–227.
10. C.E. Irvine, "Teaching Constructive Security," *IEEE Security & Privacy*, vol. 1, no. 6, 2003, pp. 59–61.
11. J. Viega and G. McGraw, *Building Secure Software*, Addison-Wesley, 2002.

Kjell J. Hole is a professor in the Department of Informatics and a member of the Selmer Center at the University of Bergen, Norway. His research interests include network and application security. Hole has a PhD in computer science from the University of Bergen. He is a member of the IEEE and the IEEE Computer Society. Contact him at kjell.hole@ii.uib.no.

Vebjørn Moen is a PhD student at the Department of Informatics and a member of the Selmer Center at the University of Bergen, Norway. He is currently visiting Dieter Gollmann's research group at Technische Universität Hamburg-Harburg. His research interests include network and software security. Moen has an MS in computer science from the University of Bergen. Contact him at moen@ii.uib.no.

Thomas Tjøstheim is a PhD student in the Department of Informatics and a member of the Selmer Center at the University of Bergen, Norway. His research interests include PKI and remote electronic voting schemes. Tjøstheim has an MS in computer science from the University of Bergen. Contact him at thomast@ii.uib.no.