Security Analysis of Electronic Voting and Online Banking Systems

Thomas Tjøstheim

The degree philosophiae doctor (PhD)

University of Bergen, Norway

2007

Universitet i Bergen

Contents

Abstract

The main focus of this dissertation is on security analysis of electronic voting and online banking systems. Six papers form the basis of the thesis and include the following topics: a model for analysis of voting systems, a case study where we apply the proposed model, a new scheme for remote electronic voting, and three case studies of commercial online banking solutions in Norway.

Acknowledgments

Going to work the last three years has for the most part been a pleasure. I have had the privilege to work with some very talented people, and it has been very motivating to study security in real world systems. Kjell Jørgen Hole convinced me to start a Ph.D. and has been my mentor throughout the period. Thanks for always encouraging me and for being available whenever I have needed help.

Thanks to Tor Helleseth for supervising me during my Masters and for suggesting the field of electronic voting, by which I became very fascinated. Many thanks to Geir Røsland for introducing his ideas on remote electronic voting and for triggering my interest in free diving and underwater hunting.

Thanks to Peter Ryan for inviting me to his group in Newcastle and for supervising me during my stay. I had a brilliant year both academically and socially. Thea Peacock has been an excellent research partner and reviewer on many papers. Thanks also to many friends at the Center for Software Reliability, I would not have enjoyed my stay to the same degree without you guys.

Thanks to all the members of the Nowires Research Group for both work related collaboration and social activities. Motivating myself to go to work each day has been easy with you guys around.

Thanks to my family for supporting me, and my girlfriend Tove Ragna for showing so little and so much interest in my work, accompanying me on conference trips, calming my nerves before important talks, and for helping me with practical computer stuff, like localising the volume button on my computer.

Introduction

Introduction

Thomas Tjøstheim

In today's society digital services have become an important part of our lives. National systems like electronic voting and online banking have been introduced to save money and increase convenience. However, if people are to use these systems to cast their votes and transfer money, they have to be convinced that these systems operate correctly. Sensitive information should be kept secret and data should be protected from malicious or accidental modification. The systems should also be available, as significant downtime could create chaos and economic loss.

Electronic voting systems are being deployed in many countries [4], and have the advantage that they can improve the speed of counting and accessibility for disabled voters. Accuracy is another advantage, since many ballots marked by humans are left unrecognisable, e.g. due to bad handwriting. However, there have been many accusations of election fraud [12]. We need to ensure that the votes are cast as intended and counted as cast, but also that they remain anonymous, to avoid vote buying or coercion. Undetected fraud could have serious consequences.

Online banking is growing in popularity worldwide. The banks save money by avoiding expensive paper handling and personal interaction with the customers. Costs saved also benefit customers, as many online banks can offer higher interest rates. Although online banks are convenient and cost-saving, new challenges are introduced when banking services are made available online. The customers have to be correctly authenticated, the banks should be available at all times, and auditing mechanisms must be in place to detect suspicious customer behavior.

Security analysis of electronic voting and online banking systems are the themes of this thesis. It consists of three papers on electronic voting [27, 26, 28] and three on online banking [25, 13, 14].

1 What is Security?

The main components of security are *confidentiality*, *integrity* and *availability* [9]. Confidentiality is the assurance that information can only be accessed by the intended and authorised recipients. Access control mechanisms can be used to preserve confidentiality. Cryptography can for example be used to make the information unreadable to anyone except the parties who hold the cryptographic key to unscramble the data.

Integrity is the assurance to an entity that data has not been altered (intentionally or unintentionally) during transmission, from origin to destination. Integrity mechanisms can be divided into two classes: prevention and detection mechanisms. Prevention mechanisms seek to maintain integrity by thwarting any unauthorised attempts to modify the data, while the purpose of detection mechanisms is to signal if the data's integrity is breached.

Availability is the assurance to authorised users that information or resources can be used as desired. Attackers can deliberately deny access to a service by carrying out denial of service (DoS) attacks. Such attacks try to break the usual access patterns to make the system unavailable. Protecting against these attacks is challenging as it is difficult to predict what constitutes expected patterns of use.

1.1 Breaking Security

Security is only as strong as the weakest link. Where the attacker can find the weakest link, the probability of an attack will be the highest. A weak link is a *risk* to the system, and can be seen as a function of the exploitable *vulnerabilities* and the danger from the *threats* [15]. Vulnerabilities are flaws in the design of the system or implementation bugs that an attacker can exploit. According to the National vulnerability database [5] a total of 6600 vulnerabilities were reported just in 2006, where 42% of these had a severity level of high. Vulnerabilities are present in virtually all software and are difficult to avoid, since testing only can prove the presence of vulnerabilities and not their absence.

Threats to the system could be either malicious or unintentional. In this thesis we are mostly concerned with the malicious threats, i.e. attackers with the capabilities and intentions to exploit vulnerabilities [15].

1.2 Secrecy is Not Security

Security by secrecy is a method in security engineering that tries to enhance the security of a system by keeping information about algorithms, countermeasures, documentation, etc. secret. Many commercial companies advocate secrecy, while most security experts agree that security by secrecy is a flawed tactic. Secrecy is a barrier to the transparency of the system. Without inspection by independent experts, there is a higher chance of serious threats going undetected. A review by the individuals who developed the system can be of limited value, compared to contributions from people without any invested interests in the system [8].

We have seen how Norwegian online banks have supported secrecy, fearing that the discovery of vulnerabilities could have economic consequences or cause a loss of reputation [13]. Similarly, commercial voting companies like Diebold and Nedap/Groenendaal have long refused to share the details of how their voting machines work. In [21] and [11], it is described how the source code was eventually leaked, and how a large number of serious threats were discovered with the potential to turn around elections.

2 Security Analysis

In this thesis, a security analysis of a computer system is an assessment of how well that system ensures the security requirements in face of a malicious attacker. Security analysis is challenging, due to the many ways an attacker can try to take advantage of the system environment. Typically, an attacker will attempt to crack the system by altering the assumptions under which the system was made secure.

Analysis at different levels of abstraction is necessary to assure that the security requirements are met. A high level analysis may focus on reading the product documentation, studying the architecture, and testing for well known flaws. Another approach is to use a catalog of threats [1] to test the system's resistance to specific threats. A protocol level analysis on the other hand focuses more on the technical core, e.g. using a formal language to verify properties of a cryptographic protocol. The different approaches all have their limitations and cannot possibly catch all the threats to a system. A method for testing the effectiveness of the analysis, is to create vulnerabilities deliberately and then see how many of them are detected. Analysis

techniques should also be updated rapidly and renewed as new attacks are discovered.

In this thesis we take a system-based approach to analysis, i.e. we focus on how the different parts and players of the system interact. Modeling real world systems is difficult, as systems can behave in unexpected ways and are constantly changing to avoid threats or to add new features. We believe a system-based approach is necessary to determine the vulnerabilities most likely to be exploited by hackers. As Anderson points out in [7], systems typically fail not due to cryptographic failures, but as a result of crude system-based failures.

3 Electronic Voting Systems

Electronic voting systems include all voting systems that make use of electronics to cast or tally votes. A variety of approaches exist with various degrees of electronics incorporated, examples include optical scan systems, direct-recording electronic (DRE) voting systems, telephone and Internet voting, and paper-based electronic voting [4].

3.1 Voting Challenges

In voting systems all participants may have the motivation to cheat. Potentially, the election officials could cheat the voters, voters could cheat the election officials, and coercers and vote buyers could influence the voters. In addition, we want to enable the voters to fool the coercers to prevent coercion or vote buying [24].

Conflicting requirements constitute another challenge. On one hand we want to enable voter verifiability, i.e. voters should be able to verify that their ballots enter the tally and that the ballots are correctly counted. While on the other hand it should not be possible for a voter to prove how she voted.

3.2 Verifiability in Voting Systems

By enabling verifiability of the election process we can limit the possibilities of election fraud. The votes should be cast as intended and counted as cast, ideally without trusting anyone. In traditional manual, paper-based elections a voter does not get any feedback indicating whether her individual vote was

correctly received or tallied. Voters have to depend on a process involving the election officials to correctly register and tally the votes.

In [22] Ryan presents two main methods for evaluating the assurance of voting systems: prior evaluation (testing) and run time monitoring. The first approach is illustrated by how DRE voting machines are tested. Typically, the DREs are tested prior to election day in addition to blind tests to verify that they operate correctly. The level of verifiability is very low, as a voter does not receive any proof of her vote being correctly cast other than the "thank you for your vote" message displayed on screen.

A mechanism to add robustness to the voting process is the use of a voter verifiable paper audit trail (VVPAT). In this scheme, the voter verifies the paper copy of her ballot, e.g. under glass (if the Mercuri method is used [17]) before it is put in a separate box. This paper trail is used to audit the election and also acts as a paper backup in case recounts are requested, due to, for example, errors or a close election.

The run time monitoring approach is typified by the new trend towards verifiable voting systems, which have minimal reliance on the players, i.e. the voters, election officials and technical components [27]. Each voter is given a verifiable receipt such that they can individually verify that their vote has been correctly entered into the tallying phase. Cryptographic constructions make this possible without compromising ballot secrecy, although Rivest [20] has shown with the ThreeBallot voting system that cryptography is not always an absolute necessity. Public bulletin boards are used to post proofs that the ballots enter the tallying phase and are counted correctly. Some notable examples of verifiable schemes are Prêt à Voter [10], Punchscan [3] and VoteHere [18]. These strive for complete transparency, given the constraints imposed by the ballot secrecy requirements, and seek to achieve assurance via detailed monitoring of the process rather than having to place trust in the system components [27]. Note that with the run time monitoring approach emphasis has been put on verifying the election rather than the system.

3.3 Moving Towards Practical Voting Systems

The new trend towards verifiable voting systems increases the voters involvement in verifying the correct operation of the election processes and thus the validity of the election results. A remaining challenge is how to create these systems, such that most voters are able to understand the fundamental principles and can gain confidence in the election process. Voting systems

consist of many components and are trying to meet intricate requirements. For many people it may simply take too much time or effort to try to figure out how a system works. In addition, prior knowledge of computer systems, programming languages, cryptographic principles or security standards may be necessary. As Ryan and Peacock point out in [24], there is a delicate balance between streamlining the voter role and letting the voter contribute to the overall dependability of the system. It is not realistic or desirable that every voter needs to understand every aspect of a voting system. Rivest[1] suggests that each user should be able to get an understanding of the system corresponding to the relevant background and the time she is willing to spend inspecting the system. In addition, it is necessary with evaluation by independent security experts to discover any flaws or bugs hidden in the technical details.

While transparency is important to enable thorough evaluation of a voting system, the transparency of the system could work against it, e.g. if a large number of errors are reported. This could dissuade users from using the system, and in the worst case cause it to be abandoned altogether.

4 Online Banking

Online banks are convenient and cost-saving, but also offer better opportunities for attackers. Only legitimate customers should be able to access their accounts and the integrity of transactions must be preserved.

4.1 Online Banking Security

Online banks in Norway have been very secretive about the internal workings of their systems. Our focus is therefore to analyse online banking solutions from the customers' perspective, i.e. how the customers are authenticated and the risks the customers take when using online banking services [25, 13, 14]. A surprising variety of authentication solutions exist. However, many of the solutions have turned out to be weak. Online banking fraud is increasing and will only continue to grow as attackers get more competent and organised. Around Christmas in 2006, six Norwegian online banks were the targets of online fraud. Attempt were made to steal a total of 2,2 million NOK from customer accounts [2].

[1] Email correspondence with Ron Rivest.

Going back to the 2003–2004 period, customers in several online banks were authenticated by entering a user ID in combination with a PIN (Personal Identification Number). These solutions where shown to be vulnerable to brute-force and DoS attacks [25, 13]. Key logging and phishing are also risks as the same PIN and user ID are used for each login. Two-factor authentication utilises two different methods to authenticate customers, and can thus improve security. A PIN calculator is an example of two-factor authentication, it requires the customer to have something (the PIN calculator) and to know something, a secret PIN that activates the PIN calculator to produce a new and often longer one-time PIN. While two-factor authentication protects better against brute-force, phishing and key logging attacks, man-in-the-middle and Trojan attacks are still threats. A fake web site could for example pick up a user's credentials or a Trojan horse could steal a user's session.

An alternative is to use a public key infrastructure (PKI) [6]. By authenticating users with the use of digital signatures, sensitive information like passwords do not have to be transmitted over a public network. Instead the customer's private key can be stored locally, for example on a smart card, such that challenges can be signed on the smart card or on the home computer. In order to benefit from the services of a PKI, it is essential that the PKI is implemented and maintained in the right way. The customers' private keys must be kept secret, the public keys must be bound to certified user IDs and non-repudiation information must be correctly stored and handled in the case of a dispute. BankID, the next generation online banking solution in Norway is based on a PKI. However, they have made some unconventional design choices, which unfortunately has led to the discovery of several serious vulnerabilities [14].

4.2 Educate Users or Improve Security Practices?

It is far from trivial to build secure online banking systems. A careful balance needs to be found between usability and security. Users should be able to understand and use the system, while at the same time their money should be kept safe. The banks also face technical challenges as defense against one attack may increase the vulnerability to another attack, e.g. protection against brute-force attacks may increase the vulnerability to DoS attacks [13].

Customers' computers are often seen as the weakest link as they are vul-

nerable to many attacks While we cannot stop people from being security unconscious, we can assume that the customers' computers are insecure and design security accordingly. Security should be considered from the start of the development life cycle and is not some magic potion one adds to the system at the end. The security services which are needed and their implementations should be determined early in the process. Ensuring security is also a continuing process, new patches and solutions must be found when new attacks are discovered.

5 Summary of Papers

The main focus of the papers in this thesis is on security analysis. Three papers analyse security in electronic voting systems, while three papers analyse security in Norwegian online banks. Four papers have been published in international conferences or journals [27, 13, 25, 28] while two papers [26, 14] have been submitted for publication.

5.1 A Model for System-Based Analysis of Voting Systems [27]

In order to ensure the election requirements we need analysis at different levels of abstraction. It was shown in [16, 23] that certain vulnerabilities only become apparent when taking a system-based view, i.e. considering the interactions between the various components of a scheme. We propose a model for system-based analysis of voting systems that is more comprehensive than previous work. The model was developed in a stepwise manner and defines a set of components and associated threat categories. Starting with the components of a base voting system, we considered the players interacting with each component and then derived a set of main threat categories. Additional components where then added to include more sophisticated systems that include for example voting devices and verifiable receipts. By breaking the model into different components, we make the model more general and can tailor it to the scheme being analysed. At the same time we only include threats that are directly related to a component and more easily avoid repetition of threats, that can occur when compiling a catalog of threats. The use of the model is demonstrated with a case study of the Randell and Ryan scheme [19], a version of Prêt à Voter that aims to promote voter understandability.

5.2 A Case Study in System-Based Analysis: ThreeBallots and Prêt à Voter [26]

We wanted to verify the validity of our model by carrying out a system-based analysis of two voter verifiable schemes, the ThreeBallot voting system [20] and Prêt à Voter [10]. A comprehensive analysis of the schemes was performed, and we were able to discover several new threats in both schemes. Threats to the ThreeBallot voting system include among others, techniques for linking the three ballots and methods for adding or subtracting votes for a candidate. Threeballots is not practical as it stands, but is of immense theoretical importance as it demonstrates that it is possible to design a voter verifiable scheme without the use of cryptography. The base version of Prêt à Voter is vulnerable to authority knowledge, chain voting, the voter retaining the left hand strip and kleptographic attacks. However, these attacks can be mitigated through distributed generation of ballot forms, the use of scratch strips, and the use of "dummy" left hand side strips. A possible trade-off is that these countermeasures can add an extra layer of complexity, which makes it more difficult for voters to understand how the system works.

5.3 Remote Electronic Voting Using Verifiable Chain Encryption [28]

In a remote voting scenario coercion and vote buying become increasingly difficult to protect against, due to the many ways a coercer can influence a voter. We wanted to demonstrate that it is possible to design a practical and receipt free remote electronic voting scheme for large scale elections, given certain assumptions. A new verifiable election encryption function for chain encryptions based on an extension of El Gamal is presented. An individual votes remotely by encrypting her ballot with the proposed encryption function, and posts the encrypted ballot to a restricted bulletin board (RBB). Voters can overwrite their vote by re-voting remotely until a previously agreed deadline. Voters do not have access to the RBB, but a set of representatives from competing parties, called the scrutinizers verify that ballots are correctly posted to the RBB. Voters also have the possibility of casting a vote in a manual, paper based voting election which will supersede their remotely cast vote. After the initial voting phase, the ballots on the RBB are re-encrypted with the election encryption function and posted to a public bulletin board (PBB). A set of tellers decrypt the votes by performing

a set of predetermined re-encryptions. Each teller posts proofs of correct operation to the PBB, such that anyone interested can verify correct procedure of the mixing. A coercion free Internet voting scheme is not practically realisable, but we limit the possibilities in this scheme, since voters can revote remotely, overwrite their remotely cast vote in a traditional poll place election, in addition to practical security requirements.

5.4 Vulnerabilities in Online Banks [25]

In 2003–2004 we studied the security of several Norwegian online banks from the customers' perspective. We discovered several simple but powerful attacks against at least three Norwegian online banks. Common to the vulnerable banks was the fact that they authenticated the customers by asking for a structured user ID (e.g. a social security number (SSN) or an account number) in combination with a PIN. Structured user IDs are easy to generate, which opens up the possibility for brute-force and DoS attacks. The basis for the attacks is to generate a set of user IDs that will cover user IDs belonging to customers of an online bank. We describe a brute-force attack that attempts to guess random customers' PINs, by trying different pairs of user IDs and PINs. Whether the PIN is static or dynamic does not matter, as randomly generated PINs are tried together with the generated user IDs. A DoS attack can be launched by guessing a PIN incorrectly a given set of times (usually three or five) for each user ID to trigger the bank's lockout policy. If a bot network is used to distribute the attacks, we believe it would be difficult for the online banks to prevent the described brute-force and DoS attacks.

5.5 Case Study: Online Banking [13]

We wanted to expand further on the results from the previous paper, and carried out a broader case study of the Norwegian online banks in the period 2003–2004. Online banks authenticating customers with the aid of a PIN calculator where shown to be vulnerable to brute-force attacks. A PIN calculator takes as input a secret PIN only known by the customer and produces a new PIN. The PIN calculators we studied generated a new PIN for given time slots. However, due to problems with clock drift between the bank server and the customer's PIN calculator a larger window of PINs was allowed. One Norwegian online bank allowed a window of 19 consecutive

PINs, effectively reducing the active number of digits in the PIN code. We also discovered a new way to filter SSNs. Many university and government employees belong to the Norwegian public service pension fund (NPSPF). The NPSPF authenticated customers only by asking for their SSN. A simple script that generates SSNs therefore made it possible to link names and addresses to corresponding SSNs.

The vulnerabilities found in 2003–2004 may have been due to the influence of ATM (automatic teller machine) design, where the same mental models have been used for online banking solutions. The designers may not have been aware of the fact that brute-force and DoS attacks scale differently against Internet based solutions. We believe that many of the identified vulnerabilities could have been avoided, if the banks had been more open about their systems and allowed for independent evaluation by security experts.

5.6 Next Generation Internet Banking in Norway [14]

BankID is a new PKI-based security infrastructure for web applications proposed by the Norwegian online banking community. At the end of 2006 it was used to authenticate 600,000 Norwegian bank customers, but the long term goal is to let BankID become a new standard for authentication and digital signatures in commercial web applications. We evaluated the potential risks to customers using BankID, and based our evaluation only on publicly available information.

BankID contradicts traditional X.509 PKI design. The customers' private keys are stored in a central depository, and not locally for example on a smart card. A customer enters her SSN, a one time PIN and a static password to access her private keys. An attacker who is able to steal a user's login credentials can steal that user's identity and sign documents on behalf of the user. A severe flaw enables distributed DoS attacks by generating SSNs and entering the wrong PIN code three times.

Another flaw is the possibility of initiating a man-in-the-middle attack by tricking the customer to download modified HTML code. The parameters specifying URLs can be changed to fool the customer into communicating with the BankID server and central infrastructure through a proxy server. The attacker can then steal the session after the user has authenticated herself.

Another concern is non-repudiation. How the customers private keys are protected and accessed without leaking information to insiders is not

explained in public BankID documents. This combined with the fact that no independent third parties are used to collect evidence in the BankID infrastructure questions the achieved level of non-repudiation.

6 Conclusions and Further Work

In this thesis we have analysed electronic voting and online banking systems. We have proposed a model for system-based analysis along with several case studies and have shown that the model is both more systematic than previous work and applicable to a wide range of different schemes. A further step could include assessing the probability of threats occurring and studying how to handle discovered threats. We have also presented a new verifiable remote electronic voting scheme, applicable for large scale elections. A natural next step could be to extend the proposed model for system-based analysis, to also include remote voting systems, and to perform a more comprehensive analysis of the scheme.

The analysis of online banking solutions has focused on commercial online banking systems in Norway. We were able to find serious vulnerabilities in all of the authentication solutions we studied, using only publicly available information. Customers have a right to know how secure the online banks really are. Openness and verifiability are vital to help people understand how these systems work, and to enable security analysis. To enable a thorough analysis, ideally as much information as possible about these systems should be available for public scrutiny. Cryptographic keys and operational passwords should of course not be revealed. A balance needs to be struck between the positive aspects of being able to find solutions to new threats, versus the risks that vulnerabilities are exploited. Whether the system is already in use, how the threats are reported, and how quickly the threats can be mitigated are all important factors to consider.

There is no such thing as perfect security. An analysis can assess how secure a system is to certain threats and can say something about the likelihood of vulnerabilities being exploited. We need analysis at different levels but also by different people, as the challenges are not only technical—economy, law and social issues are also important to consider.

References

[1] The machinery of democracy: Protecting elections in an electronic world (full report), 2006. Brennan Centre for Justice, NYU School of Law. http://www.brennancenter.org/dynamic/subpages/download_file_36343.pdf.

[2] Ny nettbanksvindel rammer norge, 2006. http://www.dagensit.no/bedrifts-it/article1084356.ece?WT.svl=article_re%admore.

[3] Punchscan, 2006. http://www.punchscan.org.

[4] Electronic voting, 2007. http://en.wikipedia.org/wiki/Electronic_voting.

[5] The national vulnerability database, 2007. http://nvd.nist.gov.

[6] C. Adams and S. Lloyd. *Understanding PKI*. Addison Wesley Prof, 2002.

[7] R. Anderson. Why cryptosystems fail. In *Conference on Computer and Communications Security*. ACM, 1993.

[8] R. Anderson. *Security Engineering: A Guide to Building Dependable Distributed Systems*. Wiley, 2001.

[9] M. Bishop. *Computer Security: Art and Science*. Addison Wesley Prof, 2002.

[10] D. Chaum, P. Y. A. Ryan, and S. Schneider. A practical, voter-verifiable election scheme. In *European Symposium on Research in Computer Security*, number 3679 in Lecture Notes in Computer Science. Springer-Verlag, 2005.

[11] R. Gonggrijp, W. J. Hengeveld, A. Bogk, D. Engling, H. Mehnert, F. Rieger, P. Scheffers, and B. Wels. Nedap/Groenendaal ES3B voting computer a security analysis. Unpublished draft. http://www.wijvertrouwenstemcomputersniet.nl/images/9/91/Es3b-en.pdf, 2006.

[12] A. Gumbel. *Steal This Vote*. Thunder's Mouth Press, U.S.A, 2005.

[13] K. J. Hole, V. Moen, and T. Tjøstheim. Case study: Online banking. In *IEEE Security and Privacy*, vol.4, no.2, pp.14–20, 2006.

[14] K. J. Hole, T. Tjøstheim, V. Moen, L-H. Netland, Y. Espelid, and A. N. Klingsheim. Next generation Internet banking in Norway. Interim report, 2007.

[15] A. Jones and D. Ashenden. *Risk Management for Computer Security*. Elsevier, 2005.

[16] C. Karlof, N. Sastry, and D. Wagner. Cryptographic voting protocols: A systems perspective. In *USENIX Security Symposium*, 2005.

[17] R. Mercuri. A better ballot box? *IEEE Spectrum Online*, October 2002.

[18] A. Neff. Practical high certainty intent verification for encrypted votes, 2004. `http://www.votehere.net/documentation/vhti`.

[19] B. Randell and P. Y. A. Ryan. Voting technologies and trust. *IEEE Security & Privacy*, vol.4, no.5, pp. 50–56, 2006.

[20] R. L. Rivest. The ThreeBallot voting system. Unpublished draft, `http://theory.lcs.mit.edu/~rivest/RivestTheThreeBallotVotingSystem.pdf`, 2006.

[21] A. Rubin. *Brave New Ballot: The Battle to Safeguard Democracy in the Age of Electronic Voting*. Morgan Road, 2006.

[22] P. Y. A. Ryan. Of elections and electrons. `http://www-formats-ftrtft.imag.fr/accepted/ryan.html`.

[23] P. Y. A. Ryan and T. Peacock. Prêt à voter: a systems perspective. Technical Report CS-TR-929, University of Newcastle upon Tyne, 2005.

[24] P. Y. A. Ryan and T. Peacock. Putting the human back in voting protocols. In *Fourteenth International Workshop on Security Protocols*, Lecture Notes in Computer Science. Springer-Verlag, 2006.

[25] T. Tjøstheim and V. Moen. Vulnerabilities in online banks. In *NordSec*, 2005.

[26] T. Tjøstheim, T. Peacock, and P. Y. A. Ryan. A case study in system-based analysis: The Threeballot voting system and Prêt à voter. In *VoComp*, 2007.

[27] T. Tjøstheim, T. Peacock, and P. Y. A. Ryan. A model for system-based analysis of voting systems. In *Fifteenth International Workshop on Security Protocols*, 2007.

[28] T. Tjøstheim and G. Røsland. Remote electronic voting using verifiable chain encryption. In *Frontiers in Electronic Elections*, 2006.

Paper I: A Model for System-Based Analysis of Voting Systems

A Model for System-Based Analysis of Voting Systems

Thomas Tjøstheim, Thea Peacock and Peter Y. A. Ryan

Abstract

There has recently been keen interest in the threat analysis of voting systems. While it is important to verify the system itself, it has been found that certain vulnerabilities only become apparent when taking a "system-based" view, i.e. considering interactions between the various components of a scheme. Threat analysis has so far been of three main forms: system-based, protocol-level and taxonomy check-lists. We discuss these approaches before presenting a model for system-based analysis of voting systems that is more systematic than previous work. The model is described in detail, and demonstrated with an example from a case study of the Randell-Ryan "Scratch Card" voting system.

1 Introduction

There has been a recent trend towards automated voting systems in an attempt to improve the speed and accuracy of elections, and to encourage voter turn-out. However, many of these new schemes have proven to be flawed, with cases of election fraud, e.g. in the US [12, 23]. "Black box" systems are of particular concern, e.g. those making use of Direct Recording Devices that give no proof that a vote has been correctly recorded [15]. This has generated much interest in research on verifiable voting systems, which have minimal reliance on the players, i.e., voters, election officials, etc., and technical components, such as the hardware and software behaving as intended. Notable examples are Prêt à Voter [8], Punchscan [3], and VoteHere [4], all of which aim to provide a high degree of transparency in the system. While cryptography is often used to enable verifiability without compromising voter

privacy, Rivest has shown with the ThreeBallot voting system that this is not, in fact, an absolute necessity [22].

Despite the progress in developing high assurance voting systems, there is nevertheless the need for careful analysis to ensure that requirements such as eligibility, coercion-resistance and accuracy are met. In [14] Karlof et al. carried out a system-based analysis of Chaum's visual crypto scheme [7] and Neff's original scheme [19, 18], i.e., taking into account interactions between the various components in each scheme. In doing so, they identified potential threats such as subliminal channels and "social-engineering"-style attacks. In a similar analysis, Ryan et al. [25] showed that Prêt à Voter [8], is robust against many of the threats mentioned in [14], but identified further possible vulnerabilities such as chain-voting and authority knowledge. See [25] for details.

Although highly useful, this type of analysis is rather ad-hoc and hence, may not uncover all the possible threats in a scheme. At a lower level of abstraction, a protocol-level analysis [20, 16] may be more systematic but is limited to the technical core of the protocol. Another approach is to develop a "catalogue of threats" [1], but perhaps as a reflection of the immensity of this task, aside from [2] there is little work to date in this direction.

In this paper, we propose a model for an analysis of threats in voting systems that is systems-based, but considerably more systematic than previous similar work [14, 25]. While [2] has a largely technical focus and concentrates on DRE systems, our model operates at a higher level of abstraction and is not scheme-specific.

In this model, the main components of a scheme such as the ballot form, voting booth, etc. are identified, and the possible threats to each component, at each phase of the protocol are considered in turn. In this way, it provides a guideline for evaluation of the system with the detail of a protocol-level analysis, but at the same time taking interactions between the various components directly into consideration. An advantage of this model is that apart from offering a more systematic approach to analysis, the components can be selected as appropriate and thus, tailored to the scheme being analysed. In addition, by working through the threat categories in the model, and at the same time applying appropriate reasoning to the scheme, the analyst is arguably better able to identify new threats than if using a catalogue of threats.

We have striven to keep the model as general as possible, hence, it can be used for a range of different systems: from manual, paper-based voting, such

as the current UK system, to more sophisticated systems that incorporate, e.g. voting devices and verifiable receipts.

The structure of the paper is as follows. In Section 2 we describe the model in detail, and in Section 3, explain how it might be used to analyse a voting system. In Section 4 we discuss the results and possibilities for future work.

2 A Model for Analysis of Voting Systems

We introduce the model in a step-wise manner, beginning with a simple manual voting system, such as the one currently used in the U.K. We then extend this model to include the capability for automated vote recording and tallying, a paper audit trail and verifiability via receipts, which the voter can check against a Web bulletin board after a vote is cast. As will be seen shortly, this is done by adding the necessary components. Hence, the model offers a high degree of modularity, as the components can be selected as appropriate to the scheme being analysed.

To derive the model, we first examined the main phases in a typical voting protocol: voting and tallying. As is the case in most current voting systems, we assume that the pre-election set-up has taken place. Typically, the electoral roll would have been established, and could include setting up of cryptographic keys, printing of paper ballot forms, ensuring that ballot boxes are empty, etc. We also assume that there is a registration process in which voters are authenticated and checked for eligibility. Note however, that a more complete analysis of a voting scheme should also include these processes. We consider them as future extensions to the model.

Taking a high-level view of the protocol, we then isolated the main components involved with each phase. The components in the model will be described in detail shortly. Working through the steps in the voting protocol, we identified possible threats that could occur directly in relation to each component. As we only consider the immediate threats, we avoid the tendency for repetition that can occur when compiling a catalogue of threats.

For uniformity, the possible threats were organised into threat categories, such as “ballot stuffing”, “absence of verifiability”, etc. Although certain threat categories do appear in several components, we only consider the threats that are directly applicable in each case. In an analysis, it is important that the details of the particular scheme be considered with care when

deciding whether or not a particular threat category applies, and if so, the way in which the threat may be manifested. It is possible that not all the threat categories in the model will apply in each case, as this clearly depends on the scheme being analysed. However, identifying robustness against a particular threat is useful in highlighting the strengths of a scheme.

Note that we do not directly identify the players in a voting scheme, such as the voters and election officials. However, as will be seen shortly, many of the threats in our model can arise from interactions between certain players and the above components. Note also that we define the components in terms of the generic case, which could be adapted according to the particular system under analysis.

For example, evaluating the threats arising from storage of votes during the voting phase will depend on whether ballot forms are cast into a ballot box or whether votes are recorded on a memory card. The general threat categories are covered by the ballot storage component.

The present model excludes remote voting systems, as this adds considerable complexity to ensuring the coercion-resistance of a scheme. Instead, we start with a model for analysis of booth-based systems, and consider remote voting as a future extension.

2.1 The Base Voting Model

In the base model, we identify two phases in the protocol: voting and tallying. A description of the system, along with the main components involved at each phase, is as follows:

During the voting phase, the voter marks her choice on the ballot form in a booth, then exits and casts the marked form in a ballot box. An official ensures that she casts one ballot form only, but should not be able to learn her vote choice or to link the voter to her cast ballot form. Here, the main components are the *voting booth, ballot form* and *ballot box*.

Tallying commences after the close of voting. Ballot boxes are collected and transferred to a designated tallying place. Officials open the ballot boxes and count the votes, watched by a team of observers. Local results are collated, and the final tally is published. The main components involved are the *voting booth, ballot form, ballot box* and the *election results*. As we aim for generality with the model, the components are chosen by taking a high-level view of a system.

We next describe the components in terms of their main functions and security requirements. This is necessary for determining relevant threats.

Ballot form - Record the voter's candidate choice(s). Once the voter has marked her choice, it should not be possible to modify it. There should be no way to link a ballot form to a voter after it has been cast.

Ballot storage - Securely store the cast ballot forms. No voter should be able to cast more than one ballot form into the box, and it should not be possible to insert fake votes.

Voting booth - Provide privacy while the voter marks the ballot form. A voter should be able to make her selection without outside interference, and there should be no opportunity to monitor or record the process.

Election results - The final count of all (legitimate) votes.

Having established the basis for an analysis of threats to a voting system, we now present the first elements of our model. The possible threats associated with each component are given in Figures 1 - 4. There are several points to clarify. Firstly, note that in Figure 1 we list possible threats that could arise from both a paper ballot form and one that is generated by a device. We discuss the differences in Section 3. Secondly, in Figure 3, we show possible threats to a ballot storage component to allow a later extension for automated vote recording. Thirdly, for generality, we have included threats that would apply to schemes that are more sophisticated than the paper-based manual system, such as those that make use of encryption.

Finally, for all components in the model, the property violated is listed alongside each threat. Here, we consider the main properties required of secure systems, i.e., confidentiality, integrity and availability, rather than the traditional requirements of voting systems such as ballot secrecy, accuracy, verifiability, etc [16, 10]. The latter could be regarded as specialisations of the former, and we find that they are rather too fine-grained for a generic model. This is particularly true of voting system requirements which tend to "overlap", such as coercion-resistance and receipt-freeness. A system may for instance satisfy receipt-freeness, but not coercion-resistance.

It is worth noting that some threats potentially violate more than one property, e.g. chain-voting and early publishing of the election results can undermine both integrity and confidentiality. However, we list these threats under integrity, which could be considered as the top-level requirement of an election: that the final count should accurately reflect the true intention of the voters. In a chain-voting attack, the coercer obtains a fresh ballot form and marks his choice. He then threatens or bribes a voter to cast it at the polling station, in return for an unused form. Hence, one or more voters may be coerced into voting in a certain way, against their free will. Partial results published ahead of time may influence voters who have not yet cast their votes. At the same time, an attacker may be able to make inferences about the identities of voters who have voted or have yet to vote. Clearly, confidentiality is also at risk in both cases.

Threat	**Property violated**
Identifiable information added by voter/official	
Voter identifiable from ballot form	
Authority knowledge	Confidentiality
Voter's choice incorrectly represented	
Ballot form spoiled	Integrity
Ballot form faked	

Figure 1: **Ballot form**

Threat	**Property violated**
Voter's activity monitored	Confidentiality
Voter records own choice	
Voter's choice influenced	Integrity
Voter smuggles out unmarked ballot form	

Figure 2: **Voting booth**

We next extend the base model by adding a voting device for automated vote recording.

Threat	Property violated
Ballot stuffing	Integrity
Ballot spoiling	

Figure 3: **Ballot storage**

Threat	Property violated
Early publishing	Integrity
Absence of verifiability	
False/erroneous count	

Figure 4: **Election results**

2.2 Extension 1: Adding a Voting Device

A description of a generic scheme using a voting device is given below.

The device authorises that the voter has the correct credentials to use the device, and then presents the vote choices to the voter. She makes her selection, e.g. on a touchscreen, which the device writes to a storage medium, such as a memory card.

We make several assumptions in this extension to the base model. Firstly, that during registration the voter optionally receives a device, e.g. smart card or one-time password which she presents to the voting machine during authorisation. Secondly, that the device is located in a booth, and the voter is checked against the electoral list during the authorisation process. Further, that the storage media are collected at the end of the voting phase.

After the voting phase has ended, officials collect the storage media from each of the voting machines. The media are transferred to a device which extracts and counts the votes. Note that although we have not identified the counting device as a component, possible threats introduced to a scheme are covered by the election results "component". As before, the results are collated, and the final tally is published. Note that there could be a network of local counters, or a central counter. With the former, there is possibly greater opportunity for data corruption in transit or early publishing of election results.

Note that the model is still useful for schemes such as Prêt à Voter in which the device only scans the voter's receipt, as the possible threats can

be evaluated as appropriate. This is illustrated shortly in an analysis of the "Scratch Card" voting system which is based on Prêt à Voter.

The functions and requirements of the voting device and storage medium are given below.

> **Voting device** - Authorise the voters' credentials and present vote choices. Record the voter's choice and write to storage media. It should not be possible to add any identifying information to a vote choice, alter, duplicate or delete it. The device should not be able to generate fake votes.
>
> Although the function of the storage medium is analogous to the ballot storage component described previously, we re-state them in terms of the physical differences to facilitate visualisation of potential threats.
>
> **Storage medium** - Store the voter's choice. Once written to the medium, it should not be possible to alter or delete any data.
>
> The device should be protected against any tampering. Likewise data transfer at the end of the voting period.

Potential threats introduced by adding a device are given in Figure 5.

Threat	**Property Violated**
Identifiable information added	Confidentiality
Voter's activity monitored	
Faulty authorisation	Integrity
Voter's choice incorrectly/not recorded	
Denial of service	Availability

Figure 5: **Voting device**

2.3 Extension 2: Adding a Paper Audit Trail

A second extension to the base model is a paper audit trail. With a voter verifiable paper audit trail (VVPAT) [17] mechanism, a paper copy is made of the voter's selection and verified by the voter. The copies are securely

stored as a back-up in case a manual re-count is necessary, e.g. if automated tallying fails or if the final tally appears suspicious in any way.

We assume that the voting device, such as the one in Section 2.2, produces a printed receipt for each vote cast. As in the "Mercuri Method" [17], the device displays the receipt under a clear screen. The voter verifies the receipt, which is then placed mechanically in a sealed box so that the voter cannot leave the polling station with it. The idea is that if the receipt is incorrect, an official could void the entry and provide the voter with another chance to vote. This is clearly a risk to voter privacy as indicated in the model.

Similar to a VVPAT, a verifiable encrypted paper audit trail (VEPAT) [24] acts as a paper back-up in case a manual recount of votes is necessary, but is intended for schemes in which the voter's choice is encrypted. Since copies are made of an encrypted vote, the risk to voter privacy is reduced. Since the threats specific to a VEPAT will be covered by the ballot form component, both types can be analysed using the same component in the model.

Possible threats to a scheme arising from the paper audit trail are shown in Figure 6.

Threat	**Property violated**
Voter identifiable from receipt	Confidentiality
Voter's choice noted by official	
Mismatch between voter's choice and paper copy	Integrity

Figure 6: **Paper audit trail**

2.4 Extension 3: Adding a Web Bulletin Board (WBB) and Verifiable Receipts

A final extension to the base model is a WBB and verifiable receipts. This is to enable the analysis of schemes which allow verifiability without compromising voter privacy, such as Prêt à Voter. In such schemes the voter receives a receipt, which may bear an encrypted value, e.g. the voter's selection. Ideally, there should be mechanisms that allow the voter to check that her vote has been encrypted correctly. She later checks her receipt against a WBB to ensure that it has been correctly recorded by the system. However, the ThreeBallots scheme enables verifiability without the use of encryption.

To achieve this, ballot forms are constructed in such a way that the portion retained by the voter as a receipt cannot be used as proof of a vote. Details can be found in [22].

In schemes which utilise encryption, the encrypted votes are typically passed through anonymising tabulation servers before final tallying. The final count is posted to a WBB, so can be verified by anyone. Further details can be found in e.g. [8, 7, 5, 9]. Note that the model can be used for schemes which do not use cryptography, as the possible threats can be evaluated as appropriate. This is illustrated in a forthcoming paper in which we use our model to analyse potential threats in Prêt à Voter and the ThreeBallots scheme: the former uses encryption, whereas the latter does not. The unifying requirement is that it should not be possible to link the voter's receipt to her (unencrypted) vote. However, with the possibility of verifying a receipt, coercion becomes a serious potential threat. This is identified in our model, and discussed in Section 3.

The WBB and verifiable receipt are defined below. Possible threats arising from these components are given in Figures 7 and 8.

> **Verifiable receipt** - Enables the voter to check that her vote has been correctly recorded by the system, without compromising voter privacy. There should be proof of authenticity, such as a verifiable digital signature, so that neither the system nor the voter can falsely claim that the receipt is invalid. It should not be possible for the voter to prove her vote using the receipt.

> **WBB** - This should be a publicly-accessible, write-only medium. The voter should be allowed access to verify that her receipt has been correctly recorded by the system. In addition, anyone should be able to verify that the intermediate decryptions of encrypted votes and/or the final tally is correct from postings to the WBB.

This completes the model, and in the next section, we discuss ways in which it may be used. Note that the model does not include certain threats such as forced abstention due to, e.g. shortage of election equipment, complicated voter registration, etc., as these are generally due to forces outside the system, and need to be addressed by means other than improvements in the protocol.

Threat	**Potential threat**
Voter identifiable from receipt	
Authority knowledge	Confidentiality
Receipt discarded/ surrendered	Integrity
Invalid signature	
Faked receipt	

Figure 7: **Verifiable receipt**

Threat	**Potential threats**
Monitoring access to the WBB	Confidentiality
Voter presented with fake WBB	Integrity
WBB modified	
Denial of service	Availability

Figure 8: **WBB**

3 Applying the Model

In this section, we describe the way in which our model could be used to identify potential threats in one of the more robust versions of the "Scratch Card" voting system [21]. This is a version of Prêt à Voter [8], which aims to promote voter understandability. The scheme offers receipt-freeness and limited voter verifiability without the use of encryption. It provides a good exemplar for an analysis as all the various components in the model can be demonstrated to their full extent.

3.1 Threat Analysis of the Randell-Ryan "Scratch Card" Voting System

An overview of the scheme is as follows. The voter randomly chooses a ballot form, an example of which is shown in Figure 9. A randomised candidate list is printed in the left hand column (LHC). Below this is a code identification number (CIN): the key to the candidate ordering. The same CIN appears at the foot of the right hand column (RHC), but is concealed with a scratch strip. Overprinted is the receipt identification number (RIN).

King	
Queen	
Knight	
Rook	
513170 (CIN)	023169 (RIN)

Figure 9: **Scratch Card ballot form**

X
023169 (RIN)

Figure 10: **Photocopied receipt**

In the privacy of the booth, the voter marks an "X" against her chosen candidate in the RHC. The LHC is detached and dropped into a clearly-marked LHC ballot box. Outside the booth, and in the presence of an official, a photocopy is made of the RHC, while the original goes into a clearly-marked RHC ballot box. The voter retains the photocopy as a receipt (see Figure 10), and can use it to check that her "encrypted" vote has been correctly recorded by the system. For example, the RIN and position of the "X" could be shown on a publicly-accessible Web bulletin board (WBB).

X
513170 (CIN)

Figure 11: **Countable vote**

At the close of voting, the scratch strips are removed from each RHC, revealing the CIN as shown in Figure 11. The votes can then be recovered by matching the LHCs to the corresponding RHCs. Note that the only purpose of the CIN, is to link the LHC to the RHC during the tallying phase.

Ballot auditing is carried out under the supervision of officials pre-, post- and during the election period. Voters and independent auditors take random ballot forms, scratch off the RINs and check that the CINs match the corresponding candidate order. Although the scheme boasts simplicity, the unwieldy tabulation process is a disadvantage. In addition, voters must rely on the correctness of the tabulation as the scheme does not provide verifiability of the final tally. See [21] for a discussion.

We now carry out a threat analysis of the scheme, first identifying the main components from the model: the *ballot form, voting booth, ballot storage, voting device, verifiable receipt, WBB* and *election results*.

3.1.1 Ballot form

Threats to confidentiality:

- Identifiable information added - Only the RIN and the voter's mark is recorded, so unless the correspondence between CIN and candidate order is leaked by the authority or the CIN-RIN noted by an official (see below), the RHC cannot later be identified at the WBB.

- Voter identifiable from ballot form - A potential threat, if an authority notes down the CIN-RIN correspondence from the RHC and the CIN-candidate order correspondence from the LHC, a voter would be able to prove her vote to that election official. A suggested mitigation is to have independent authorities for the LHCs and RHCs, the above attack would then require the cooperation of two dishonest election officials.

- Authority knowledge is a potential threat, as information about CIN-RIN and CIN-candidate list pairings could be leaked during creation, storage and distribution of ballot forms. A possible countermeasure is to have the CINs in the LHCs covered by scratch strips, which would only be removed during the tabulation process. Note that this would protect against authority knowledge during storage and distribution of ballots, but not during creation of ballots. However, it is possible to distribute the creation of ballots, by first covering the LHC CINs with

scratch strips, and getting a different group of ballot clerks to print the candidate order on the LHC.

It is interesting to note that in identifying the voter from information on a legally marked ballot form, the attacker makes use of a subliminal channel. In contrast, if a voter is identified from e.g. marks added to the ballot form by a dishonest official, the information flow is via an illegal channel.

Threats to integrity:

- Voter's choice incorrectly represented - The voter's choice could be incorrectly represented if there are multiple ballots with identical CINs. A RHC could then be incorrectly linked to a LHC with a different candidate order.
- Ballot form spoiled - A possible threat if the LHC CIN does not match the RHC CIN. However, this should be caught during both random pre-auditing and auditing during the election.
- Ballot form faked - This could be done with knowledge of how CINs are formed, but the chance of a faked ballot being caught during auditing should act as a deterrent. Anti-counterfeiting devices would be another possible mitigation against this attack. Note that [21] does not describe formation of the CINs.

3.1.2 Voting booth

Threats to confidentiality

- Voter monitored - A possible threat, e.g. with a hidden camera in the booth.
- Voter records own choice - A voter could e.g. use a camera phone, to prove the correspondence between candidate list and RIN, and later prove how she voted by showing to her scanned receipt (RHC) at the WBB.

  Note that the above would be threats in almost any scheme, but should nevertheless be evaluated in an analysis.

Threats to integrity:

- Voter choice influenced - A possible threat, e.g. by a subliminal message in the booth.
- Voter smuggles out unmarked ballot form - Chain-voting is a potential threat. The coercer marks the ballot and can later check the RIN of that ballot against the WBB, to ensure that the voter has complied with his instructions.

3.1.3 Ballot box

Threats to integrity:

- Ballot stuffing - This could be carried out e.g. by corrupt officials
- Spoiling - A possible threat, e.g. ballot forms could be lost or substituted by a dishonest election official. Having a VVPAT mechanism [17] in place is a possible mitigation. However, note that the use of the WBB only ensures that the ballots enter the counting phase.

Both attacks would require a certain amount of coordination as the CINs and RINs on the faked/substituted LHCs and/or RHCs would have to be correctly matched. The suggested mitigation in [21] is for LHCs and RHCs to be handled by independent authorities.

3.1.4 Voting device

Threats to confidentiality:

- Identifiable information added by device - As the device only scans the receipt this is not a particular problem.
- Voter choice incorrectly/not recorded - This is a possible threat, but would be discovered if voters are diligent in checking their receipts on the WBB. Another countermeasure is to have a VVPAT mechanism in place.
- Voter's activity monitored - This could be carried out e.g. via wireless connection, but as long as the CIN-RIN pairings are not revealed until the time of counting, the voter's choice cannot be learned from the RHC scanned by the device.

Threats to integrity:

- Faulty authorisation - Since the device does not authorise anything, this is not a threat to the scheme.

Threats to availability:

- DoS - A possible threat, e.g. due to device failure. However, the voter does not face the possibility of losing a vote if unable to scan her receipt, as may be the case with some touchscreen voting machines.

3.1.5 Verifiable receipt

Threats to confidentiality:

- Voter identifiable from receipt - This is not a threat (assuming correct operation of the scheme) as the LHCs and RHCs are both cast at the time of voting, and the voter cannot prove correspondence between RIN and candidate order. However, this is a potential threat if a corrupt official notes the CIN-RIN and CIN-candidate list correspondences on the voter's ballot form.

  Randomisation attacks are also possible. With this, an attacker could require e.g. that the first candidate is marked, regardless of which candidate ordering is used. The level of threat is determined by the extent a voter can pick a ballot of her own choosing and the number of candidates in an election. In the case of few candidates, it might be easy for the voter to pick a ballot where she can vote as she wishes while satisfying the coercer. However, as Ben Adida points out in [5], a more complex randomisation attack is possible by forcing a voter to vote for a candidate on the ballot form that is determined by the ballot identifier (RIN). A randomisation attack may benefit the low key candidates as the votes would be spread evenly across the candidates.

- Authority Knowledge - Kleptographic attacks [11] are a possible threat, where e.g. a cryptographic operation on the RIN or CIN would give away information about the corresponding candidate list. Such an attack would obviously require a lot of searching, and would be dependent on how the RIN and CIN numbers are generated.

- Discarded receipts/surrendered receipts may indicate receipts that will not be checked and hence could be altered without detection. A possible countermeasure is to have a VVPAT-style mechanism in place.

Threats to integrity:

- Invalid signature - A possible threat if the mechanism for digitally signing receipts is malicious or fails. Likewise the mechanism for checking the signature on a receipt. The voter is then unable to prove an incorrectly recorded receipt.
- Fake receipt - A voter could falsely claim to be disenfranchised with a fake receipt. A suggested mitigation is to frank the receipts [21].

  For both the above, a possible countermeasure is to digitally sign the receipts and then have immediate checks on the signatures.

3.1.6 WBB

Threats to confidentiality:

- Monitoring access to the WBB - This is not a particular threat as without knowledge of the RIN-candidate list correspondence, the value of the voter's vote cannot be learned from postings to the WBB. However, see threat under WBB modified.

Threats to integrity:

- The voter could be presented with a fake WBB, e.g. in a spoofing attack, and be misled into believing her vote has been recorded correctly when in reality, it has been changed.
- WBB modified - There is a risk that the WBB could be modified after the voter has checked her receipt. The WBB is supposed to be a write-only medium, but this seems hard to achieve in practice. Apart from the challenges of implementing a write-only WBB, a practical issue is how handle detected errors. Voters can e.g. complain if they cannot find their receipt at the WBB or if the position of the voter's choice has been shifted. A write-only WBB would soon get quite disorganised if the old errors are kept, and additional columns with the corrected

postings are added. Note however, that an attacker set on altering the election results in the "Scratch Card" voting system would not actually need to modify the WBB, since the scheme does not, in any case, provide verifiability of the final tally.

For both the attacks, a VVPAT mechanism is a possible countermeasure.

Threats to availability:

- DoS - A possible threat, e.g. due to network overload, power failure, etc.

3.1.7 Election results

Threats to integrity:

- Early publishing - A potential threat. To mitigate this, vote counting at local stations, the final tally and publishing of results should be carefully synchronised.
- Absence of verifiability - As the voter is only able to check that her receipt has been correctly recorded on the WBB, this should be regarded as a potential threat.
- False/erroneous count - There is a danger that this could go undetected, as the scheme offers limited verifiability. Again, a VVPAT mechanism is a possible countermeasure.

From the analysis above, it is clear that having a VVPAT mechanism in place would counter many of the threats to the integrity of the scheme. We next investigate whether or not this would add any further threats.

3.1.8 Paper audit trail

Threats to confidentiality:

- Voter identifiable from receipt - See the "verifiable receipt" component above for a discussion of this potential threat.
- Voter's choice noted by official - This is not a threat assuming the scheme operates as intended. However, the above also applies.

Threats to integrity:

- Mismatch between voter's choice and paper copy - Not a threat, as in the "Scratch Card" scheme, two copies could be made of the RHC: one of which the voter retains as a receipt, the other to act as a paper back-up.

It appears that a VVPAT mechanism would not introduce any additional threats, at least threats that may not have been present before. However, it may magnify any existing threats to confidentiality.

The analysis shows that the main problems with the scheme are firstly, that it only offers partial verifiability as the voter is only able to verify that her receipt has been correctly recorded by the system. Secondly, the voter may be open to coercion if the CIN-RIN correspondence on her ballot form, together with her ID is noted by an official. While a possible countermeasure for the latter has been suggested, the former requires trust in the correctness of the tabulation process.

The analysis also demonstrates that the model offers a systematic way to carry out a threat analysis of voting systems, i.e., by identifying the main phases and components in a scheme, and evaluating potential threats in direct relation to each component during a run of the protocol, taking into account its particular design aspects. We have aimed for generality so that the model is adaptable, and found this to be the case in the analysis of the "Scratch Card" scheme. The appropriate components could be readily selected from the model, and the vulnerabilities evaluated against the threat categories provided.

It should be noted that while every effort has been made to ensure completeness of our model, given the open-endedness of systems it is difficult to guarantee that it captures all possible threats.

In the next section we discuss the results and mention some limitations of the work.

4 Discussion and Future Work

We have presented a model for the systematic analysis of threats to voting systems that can be applied to a wide range of different schemes. This is further demonstrated in a forthcoming paper in which we use the model

to analyse potential threats in Prêt à Voter [8] and the ThreeBallot voting scheme [22].

In anticipation of some of these threats, error detection mechanisms have been built into many current schemes, e.g. randomised partial checking [13] of the mix process to ensure correct decryption of votes without compromising voter privacy. While a systematic threat analysis is valuable for identifying the need for error detection mechanisms, it can also be useful for assessing the effectiveness of any that are existing within the scheme, especially when taking interactions between the various players and components into consideration.

A further step from an analysis such as the one performed above could involve assessing the likelihood of certain threats occurring. This goes beyond the model: not only assessing the potential threats but also the probability of their occurrence, and could involve a more complex and informed analysis of the scheme in relation to both the sociological and technical aspects of its environment. Estimating the security of a scheme would then require balancing the probability of the threats occurring against the effectiveness of any error detection mechanisms that may be in place. Bryans et al. discuss this issue in [6], and make a distinction between accidental and malicious error. Our model can be used for analysing potential threats through accident or malice, e.g. a user interface could be deliberately confusing, or confusing due to poor design.

Bryans et al. also mention the need to consider threats to the reputation of a voting system [6]. Interestingly, transparency in a voting scheme could work against it, e.g. a large number of reported errors in recorded votes could dissuade voters from using it, and cause it to be abandoned altogether. As previously mentioned, there has been a recent move towards increased transparency in voting systems as a way to provide verifiability and to reduce dependency on the "correctness" of the system. However, possible threats to the reputation of the system are worth careful consideration. Although we briefly touched on this issue in the previous section, our model does not directly analyse threats to reputation, as this lies outside the current (largely technical) scope. Once again, the analyst would need to merge the sociological and technical aspects of a scheme in assessing the strength of its reputation.

Another important point raised in [6] is the importance of error handling and recovery strategies, alongside error detection mechanisms. This is a currently neglected field in research on voting systems, and error handling

and recovery is lacking in many current voting schemes. This is a highly complex issue, involving decisions not only on the way in which recovery should be effected, but also when the appropriate mechanisms should be invoked. It is likely that decisions would have to be made as to when margins of error are regarded as insignificant, and when they become unacceptable. Patterns of error may have to be studied, e.g. in deciding whether a particular error is accidental or malicious. This may, in turn, affect decisions on how best to deal with the error.

It would be highly useful to have a systematic model not only for the threat analysis, but also for dealing with any errors or security breaches that may occur as a result of these threats. This could take the form of a model for the analysis of potential threats based on the components in a scheme, in conjunction with a series of "decision trees" offering possible ways to handle such threats should they occur. We envisage this as a possible extension of our model, and a subject of future work.

Acknowledgments

The authors would like to thank Jeff Yan for many helpful comments.

References

[1] Workshop on developing an analysis of threats to voting systems, 2005. http://vote.nist.gov/threats/index.html.

[2] The machinery of democracy: Protecting elections in an electronic world (full report), 2006. Brennan Centre for Justice, NYU School of Law, http://www.brennancenter.org

[3] Punchscan, 2006. http://www.punchscan.org.

[4] VoteHere, 2006. http://www.votehere.net/default.php.

[5] B. Adida. *Advances in Cryptographic Voting Systems*. PhD thesis, Massachusetts Institute of Technology, 2006.

[6] J. Bryans, B. Littlewood, P. Y. A. Ryan, and L. Strigini. E-voting: Design for dependability. In *Availability, Reliability and Security (ARES)*. IEEE, 2006.

[7] D. Chaum. Secret-ballot receipts: True voter-verifiable elections. *IEEE Security and Privacy*, 2(1):38–47, January-February 2004.

[8] D. Chaum, P. Y. A. Ryan, and S. Schneider. A practical, voter-verifiable election scheme. In *European Symposium on Research in Computer Security*, number 3679 in Lecture Notes in Computer Science. Springer-Verlag, 2005.

[9] G. Danezis. *Better Anonymous Communications*. PhD thesis, University of Cambridge, 2004.

[10] A. Fujioka, T. Okamoto, and K. Ohta. A practical secret voting scheme for large scale elections. In *Workshop on the Theory and Application of Cryptographic Techniques: Advances in Cryptology*, pages 244–251. ACM, 1992.

[11] M. Gogolewski, M. Klonowski, P. Kubiak, M. Kutylowski, A. Lauks, and F. Zagorski. Kleptographic attacks on e-voting schemes. In *Workshop on Electronic Voting and E-Government in the UK*, 2006.

[12] A. Gumbel. *Steal This Vote*. Thunder's Mouth Press, U.S.A, 2005.

[13] M. Jakobsson, A. Juels, and R. Rivest. Making mix nets robust for electronic voting by randomized partial checking. In *USENIX Security Symposium*, pages 339–353, 2002.

[14] C. Karlof, N. Sastry, and D. Wagner. Cryptographic voting protocols: A systems perspective. In *USENIX Security Symposium*, 2005.

[15] T. Kohno, A. Stubblefield, A. D. Rubin, and D. S. Wallach. Analysis of an electronic voting system. In *Symposium on Security and Privacy*. IEEE, 2004.

[16] S. Kremer and M. Ryan. Analysis of an electronic voting protocol in the applied pi-calculus. In *European Symposium on Programming*, number 3444 in Lecture Notes in Computer Science, pages 186–200. Springer-Verlag, 2005.

[17] R. Mercuri. A better ballot box? *IEEE Spectrum Online*, October 2002.

[18] A. Neff. A verifiable secret shuffle and its application to e-voting. In *Conference on Computer and Communications Security*, pages 116–125. ACM, 2001.

[19] A. Neff. Practical high certainty intent verification for encrypted votes, 2004. http://www.votehere.net/documentation/vhti.

[20] T. Peacock. *Guess My Vote: a Study of Opacity and Information Flow in Voting Systems*. PhD thesis, School of Computing Science, Newcastle University, 2006.

[21] B. Randell and P. Y. A. Ryan. Voting technologies and trust. *IEEE Security & Privacy*, October 2006.

[22] R. L. Rivest. The ThreeBallot voting system. Unpublished draft, http://theory.lcs.mit.edu/∼rivest/ Rivest-TheThreeBallotVotingSystem.pdf, 2006.

[23] A. Rubin. *Brave New Ballot: The Battle to Safeguard Democracy in the Age of Electronic Voting*. Morgan Road, 2006.

[24] P. Y. A. Ryan. Verified encrypted paper audit trails. Technical Report CS-TR-966, University of Newcastle upon Tyne, 2006.

[25] P. Y. A. Ryan and T. Peacock. Prêt à voter: a systems perspective. Technical Report CS-TR-929, University of Newcastle upon Tyne, 2005.

Paper II: A Case Study in System-Based Analysis: ThreeBallots and Prêt à Voter

A Case Study in System-Based Analysis: The ThreeBallot Voting System and Prêt à Voter

Thomas Tjøstheim, Thea Peacock and Peter Y. A. Ryan

Abstract

Threat analysis of voting systems is a field of increasing interest. While it is important to verify the system itself, it has been found that certain vulnerabilities only become apparent when taking a "system-based" view, i.e. considering interactions between the various components of a scheme. In this paper we apply a model for system-based analysis to carry out a systematic threat analysis of the ThreeBallot voting system and Prêt à Voter.

1 Introduction

There has been much progress in developing high assurance, verifiable voting systems. Ideally, there should be minimal reliance on the players, i.e. voters, election officials, etc., and technical components, such as the hardware and software behaving as intended. Notable examples are Prêt à Voter [8], Punchscan [3], and VoteHere [4], all of which aim to provide a high degree of transparency in the system. While cryptography is often used to enable verifiability without compromising voter privacy, Rivest has shown with the ThreeBallot voting system that this is not, in fact, an absolute necessity [18].

Recently, interest has grown in analysis techniques to ensure that voting systems meet election requirements, such as eligibility, coercion-resistance and accuracy. In [13] Karlof et al. carried out a system-based analysis of Chaum's visual crypto scheme [7] and Neff's scheme [16, 15], identifying potential threats such as subliminal channels and "social-engineering"-style attacks. In a similar analysis, Ryan et al. [20] showed that Prêt à Voter [8] is robust against many of the threats mentioned in [13], but identified further possible vulnerabilities such as chain-voting and authority knowledge.

By considering the interactions between the various components in the above analysis new threats were identified. Although highly useful, this type of analysis did not consider the interactions systematically and hence, may not have uncovered all the possible threats. A more formal analysis of a voting protocol [17, 14] on the other hand may be more systematic, but is limited to the technical core of the protocol. Another approach is to develop a "catalogue of threats" [1], aside from [2] there is little work to date in this direction.

In [22], we proposed a model for analysis of threats in voting systems that is essentially "system-based", but considerably more systematic than previous similar work [13, 20]. While [2] has a largely technical focus and concentrates on DRE systems, our model operates at a higher level of abstraction and is not scheme-specific. In Appendix A we give a brief introduction to the model.

The structure of the paper is as follows. In Section 2 and 3 we apply the model to carry out a system-based analysis of the ThreeBallot voting scheme and Prêt à Voter, respectively. Finally, in Section 4 we discuss the main results of the analysis.

2 Threat Analysis Case Study: The ThreeBallot Voting System

2.1 Outline of the ThreeBallot Voting System

We now present an outline of the ThreeBallot voting system, for details see [18]. ThreeBallots is a verifiable paper-based voting system. Contrary to other verifiable voting systems, such as Prêt à Voter and PunchScan no encryption is used to provide voter verifiability, allowing a new level of transparency. In the following we describe the setup of the election. A voter votes using a multi-ballot, which consists of three individual ballots separated by perforated lines. Note that three single ballots together also could act as a multi-ballot. Each ballot lists the candidate choices together with a "bubble" for each candidate, and a ballot ID printed at the bottom of the ballot, see Figure 1.

A multi-ballot must be marked according to specific rules defined by the ThreeBallot voting system. In the privacy of the voting booth the voter fills in exactly one bubble for each candidate, and fills in an extra bubble for the

Ballot		Ballot		Ballot	
Odin	◯	Odin	◯	Odin	◯
Thor	◯	Thor	◯	Thor	◯
Trym	◯	Trym	◯	Trym	◯
Loki	◯	Loki	◯	Loki	◯
30111979		15031983		45140852	

Figure 1: An example of a multi-ballot

candidate(s) she prefers. After the voter has marked her multi-ballot, it is passed through a *checker machine* that optically scans the multi-ballot and verifies that the three ballots together have been correctly filled out according to the ThreeBallot voting system's rules. There should be exactly one mark for each of the non-chosen candidates, while there should be exactly two marks for the chosen candidate(s), if properly formed, the checker device prints a "red stripe" at the bottom of each ballot. The checker machine also lets the voter randomly choose a copy of one of the three ballots, to retain as a receipt. The receipt can later be used to verify that this part of the voter's multi-ballot has been correctly registered at the public Web bulletin board (WBB). While the three ballots together prove the voter's choice, the receipt does not give away any information about how the voter voted. After the checker machine has verified the multi-ballot, the voter casts her three ballots to the ballot box in the presence of a voter official. Each ballot is then scanned and posted to the WBB. Tallying is straightforward, the number of marks for each candidate are added up, and the result for each candidate is obtained by subtracting the number of voters from each candidate total (since each voter adds a mark for the candidates she votes against). Anyone can verify the final tally from the WBB.

2.2 Threat Analysis of ThreeBallots

2.2.1 Ballot form

- Identifiable information added by voter/official: A voter could mark three ballots in an identifying way, to later prove the triplet of ballots used to cast the vote. In [18] no method is described with respect

to how the ballots are scanned to the WBB. If, for example only the ballot ID and the index values of the corresponding candidate marks are registered at the WBB, it would be very difficult for a voter to prove her vote by adding identifying marks to the ballots. On the other hand, only registering a representation of the original ballots may open up for more errors, although such errors should be detected e.g. by a helper organisation or the voters themselves with high probability.

- Voter identifiable from ballot form: The ThreeBallot voting system is vulnerable to the "Italian attack", i.e. an attack where the coercer makes the voter vote for a selection of candidates that most likely will be unique. The three ballots can then be identified later at the WBB. A prerequisite for the attack is that there is a sufficient number of candidates to choose from, such that unique candidate selections can be made.

  In [18] no method for separating the ballots is described. The patterns of tearing may reveal the three ballots forming a multi-ballot. However, it will be very time consuming for an election official to physically go through all the ballots to identify corresponding ballots.

  A voter could prove her vote by writing down, memorizing or taking photos of the ballot IDs of the ballots forming the valid triple. If in addition the voter makes the ballot IDs known to the coercer before the WBB phase, the coercer will be quite certain of how the voter voted. A countermeasure mentioned in [18] is to use ballot IDs that are hard to memorize. Another possible mitigation is the "Shamos checker", which prevents the voter from learning the ballot IDs of the two ballots not chosen as receipts. A brief description of the Shamos checker is as follows:

  - If the multi-ballot is valid, 3 random ballot IDs are generated, without being shown to the voter. The voter selects one of the ballots to retain as a receipt.
  - The ballots that are not selected are put into a holding bin, while the machine produces the ballot selected as the receipt.
  - A voter can then verify that her receipt is identical to the selected ballot. If this is not the case, the "I got a bad receipt" button could be pressed, and the ballots in the holding bin will be put

into a spoiled ballot bucket. Otherwise, the ballots in the holding bin will be put into the ballot box.

Although the voter is not able to remember the ballot IDs of the two other ballots, new threats may be introduced. The machine could learn the correspondences between ballot IDs and ballots or could choose ballot IDs such that they can be easily correlated later. Another possible weakness is protection of the spoiled ballot box. Spoiled ballots could for instance be substituted with the ballot the checker produces to the voter. The voter could also refuse to put the last ballot into the ballot box, in order to steal votes from the other candidates.

- Authority knowledge: Election officials may learn the correspondences between ballot IDs of the three ballots forming the multi-ballot during ballot construction. Printing of separate ballot forms is one possible countermeasure. Another possibility mentioned in [18] is to let the voters pick ballot IDs from a bucket of stickers. A dishonest election official could learn the correspondence of ballot IDs when ensuring that the voter casts exactly three ballots. This could be envisaged as a "social engineering" type attack, in which a voter who has not understood the importance of not revealing the ballot IDs, could be tricked into revealing them. Voter education is important to ensure the voters' understanding of the system.

- Voter's choice incorrectly represented: A dishonest voter or election official could try to add or remove marks, after the ballots have been verified by the checker machine. This may or may not be detected during verification of the final tally, depending on the extent of the manipulation. It would, for instance, be quite obvious if the total number of votes for a particular candidate was more than the number of voters, or if a candidate has a negative number of votes. However, the attack cannot be traced without violating voter privacy. Checksums calculated over the original marked ballot is mentioned as a possible countermeasure. On the other hand, this would require a more complex checker device, which may be more vulnerable to tampering.

- Ballot form spoiled: An election official could mark a ballot in order to invalidate it (if a checksum scheme is used), or physically destroy a ballot form. If one of the three ballots belonging to a voter is modified,

the voter will detect it with a probability of 1/3, provided she checks her receipt against the WBB. Thus, it would be difficult for a dishonest voting official to carry out any substantial ballot spoiling attacks without detection, given that voters are diligent in checking their receipts against the WBB. However, tracing the attacks to the dishonest election official may be difficult, so this could be a way of launching a DoS attack.

- Ballot form faked: An interesting point is the level of authentication provided by the "red stripe" printed by the checker machine to prove that the ballots have been correctly formed. Given a set of valid ballots, a dishonest voter/election official may be able to fake the stripe and cast illegally formed multi-ballots. In the ThreeBallots scheme the names of all the voters and all the ballots ($3n$ if there are n voters) are posted to the WBB. Therefore, ballot faking attacks may be detected, but not necessarily traced. Although this approach allows public scrutiny of who voted, it may make forced abstention attacks easier to carry out.

2.2.2 Voting booth

- Voter's activity monitored: As for all schemes that require a polling station, a camera in the booth is a threat. Shoulder surfing may be more difficult than for other schemes, since the representation of the voter's choices is more complex. The candidates the voter votes *against* are marked once, while the voter's candidate choice has exactly two marks, a quick glance may therefore not immediately reveal how the voter voted.

- Voter records own choice: The voter could use a camera phone to record her vote, for example.

- Voter's choice influenced: There could be a subliminal message in the booth to persuade the voter to vote for one of the candidates, for instance. Note that the above threats to the voting booth component would be present in almost any scheme, but should nevertheless be evaluated in an analysis.

- Voter smuggles out unmarked ballot form: A chain voting attack could be initiated if a voter smuggles out an unmarked ballot form. The

coercer can then confirm how the voter votes by checking the ballot IDs at the WBB. Note that if the ballot ID stickers approach is used, the stickers could also be smuggled out, so the coercer could control the voters' behaviour.

2.2.3 Ballot storage

- Ballot stuffing: It is assumed that a voter casts exactly three properly formed ballots. A voter violating this rule could, for instance, cast only two ballots where only the marks for the desired candidate are included and discard the last ballot which contains the mandatory marks for the other candidates. The scheme does not specify a method to ensure that the voter casts exactly three ballots. An enforcement of this rule should preserve voter privacy as well.

  A voter or dishonest election official may be able to add extra votes to a multi-ballot if the checker malfunctions. Another threat is a voter who verifies two multi-ballots through the checker, and combines these to one "badly" formed ballot. It should be very difficult to pass more than one ballot through the checker, e.g. with election officials closely observing the process, or authentication mechanisms implemented in the device. Two voters could, however, bypass this by colluding to cast one illegal vote. This could proceed in the following way: each voter gets a properly formed multi-ballot verified by the checker, the first voter smuggles out the multi-ballot, while the second voter combines the two legal ballots to form an illegal ballot. This will be more effective than casting two legitimate votes, as they can construct a vote that, for example, gives three marks for their candidate and none for the others. The net effect is therefore that they loose one mark for their desired candidate, but steal two marks from the other candidates. Figure 2 illustrates how two legally formed ballots could be combined to form an illegal multi-ballot; for purpose of illustration we have chosen some easily recognisable ballot IDs.

  A possible countermeasure, could involve the checker machine printing three equal images or identifiers on the back of the ballots (chosen randomly from a large set) and adding a perforated line above the images. The voter then proceeds to an election authority who verifies that the images are equal to each other and tears them off. The voter

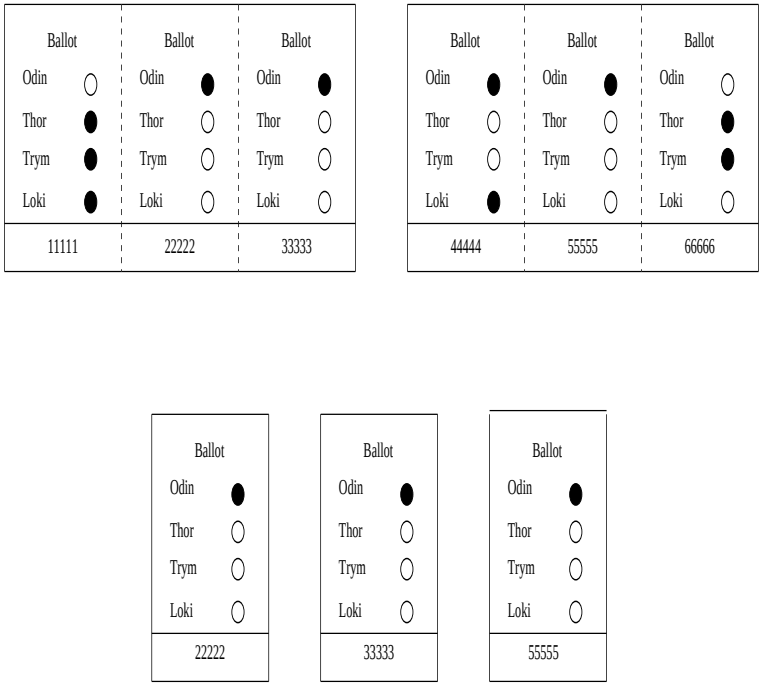

Figure 2: Creating an illegal multi-ballot

can then cast her ballots to the ballot box. This procedure would also ensure that the voter casts exactly three ballots. However, it may introduce more threats by adding an extra layer of complexity to the scheme.

- Ballot spoiling: The ballot box could e.g. be destroyed or replaced with one containing fake ballots, but voters will detect such attacks if they verify their receipts against the WBB. In [18] the use of "helper organisations" (e.g. the League of Women Voters) is envisaged as an additional help to verify WBB integrity.

2.2.4 Election results

- Early publishing: A threat to early publishing in ThreeBallots is the fact that by simply tallying the marks for each candidate on the voters' receipts one can get an indication of who is winning the election at that particular polling place. This threat was pointed out to us through per-

sonal correspondence with Roberto Samarone Araújo, Ricardo Felipe Custódio and Jeroen van de Graaf. A prerequisite for the attack is that voters are willing to show their receipts to some organisation that is awaiting people at the polling station. Given that voters mark their candidate choice in a random pattern and randomly choose one of the ballots to copy, to retain as a receipt, there is a risk that a statistical analysis will reveal which candidate is winning the race. This has also been confirmed through simulations carried out by Araújo et al.

- Absence of verifiability: Vulnerabilities covered under WBB and encrypted receipts.

- False/erroneous counts: This is not a significant threat due to the transparency of the scheme. All ballots are posted to the WBB, so that anyone can calculate and verify the final tally. This is a huge advantage compared to traditional paper based schemes and the verification is easier to understand for the average voter, than the mix net or homomorphic approach used in encrypted receipt schemes.

2.2.5 Voting device

- Identifiable information added: A checker could encode information about which ballots belong with each other, e.g. in the way the red lines are printed onto the ballots. Testing that the checker device is properly calibrated both before and during the election is a possible countermeasure.

- Voter's activity monitored: A malicious program inserted into the device could register information about the ballots and record sequence numbers. Another threat is a wireless component in the device which communicates to the outside, so the voter using the device could be linked with the information passed through the checker, i.e. a voter's choices.

- Faulty authorisation: Not a threat, as the voter's credentials are not authorised by the checker machine. However, ensuring that the voter only gets to check one multiple-ballot through the checker is a possible mitigation against ballot stuffing attacks.

- Voter's choice incorrectly/not recorded: A malicious checker could, for example add extra marks to the ballots on the chance that the voter does not notice. A malicious checker could also allow improperly formed ballot forms. However, this would require prior knowledge about the checker or the possibility that a voter could modify the checker machine.

- Denial of Service: A machine breakdown is not a threat to this scheme, as it is to DREs, where the votes are stored in the machines memory. There is less risk of losing votes, but voters should be prevented from voting until a new electronic device is found, to ensure that only properly formed ballot forms are cast.

2.2.6 Verifiable receipt

- Voter identifiable from receipt: Randomisation attacks are not a particular threat, since the voter can fill out the receipt according to the coercer's wishes, but still trick the coercer in the way the two corresponding ballots are filled out.

- Authority knowledge: Kleptographic attacks are not a particular threat, as the scheme does not make any use of cryptography. One possibility could be to encode information into the ballot ID number, e.g. a specific hash function computed over a ballot ID might reveal which ballots correspond to each other. Printing of single ballots, where the voter picks three single ballots randomly would counter this threat. An interesting vote buying attack is described in [6], in which the voter can effectively sell parts of her ability to verify her vote. The coercer pays or intimidates voters to construct receipts that only contain a mark for one candidate, e.g. Thor. If a sufficient number of people are coerced, the coercer will then have a higher probability of getting away with changing votes from say Odin to Thor, as these ballots do not constitute receipts.

- Receipt discarded/surrendered: May indicate receipts that will not be checked.

- Invalid signature: A possible threat if the checker does not sign the receipt properly, or the signature verifier does not work properly. The

voter would then be unable to convince the system that she has been disenfranchised.

- Faked receipt: The voter could claim to be disenfranchised by falsely claiming a faked receipt.

2.2.7 WBB

- Monitoring access to the WBB: A coercer with access to the the web server log at the public bulletin board web site could register which ballot receipt IDs are checked, and use this information to detect whether voters are trying to cheat by presenting fake triplets. The web interface used to access the WBB should therefore be implemented carefully to not reveal ballot IDs of receipts. Similarly, a dishonest employee of a helper organisation could sell receipts IDs. Information about which ballot IDs constitute receipts, together with the scheme's logics for filling out multi-ballots, could also be used to match ballots from the WBB in order to re-construct valid multi-ballots. In [23] several such successful attacks are described with various simulated election races.

- Voter presented with fake WBB: A false WBB could be shown to mislead a voter into believing that her receipt has been correctly recorded when in fact it has not.

- WBB modified: The WBB should be a write-only medium, but this seems hard to enforce in practice. There is a risk that the WBB could be modified after the voter has checked her receipt at the WBB.

- Denial of Service: DoS attacks are a possible threat.

3 Threat Analysis Case Study: Prêt à Voter'05

3.1 Outline of the Prêt à Voter Scheme'05

We now present a brief outline of the Prêt à Voter'05 scheme, for full details see [8]. Once registered in the polling station, voters select a ballot form, sealed in an envelope, at random. A typical example is shown below.

Odin	
Thor	
Trym	
Loki	
	$7rJ94K$

In the isolation of the booth, the voter makes her selection by placing a cross in the right-hand (RH) column against the candidate of choice. The left-hand (LH) column that carries the candidate list is discarded, leaving the ballot receipt. In this case, voting for Thor, the receipt would appear as follows:

X
$7rJ94K$

The voter then leaves the booth and casts her vote in the presence of an official: the receipt is placed under an optical reader, or similar device, to record the cryptographic value at the bottom of the strip, and the numerical representation of the cell into which the cross has been entered. The voter retains a digitally signed, hard copy of the right hand strip (RHS) as her receipt.

The candidate list on the ballot forms is randomised. Thus, with the left hand strip (LHS) removed and without knowledge of the appropriate cryptographic keys, the RHS does not indicate which way the vote was cast. A nice side-effect of using a randomised candidate list is that a random order does not favour any of the candidates, whereas a fixed candidate list may influence voters to vote for candidates that are listed early.

The cryptographic value printed on the bottom of the receipt, the "onion", is the key to extraction of the vote. Buried cryptographically in this value, is the seed information needed to reconstruct the candidate list. Thus, only a threshold subset of tellers holding the appropriate keys are able to reconstruct the candidate order and so interpret the vote value encoded on the receipt.

Once the election has closed, the receipts are transmitted to a central tabulation server which posts them to a secure Web bulletin board (WBB). This is an append-only, publicly visible facility. Only the tabulation server can write to this and, once written, anything posted to it will (in theory) remain unchanged. A voter can visit the WBB and confirm that her receipt appears correctly.

After a suitable period to allow voters to check their receipts, the tellers perform a robust, anonymising, decryption mix on the batch of posted receipts. The paper receipt allows voters to prove the absence or corruption of their receipt in the event that it fails to appear correctly on the WBB.

Various mechanisms are deployed to detect and deter any corruption in the construction of the ballot forms. The approach suggested in [8] is to perform a random pre-audit of the ballot forms.

3.2 Threat Analysis of Prêt à Voter'05

3.2.1 Ballot form

- Identifiable information added by voter/official: Not a threat as the LHS is detached and the RHS only states the onion and the voter's mark.

- Voter identifiable from ballot form: Only the numerical value of the voter's mark and the corresponding onion are recorded to the WBB, so unless the correspondence between onion and candidate order is leaked, the voter will not be able to prove how she voted. A voter could e.g. prove her vote by retaining the LHS. Possible mitigations are:

  - Enforcing destruction of the LHS in front of an official. However, the official may learn the correspondence between onion and candidate order and give away the information.
  - Mechanical destruction of the LHS, though this could be difficult to carry out in practice.
  - Having decoy LHS freely available in the booth. However, an adversary could mark decoy strips in a subtle way. A coercer may also be able to arrange that only "dummy" strips with a particular candidate ordering are available.

- Authority knowledge: Information could be leaked during storage and distribution, or later, once receipts have been posted to the WBB. Distributed generation of ballot forms has been proposed as a countermeasure in [21]. However, onion and candidate list correspondences could still be revealed by tellers acting in collusion.

- Voter's choice incorrectly represented: A possible threat if the onion is not a true encryption of the candidate order. Suggested mitigations are as follows:

  - Voter casts dummy votes. Given the RHS and associated onion, return the candidate she selected.
  - Return the seed and run a checking algorithm for well-formedness.

  The problem with the first method is that it is only a partial check of the ballot form construction. In addition tellers working in collusion, could return a fake candidate ordering. The second method is more thorough, but is only available to auditors. It is important to ensure that ballot forms are not re-used once they have been used for checking. Another possibility is to use an offline auditing mechanism, where audit information is posted on the ballot forms but concealed with a scratch strip [5].

  A better solution might be for the voter to verify the construction of the ballot she actually uses to vote. One possibility is the use of two-sided ballot forms, where the voter can verify correct construction of one of the sides, while using the other ballot side to vote. This adds a "cut-and-choose step": since the voter can check an arbitrary side, there is greater assurance that the other side is also correct [21].

  Another possible threat is invalid decryption of receipts. However, this will, with a high probability, be caught during randomised partial checking (RPC) [12].

- Ballot form spoiled: A ballot form could be spoiled by a dishonest election official, e.g. by adding additional marks to a ballot form or physically destroying a ballot form. Another threat is that the onion could be modified during scanning to the WBB. All of these attacks would be detected if voters check their receipts, but a dishonest election official could initiate a DoS attack in this way.

- Ballot form faked: Fake ballot forms can be constructed with knowledge of the tellers' public keys. A badly constructed fake ballot could also be used to initiate a DoS attack, as this would be caught during RPC of the mixing/decryption phase with high probability. However, ballot stuffing attacks should be difficult to carry out as the casting of ballots is supervised, so in principle, a voter is only able to cast one vote.

3.2.2 Voting booth

- Voter's activity monitored: The voter could be monitored by a hidden camera in the booth.
- Voter records own choice: A camera phone could be used to record the voter's choice.
- Voter choice influenced: There could be a subliminal message in the booth.
- Voter smuggles out unmarked ballot: Chain voting attacks are a threat as the coercer can control how the voter votes by checking the WBB for the corresponding onion. A countermeasure proposed in [20] is to cover the onion with a scratch strip and not reveal the onion before the tallying phase.

3.2.3 Ballot storage

This has not been specified in Prêt à Voter, but the device would presumably record receipts, e.g. by writing to a disk and transmitting immediately to the WBB.

- Ballot stuffing: Extra votes could be recorded by a faulty/malicious device.
- Ballot spoiling: Recorded data could be lost or corrupted. In addition, the disks could be substituted by a malicious party. However, if voters verify their receipts at the WBB, in combination with a VEPAT mechanism such errors will be detected.

3.2.4 Election results

- Early publishing: Tallying and publishing of final results to the WBB should be synchronised.
- Absence of verifiability: Not a threat, unless there is a DoS from the WBB.
- False/erroneous count: The risk of an erroneous count should be minimal as there are various mechanisms to verify decryption and tallying of votes.

3.2.5 Voting device

- Identifiable information added by device: Not a threat, as the device only scans the receipt.
- Voter's activity monitored: A possible threat (e.g. via wireless connection), but as long as the crypto primitives used in the ballot form construction remain secret, the voter's choice cannot be learned from the RHS scanned by the device.
- Faulty authorisation: Not a threat, as the device does not authorise the voter.
- Voter choice incorrectly/not recorded: A possible threat, but would be discovered if voters are diligent in checking their receipts on the WBB. The use of a VEPAT mechanism or helper organisations are countermeasures as well.
- Denial of Service: Device failure is a threat, but the voter does not face the possibility of losing her vote if unable to scan her receipt, as may be the case with some electronic schemes.

3.2.6 Verifiable receipt

- Voter identifiable from receipt: Randomisation attacks are a threat. An attacker could, for example, require that the first candidate is marked, regardless of which candidate ordering is used. The level of threat is determined by the extent a voter can pick a ballot of her own choosing

and the number of candidates in an election. In the case of few candidates, it might be easy for the voter to pick a ballot where she can vote as she wishes while satisfying the coercer. A randomisation attack may benefit the low key candidates as the votes will be spread evenly across the candidates.

- Authority knowledge: Kleptographic channels [11] are a threat, i.e. crypto variables chosen in such a way as to leak information to a colluding party. In Prêt à Voter'05, this is possible by choosing a seed value such that a keyed hash of the onion value reveals the candidate order. However, this would require a great deal of searching. In Prêt à Voter'06 [21] distributed creation of ballot forms is suggested as a possible countermeasure against kleptographic attacks.

  An important advantage with the Prêt à Voter scheme is that the voter does not need to communicate their choice to any device, and as such subliminal or semantic channels are not threats.

- Discarded receipts/surrendered receipts: May indicate receipts that will not be checked and hence could be altered without detection. A possible countermeasure is a verifiable encrypted paper audit trail (VEPAT) mechanism [19].

- Invalid signature: A possible threat if the mechanism for digitally signing receipts is malicious/fails; this also applies to the mechanism for checking the signature on the receipt. The voter is then unable to prove an incorrectly recorded receipt.

- Faked receipt: A voter could falsely claim to be disenfranchised with a fake receipt. This could be mitigated by using signatures as proof of authenticity of the receipts.

3.2.7 WBB

- Monitoring access to the WBB: There is a risk that the WBB could be modified after the voter has checked her receipt and prior to the randomising mix phase. Although specified as a write-only medium, this is difficult to enforce in practice. Fraud will be detected if voters verify their receipts more than once, but voters may be reluctant to do

so. As mentioned before a helper organisation in combination with a VEPAT may help ensure the integrity of the WBB.

- Voter presented with fake WBB: The voter could be presented with a fake WBB, e.g. in a spoofing attack, and be misled into believing her vote has been recorded correctly when in reality it has been changed.

- WBB modified: The integrity of the scheme is dependent on a certain percentage of voters verifying their receipts at the WBB. According to Carl Ellison [9]: "if there is a human step that is optional, then one can assume the human will not perform it. Some will and some won't, but for the purpose of security analysis, one must assume the worst case." A suggested mitigation is to have a VEPAT mechanism [19] in place, and for independent authorities to check the correspondence between the receipts and the contents of the WBB.

- Denial of service: A possible threat, e.g. due to network overload or power failure. DoS may also be an issue if a decryption mix net is used, e.g. if the tellers keys are corrupted/deleted. As discussed in [21], the advantage of a re-encryption mix is that faulty tellers can be removed if necessary.

4 Final Remarks

In the following we discuss the main results of applying the model defined in [22] to analyse the ThreeBallot voting system and Prêt à Voter.

4.1 The ThreeBallot Voting System

To ensure confidentiality in the ThreeBallots scheme it is important that once the multi-ballot has been split into three ballots in the voting booth, it should not be possible to link them at the WBB later. This is to avoid vote buying or coercion. However, as the analysis shows, it may be possible to link corresponding ballots in several ways, e.g. by remembering the ballot IDs, marking the candidate choices in a special way (the "Italian attack"), adding identifiable marks to the ballots, or by trying to reconstruct multi-ballots by matching ballots posted to the WBB. Another threat to confidentiality is registering which ballot IDs constitute receipts. This information can be

used to verify voter behaviour, e.g. making it harder for a voter to present a "fake" triplet of ballots to the coercer.

Threats to the integrity of the scheme include all threats that may violate the principle that the final count should accurately reflect the true intention of the voters. Examples include ballot modifications, ballot faking, or tampering with the election results. New marks could be added or deleted from a multi-ballot after it has been verified, using checksums is a possible countermeasure. Another threat is combining two verified multi-ballots to one "badly" formed multi-ballot. An important requirement to ensure integrity is that the voter should cast exactly three ballots.

Threats to availability include DoS attacks against the checker device, ballot spoiling attacks and denying access to the WBB. While the scheme advocates transparency, e.g. by posting the names of all who voted and all the ballots to the WBB, it may make it easier to launch DoS attacks. A dishonest election official could add ballots such that the number of ballots does not correspond with the number of voters. Another possible approach is to deliberately spoil receipts to get a number of voters to complain about incorrectly recorded receipts.

We identify some interesting trade-offs in the scheme. A countermeasure against the remembering of ballot IDs is to make them harder to remember, e.g. by using a bar code or mixing fixed noise with the ballot ID. Although, these approaches make it more difficult for the voter to remember the IDs of the ballots, it also makes the verification against the WBB harder. Another interesting trade-off relates to the simplicity of the checker machine. Ideally, the checker machine should be a stateless memoryless machine that only checks if a multi-ballot has been correctly formed or not. A more complex machine may thwart some of the attacks above, e.g. by authorising the voters or adding checksums to the ballots. On the other hand, a more complex checker is more vulnerable to tampering, since malicious software could be used to learn and communicate the ballot IDs of three ballots.

ThreeBallots achieves voter verifiability and unconditional privacy, i.e. privacy that relies neither on trusted third parties, nor on computational intractability assumptions (e.g. hardness of factoring). The scheme may not be practical as it stands, but is of immense theoretical importance as it demonstrates that it is possible to design a verifiable scheme with unconditional privacy without use of cryptography.

4.2 Prêt à Voter

In Prêt à Voter the confidentiality of the scheme is dependent on keeping the candidate order and onion correspondence secret. Threats to confidentiality include authority knowledge, chain voting, the voter retaining the LHS and kleptographic attacks. However, several mitigations have been suggested against most of these threats: distributed generation of ballot forms to counter authority knowledge and kleptographic attacks; scratch strips to mitigate chain voting attacks; and "dummy" LHS strips available in the booth to make it more difficult for the voter to prove how she voted. Prêt à Voter is vulnerable to randomisation attacks, but the impact on the election results may be limited, since the coercer cannot directly choose which candidate the voter votes for. However, low key candidates may benefit from the attack, if the votes are spread more evenly across the candidates.

Threats to the integrity of the scheme are largely related to the onion not corresponding to the candidate list. This means that the voters' choices are not correctly recorded. Voters can however verify the correctness of ballot forms by casting dummy votes, or using a two-sided ballot form approach, where the voter verifies a random side and uses the other side to vote. Other threats to integrity include fake WBBs and the fact that the WBB could be modified after the voter has verified her receipt.

The availability of the scheme may be violated through ballot spoiling or faking attacks, or deletion of mix administrators' keys. The latter threat is especially true for the "classic" version of Prêt à Voter which uses decryption mixes. A ballot spoiling attack could be a method to launch a DoS attack, since this would be discovered during randomised partial checking of the mix net.

Prêt à Voter is robust against most threats when considering the various countermeasures proposed for the scheme. A possible trade-off by introducing those countermeasures is a higher level of complexity that may weaken the voters' understanding of the system.

References

[1] Workshop on developing an analysis of threats to voting systems, 2005. http://vote.nist.gov/threats/index.html.

[2] The machinery of democracy: Protecting elections in an electronic world (full report), 2006. Brennan Centre for Justice, NYU School of Law, http://www.brennancenter.org

[3] Punchscan, 2006. http://www.punchscan.org.

[4] VoteHere, 2006. http://www.votehere.net/default.php.

[5] B. Adida and R.L. Rivest. Scratch & vote: Self-contained paper-based cryptographic voting. In *Workshop on Privacy in the Electronic Society*, 2006.

[6] A. W. Appel. How to defeat Rivest's threeballot voting system. http://www.cs.princeton.edu/∼appel/papers/DefeatingThreeBallot.pdf, 2006.

[7] D. Chaum. Secret-ballot receipts: True voter-verifiable elections. *IEEE Security and Privacy*, 2(1):38–47, January-February 2004.

[8] D. Chaum, P. Y. A. Ryan, and S. Schneider. A practical, voter-verifiable election scheme. In *European Symposium on Research in Computer Security*, number 3679 in Lecture Notes in Computer Science. Springer-Verlag, 2005.

[9] C. Ellison. Upnp security ceremonies design document, 2003.

[10] A. Fujioka, T. Okamoto, and K. Ohta. A practical secret voting scheme for large scale elections. In *Workshop on the Theory and Application of Cryptographic Techniques: Advances in Cryptology*, pages 244–251. ACM, 1992.

[11] M. Gogolewski, M. Klonowski, P. Kubiak, M. Kutylowski, A. Lauks, and F. Zagorski. Kleptographic attacks on e-election schemes with receipts. In *International Conference on Emerging Trends in Information and Communication Security*, Lecture Notes in Computer Science. Springer-Verlag, 2006.

[12] M. Jakobsson, A. Juels, and R. Rivest. Making mix nets robust for electronic voting by randomized partial checking. In *USENIX Security Symposium*, pages 339–353, 2002.

[13] C. Karlof, N. Sastry, and D. Wagner. Cryptographic voting protocols: A systems perspective. In *USENIX Security Symposium*, 2005.

[14] S. Kremer and M. Ryan. Analysis of an electronic voting protocol in the applied pi-calculus. In *European Symposium on Programming*, number 3444 in Lecture Notes in Computer Science, pages 186–200. Springer-Verlag, 2005.

[15] A. Neff. A verifiable secret shuffle and its application to e-voting. In *Conference on Computer and Communications Security*, pages 116–125. ACM, 2001.

[16] A. Neff. Practical high certainty intent verification for encrypted votes, 2004. http://www.votehere.net/documentation/vhti.

[17] T. Peacock. *Guess My Vote: a Study of Opacity and Information Flow in Voting Systems*. PhD thesis, School of Computing Science, Newcastle University, 2006.

[18] R. L. Rivest. The ThreeBallot voting system. Unpublished draft, http://theory.lcs.mit.edu/∼rivest/ Rivest-TheThreeBallotVotingSystem.pdf, 2006.

[19] P. Y. A. Ryan. Verified encrypted paper audit trails. Technical Report CS-TR-966, University of Newcastle upon Tyne, 2006.

[20] P. Y. A. Ryan and T. Peacock. Prêt à voter: a systems perspective. Technical Report CS-TR-929, University of Newcastle upon Tyne, 2005.

[21] P. Y. A. Ryan and S. A. Schneider. Prêt à voter with re-encryption mixes. In *ESORICS*, number 4189 in Lecture Notes in Computer Science, pages 313–326. Springer-Verlag, 2006.

[22] T. Tjøstheim, T. Peacock, and P. Y. A. Ryan. A model for system-based analysis of voting systems. In *Fifteenth International Workshop on Security Protocols*, 2007.

[23] C. E. M. Strauss. A critical review of the triple ballot voting system, part2: Cracking the triple ballot encryption. Unpublished draft, http://cems.browndogs.org/pub/voting/tripletrouble.pdf, 2006.

A Brief Overview of the Model

The model presented in [22] defines a set of components and associated threat categories to those components. The model was developed in a stepwise manner, where we first introduced a base model for a simple manual voting system, such as the one currently used in the U.K. We then extended the base model to include various features, such as a voting device, paper audit trail, verifiable receipts, etc. The threat categories to each component were determined by looking at the direct threats that could violate the purpose and requirements of a component. When deciding whether a threat category applies or not, it is important to consider the details of the particular scheme and how a threat may be manifested.

When using the model, the main components of a scheme such as the ballot form, voting booth, etc. are identified and the possible threats to each component at each phase of the protocol are considered in turn. In this way, it provides a guideline for evaluation of the system with the detail of a protocol-level analysis, but at the same time taking interactions between the various components directly into consideration. An advantage of the model apart from offering a more systematic approach to analysis, is that the components can be selected as appropriate and thus tailored to the scheme being analysed. In addition, by working through the threat categories in the model, and at the same time applying reasoning as appropriate to the scheme, the analyst is arguably better able to identify new threats than if using a catalogue of threats.

The model was designed to be as general as possible, so that it can be used for a range of different systems: from manual, paper-based voting, such as the current UK system, to more sophisticated systems that make use of, e.g. voting devices and verifiable receipts.

The possible threats associated with each component are given in Figures 3 - 10. Note that for all components in the model, the property violated is listed alongside each threat. Here, we consider the main properties required of secure systems, i.e., confidentiality, integrity and availability, rather than the traditional requirements of voting systems such as ballot secrecy, accuracy, verifiability, etc [14, 10].

Note that the model does not include certain threats such as forced abstention due to shortage of election equipment, complex registration, etc., as these are generally due to forces outside the system and need to be addressed by means other than improvements in the protocol.

Threat	**Property violated**
Identifiable information added by voter/official	Confidentiality
Voter identifiable from ballot form	
Authority knowledge	
Voter's choice incorrectly represented	Integrity
Ballot form spoiled	
Ballot form faked	

Figure 3: **Ballot form**

Threat	**Property violated**
Voter's activity monitored	Confidentiality
Voter records own choice	
Voter's choice influenced	Integrity
Voter smuggles out unmarked ballot form	

Figure 4: **Voting booth**

Threat	**Property violated**
Ballot stuffing	Integrity
Ballot spoiling	

Figure 5: **Ballot storage**

Threat	**Property violated**
Early publishing	Integrity
Absence of verifiability	
False/erroneous count	

Figure 6: **Election results**

Threat	**Property Violated**
Identifiable information added	Confidentiality
Voter's activity monitored	
Faulty authorisation	Integrity
Voter's choice incorrectly/not recorded	
Denial of service	Availability

Figure 7: **Voting device**

Threat	**Property violated**
Voter identifiable from receipt	Confidentiality
Voter's choice noted by official	
Mismatch between voter's choice and paper copy	Integrity

Figure 8: **Paper audit trail**

Threat	**Potential threat**
Voter identifiable from receipt	
Authority knowledge	Confidentiality
Receipt discarded/ surrendered	Integrity
Invalid signature	
Faked receipt	

Figure 9: **Verifiable receipt**

Threat	**Potential threats**
Monitoring access to the Web bulletin board (WBB)	Confidentiality
Voter presented with fake WBB	Integrity
WBB modified	
Denial of service	Availability

Figure 10: **WBB**

Paper III: Remote Electronic Voting Using Verifiable Chain Encryption

Remote Electronic Voting Using Verifiable Chain Encryption

Thomas Tjøstheim and Geir Røsland

Abstract

In this paper, we describe a new remote electronic voting scheme. A probabilistic multiple-key encryption function based on an extension of ElGamal constitutes the cryptographic basis for the scheme. Ballot encryption, mixing, and tallying are carried out through the construction of verifiable chain encryptions. Verifiability is based on the use of publicly available bulletin boards and scrutinizers representing the different candidates (or parties). The proposed scheme is receipt free and applicable for large scale elections.

Keywords: electronic voting, receipt freeness, scrutinizers, mix networks, coercion.

1 Introduction

Elections are the foundation of any democracy. However, quoting Tom Stoppard: "It's not the voting that's democracy, it's the counting". Throughout the history of voting there have been many examples of vote fraud [1, 2]. Initially, it may seem an easy task to design a fair election protocol: how difficult can it be to count the number of votes cast for each candidate? Further research, quickly reveals that the election process is more complex than first thought.

We propose a new remote electronic voting scheme for large scale elections. Over the last 4 decades, voter turnout has gradually decreased in established democracies [3]. Remote voting is convenient for the voters, and seems to be one effective way to increase voter turnout [4]. There is also a demand for better accuracy. In the 2000 USA presidential election, it is

estimated [5] that between 4 and 6 million votes were lost, mainly because of "unreadable ballots". Electronic elections have the potential to offer high accuracy, combined with verifiability and error recovery.

By utilizing a public network to transfer ballots from voters to election authorities, we introduce many potential security issues. Similar to financial transactions, we need to assure the confidentiality and integrity of the information passing through the network. Elections have the additional requirement of anonymity. The fact that the voters' choices must be kept secret, complicates verification and error recovery. Errors must be traced while preserving anonymity, and care must be taken not to provide the voters with verification information that could be exploited by vote buyers or coercers.

The absence of neutral third parties complicates the analysis of election protocols. Anyone, be it candidates, election officials, or the voters themselves could have the motivation to cheat. In this paper, we limit the security analysis to only consider aspects that are directly related to the remote electronic voting scenario. Obviously, threats that are similar for other networked applications, like worms, viruses and implementation bugs will need to be addressed before using the scheme in practice. Rubin [6] considers remote voting and the security of the hosts and the Internet itself.

The rest of the paper is organized as follows: Section 2 introduces the notion *receipt freeness* and shows how our scheme is receipt free, Section 3 describes the election protocol, Section 4 analyzes some important parts of the scheme, Section 5 and Section 6 describe the cryptographic building blocks that form the basis for our scheme, and Section 7 concludes the paper.

2 Receipt Freeness

Receipt freeness is a strong form of privacy, where the secrecy of the ballot is maintained even if a voter cooperates with an adversary, that is, a voter cannot prove her choice of vote to another party. A coercer or vote buyer is dependent on controlling voter behavior to confirm votes, and a receipt free scheme will make it harder to verify how a voter has voted.

The notion of receipt freeness first appeared in a paper by Benaloh and Tuinstra [7]. It was shown later by Hirt and Sako [8] that their scheme lacked the claimed receipt freeness property. An extensive summary of previously proposed receipt free schemes is given by Juels, Catalano and Jakobsson

[9]. All of the discussed schemes, except the one suggested by Juels et al. [9], assume the existence of an *untappable channel* between the voters and voting authorities. An untappable channel is a physical communication channel with perfect secrecy. Such an assumption makes the schemes unsuitable for Internet based voting, since the Internet is a public channel which cannot be untappable. The receipt free scheme by Juels et al. [9] defends against forced-abstention attacks and simulation attacks[1] even in the event of a collusion between a minority of the tallying authorities. A drawback with their scheme is that it is not suitable for large scale elections, as the overhead for tallying authorities is quadratic in the number of voters.

We propose an efficient and verifiable receipt free scheme for large scale elections. Obtaining universal verifiability and receipt freeness at the same time raises some problems, as these are somewhat conflicting properties. How can we assure voters of correct ballot postings without providing them with a receipt? A Public Bulletin Board (PBB) cannot be used, since voters would be able to show an adversary how their ballot was constructed, and then point to it on the PBB. We suggest using scrutinizers that represent different political candidates (or parties) to jointly monitor valid posting of ballots. The voters post their votes to a Restricted Bulletin Board (RBB) which only the RBB administrator and scrutinizers have read access to. A collaborative effort to cheat by the scrutinizers would be highly unlikely, given opposing political stands. The use of election observers to scrutinize the election is a trust model that is similar to what is used in traditional poll place elections. After the initial casting phase the ballots go through a mix net [10] to anonymize the voters' choices, before decrypting the ballots and tallying the votes. During the mixing and tallying stages of the election, PBBs are used to post verification information to enable universal verifiability.

An Internet-based receipt free scheme without any possibility of coercion is not practically realizable. Our goal has been to build an understandable, scalable, and secure scheme, that gains the voters' trust. Coercion and vote buying are theoretically possible if for example the coercer has continuous control over the voter or is collaborating with a dishonest election authority. However, the extent of coercion and vote buying will be strictly limited (discussed in Section 6.6).

[1] Attacks where the voter is forced to disclose her secret keys, enabling the coercer to vote on behalf of the voter (simulating to be the voter).

3 A New Remote Electronic Voting Protocol

3.1 Assumptions

A prerequisite for the scheme is the existence of a national public key infrastructure [11]. Many countries already have, or are developing such infrastructures [12]. Each voter goes through a registration procedure to obtain a signed election certificate. The certificate and corresponding private key are used to prove voter eligibility to the administrators of the scheme. The election administrators will similarly use digital certificates to authenticate themselves when communicating with the voters.

Re-voting can provide resistance to vote buying and coercion. However, it must be impossible for the coercer to detect whether the voter is trying to re-vote or not. Note that even if the communication between the voter and election officials is encrypted, the mere occurrence of traffic between the two parties could indicate an attempt of re-voting. We assume that the traffic generated from re-voting can be hidden from the attacker. This could for instance be done by periodically generating dummy traffic, that is, encrypted random messages sent from the voters [13].

3.2 Cryptographic Basis of the Protocol

The main cryptographic building block of the scheme is a new probabilistic multiple-key encryption function (explained in detail in Section 4) based on an extension of ElGamal [14]. We call the developed function the *election encryption function*, and briefly describe how it is used to create a chain of encryptions. Figure 1 shows the construction of an encryption chain when applying the election encryption function to ballot encryption, mixing and decryption of ballots. The voter first encrypts her ballot B and posts the encrypted ballot X to the RBB. The RBB administrator re-encrypts X and posts the result Y to the PBB. Each mixer re-encrypts the ballots at the PBB before the talliers re-encrypt the result Z, which returns the ballots to the initial states B, given correct operation of the election. The chain is *locked* in the sense that an encrypted ballot only can be decrypted if it has been encrypted in the correct sequence by the voter and all of the designated election administrators. The posting to the RBB and transition from RBB to PBB are only verified by the scrutinizers. While the rest of the steps in the encryption chain are universally verifiable, since verification information

is posted to the PBB.

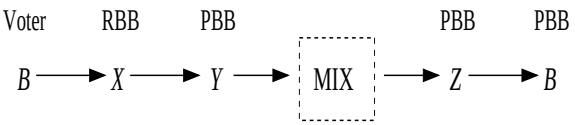

Figure 1: Building a locked encryption chain.

3.3 The Election Protocol

In the following, we first list the participants and (cryptographic) primitives of the election protocol (see Figure 2), before describing each step of the protocol. Note that the numbering given in the explanation of the protocol matches the numbers in Figure 2.

Participants

- S – Set of scrutinizers, $s \in S$.
- M – Set of mix administrators, $m \in M$.
- T – Set of talliers, $t \in T$.
- V – Set of voters, $v \in V$.
- A – RBB administrator.

Primitives

- Private channels
- Election encryption function
- Public hash function H
- PBB
- RBB
- Verifiable mix net
- Threshold secret sharing scheme

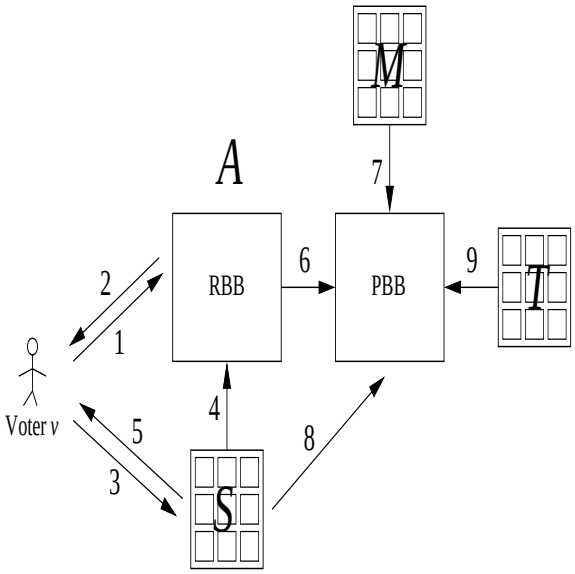

Figure 2: Protocol overview.

Protocol

1. Each voter v first encrypts her ballot using the election encryption function and then signs a hash of the encrypted ballot with her private key. Voters are authenticated by signing a challenge from the RBB administrator A.

2. Only if authentication of v is successful and if the voter signed hash verifies correctly, will A post v's encrypted ballot, the hash, and the signed hash to the RBB. An index value i of the posting is returned to v.

3. The voter v authenticates herself to the scrutinizers and states the hash of her encrypted ballot. The motivation for authenticating v, besides double checking that v is an eligible voter, is to avoid unnecessary investigation of requests made by ineligible voters.

4. If authentication of v succeeds, the scrutinizers check the RBB for the received hash.

5. Given that the scrutinizers find a matching hash on the RBB, the associated index value i of the posting is returned to v. Voter v checks

if this is the same index value returned from A and submits a complaint to the election officials if they are unequal (not shown in Figure 2).

If the hash is not found, the scrutinizers investigate the correspondence between v's posted signature and encrypted ballot. A could have modified the posted ballot or v could have asked for the wrong hash. An honest A can prove that the voter must have asked for the wrong hash by showing to the correspondence between the signed hash and the hash of the encrypted ballot posted to the RBB. Similarly, A cannot cheat without compromising the voter's private key.

During the initial voting phase, voters are able to update their cast votes. Steps 1–5 can be repeated any number of times, as the RBB only keeps record of each voter's most recently posted ballot.

6. The scrutinizers double check the RBB to make sure that A has not modified any values before transition to the PBB. If the check succeeds, A re-encrypts and permutes the ballots at the RBB by using his secret key and the election encryption function. The scrutinizers verify A's encryption and posting to the PBB.

7. A selection of mix administrators constitute the mix net. Each mix administrator uses the election encryption function and his secret key to permute the previously posted batch of ballots at the PBB, and posts the output batch back to the PBB.

8. For each mix, the scrutinizers verify that the ballots are not invalidated or substituted before the next mix can start. The PBB is used to post verification information to enable public verification of the mixing.

9. Given that all verifications hold, a certain number of the talliers jointly compute and publish the decryption key. All ballots can then be decrypted, and a tally can be published. The reason for using a threshold secret sharing scheme is to add robustness, as an attacker could prevent talliers from participating.

4 Election Encryption Function

The election encryption function is a probabilistic multiple-key encryption function for chain encryptions. It forms the basis for the initial ballot encryp-

tion, the mix net and the decryption of ballots by the talliers. Our proposal is based on an extension of ElGamal [14] and allows any number of keys. The main advantages with the proposed election encryption function is added robustness by enabling locking of the order of re-encryptions and an efficient verification algorithm to ensure correct operation of the mixers. We adapt the technique of randomized partial checking [17] to verify correctness of the mixes, but make an improvement by verifying the output without revealing single ballot correlations.

Select a prime p such that q is a large prime divisor of $p-1$. Select a generator β_0 of G_q, the subgroup of $\mathbb{Z}_p^* = \{1, 2, \ldots, p-1\}$, of order q, and n secret a_i's, such that

$$\begin{aligned} \beta_0^{a_1} &= \beta_1 \pmod p \\ \beta_1^{a_2} &= \beta_2 \pmod p \\ &\vdots \\ \beta_{n-1}^{a_n} &= \beta_n \pmod p \ . \end{aligned}$$

For each step in the encryption function, a random element $k_i \in \mathbb{Z}_q^*$ is chosen. A voter v encrypts her ballot B by

$$E_1(B, k_1) = (y_1, z_1) \ ,$$

where

$$y_1 = \beta_0^{k_1} \pmod p \quad \text{and} \quad z_1 = B\beta_n^{k_1} \pmod p \ .$$

The next $n-1$ encryptions are performed by consecutive mix servers, and are given as

$$E_i(y_{i-1}, z_{i-1}) = (y_i, z_i) \ ,$$

where

$$y_i = y_{i-1}^{a_{n-i+2}}\beta_0^{k_i} \pmod p \quad \text{and} \quad z_i = z_{i-1}\beta_{n-i+1}^{k_i} \pmod p \ .$$

The final encrypted pair (y_n, z_n) is given as

$$y_n = y_{n-1}^{a_2}\beta_0^{k_n} \pmod p \quad \text{and} \quad z_n = z_{n-1}\beta_1^{k_n} \pmod p \ .$$

Anyone who knows the secret key a_1, can obtain the original ballot B by performing the decryption

$$d_K(y_n, z_n) = z_n(y_n^{a_1})^{-1} = B \pmod{p} \ .$$

This works since

$$\begin{aligned} z_n(y_n^{a_1})^{-1} &= z_{n-1}\beta_1^{k_n}(y_{n-1}^{a_2}\beta_0^{k_n})^{-a_1} \\ &= z_{n-2}\beta_2^{k_{n-1}}\beta_1^{k_n}((y_{n-2}^{a_3}\beta_0^{k_{n-1}})^{a_2}\beta_0^{k_n})^{-a_1} \\ &= B\beta_n^{k_1}\cdots\beta_2^{k_{n-1}}\beta_1^{k_n}((\beta_0^{-k_1})^{a_1a_2\cdots a_n}\cdots(\beta_0^{-k_{n-1}})^{a_1a_2}(\beta_0^{-k_n})^{a_1}) \pmod{p} \ . \end{aligned}$$

Note that $\beta_i = \beta_0^{a_1a_2\cdots a_i}$, so

$$\begin{aligned} z_n(y_n^{a_1})^{-1} &= B\beta_n^{k_1}\cdots\beta_2^{k_{n-1}}\beta_1^{k_n}(\beta_n^{-k_1}\cdots\beta_2^{-k_{n-1}}\beta_1^{-k_n}) \\ &= B \pmod{p} \ . \end{aligned}$$

4.1 Locking the Order of Re-Encryptions

The encryption function described so far has a possible weakness, for $i \geq 2$ one can substitute an encrypted ballot with a new ballot B' and re-encrypt by selecting

$$y_i = \beta_0^{k_i} \quad \text{and} \quad z_i = B'\beta_{n-i+1}^{k_i} \pmod{p} \ .$$

Successive re-encryptions of all E_is of higher order are then required before the substituted ballot can be decrypted. This attack would probably be detected as each re-encryption is being verified by the scrutinizers. However, we want to avoid the possibility of an attack from a collaboration of dishonest administrators. A minor modification to the scheme enables us to lock the order of re-encryptions. We add the values (β_0^r, β_n^r), for a random $r \in \mathbb{Z}_q^*$, and require y_1 to be computed as

$$y_1 = \beta_0^r\beta_0^{k_1} \pmod{p} \ .$$

If we include the value β_n^r in the decryption

$$d_K(y_n, z_n) = z_n(y_n^{a_1})^{-1}\beta_n^r \pmod{p} \ ,$$

and this yields a valid ballot, we can be certain that the ballot the voter encrypted has been correctly re-encrypted by all of the key holders.

The public values of the election scheme are $(\beta_0, \beta_1, \cdots, \beta_n, \beta_0^r, \beta_n^r, p, q)$ while the private are $(a_1, a_2, \cdots, a_n, r)$, where each a_i is private to the corresponding administrator of the scheme.

5 Mix Network

The mix net secures the voters' privacy. Each mixer processes the ballots in batches, and shuffles the output according to a random permutation. The purpose of shuffling is to prevent an observer from linking the order of outputs to the order of inputs. A single mixer only knows the local permutation used, so the voters' anonymity is preserved as long as at least one mixer is honest.

Mixing is based on the use of a re-encryption mix net that uses the election encryption function defined in Section 4. A single mixer re-encrypts ballots in batches and permutes the ballots according to a random permutation π (or simply through sorting by decreasing/increasing values). The mixing phase starts by moving ballots from the RBB to the PBB. The mix net is universally verifiable, all information needed to verify the mixes are made available at the PBB. However, the first mix is verified by using the trust model of the scrutinizers. The voters cannot verify the first mix, since they then would be able to not only show a correspondence between their voting intentions and the encrypted ballot at the RBB, but guarantee for a vote being cast since re-voting no longer is possible after the transition to the PBB.

5.1 Mix Net Verification

How can we verify that a mixer performs correctly when re-encrypting ballots? A mixer could cheat by trying to substitute or invalidate ballots. Given

$$y_i = y_{i-1}^{a_{n-i+2}} \beta_0^{k_i} \pmod{p} \quad \text{and} \quad z_i = z_{i-1} \beta_{n-i+1}^{k_i} \pmod{p} \ ,$$

we want to verify that y_i and z_i were properly formed. We give the following protocol for the verification of a single (y_i, z_i) pair. The administrator reveals k_i. Note that this does not pose a security risk, as this only gives away the correspondence between z_i and z_{i-1}. An attacker still has to solve the discrete logarithm problem to find the administrator's secret key a_{n-i+2}. When k_i is known, one can verify z_i directly, but, we still need to verify y_i, without revealing a_{n-i+2}. We pick an $\alpha = y_{i-1} \beta_{n-i+1}^t$, for a secret random value $t \in \mathbb{Z}_q^*$, and ask the administrator to encrypt α, and receive $\sigma = \alpha^{a_{n-i+2}}$. If

y_i was rightly formed, then verification of y_i succeeds since

$$\begin{aligned}
\sigma\beta_0^{k_i}\beta_{n-i+2}^{-t} &= \alpha^{a_{n-i+2}}\beta_0^{k_i}\beta_{n-i+2}^{-t} \\
&= (y_{i-1}\beta_{n-i+1}^{t})^{a_{n-i+2}}\beta_0^{k_i}((\beta_{n-i+1})^{a_{n-i+2}})^{-t} \\
&= y_{i-1}^{a_{n-i+2}}\beta_0^{k_i}\beta_{n-i+1}^{a_{n-i+2}t}\beta_{n-i+1}^{-a_{n-i+2}t} \\
&= y_{i-1}^{a_{n-i+2}}\beta_0^{k_i} \\
&= y_i \pmod{p} \ .
\end{aligned}$$

However, instead of validating a single pair (y_i, z_i) we validate the product of y_is and z_is. Note that for all y_is and z_is of a batch we can write

$$\prod y_i = \prod(y_{i-1}^{a_{n-i+2}}\beta_0^{k_i}) \ = \ (\prod y_{i-1})^{a_{n-i+2}}\beta_0^{\sum k_i} \pmod{p} \ ,$$

and

$$\prod z_i = \prod(z_{i-1}\beta_{n-i+1}^{k_i}) \ = \ \prod(z_{i-1})\beta_{n-i+1}^{\sum k_i} \pmod{p} \ .$$

The mixer publishes $\sum k_i$, such that the encryption of z_is can be verified directly. The scrutinizers pick an $\alpha = (\prod y_{i-1})\beta_{n-i+1}^{t}$, for a random secret value t. The mixer returns $\sigma = \alpha^{a_{n-i+2}}$. Anybody interested can now verify that

$$\begin{aligned}
\sigma\beta_0^{\sum k_i}\beta_{n-i+2}^{-t} &= \alpha^{a_{n-i+2}}\beta_0^{\sum k_i}\beta_{n-i+2}^{-t} \\
&= (\prod(y_{i-1})\beta_{n-i+1}^{t})^{a_{n-i+2}}\beta_0^{\sum k_i}\beta_{n-i+1}^{-a_{n-i+2}t} \\
&= (\prod(y_{i-1}))^{a_{n-i+2}}\beta_0^{\sum k_i} \\
&= \prod(y_i) \pmod{p} \ .
\end{aligned}$$

However, a dishonest administrator can still cheat by

1. Replacing a set of y_{i-1}s or z_{i-1}s with a different set that gives the same product.
2. Calculating $\prod y_i$ and $\prod z_i$ with different k_is, while ensuring that the k_is sum to the same number.

Cheating attempts can be thwarted by using a technique called randomized partial checking [17]. The simple, but effective idea by Jakobsen, Juels and Rivest is to reveal half of the messages in all batches in order to verify mix correctness. Voter privacy is obtained by the mix-net as a whole, by always selecting ballots not compromised in the previous batch. We adapt this technique to our scheme, but verify the output without revealing single ballot correlations.

For each mix, the scrutinizers jointly and randomly pick half of the ballots in the input batch, and require that the mixer can show a set of corresponding ballots in the output batch. The scrutinizers and anyone else interested verify that the product of the chosen ballots in the input batch actually maps to the product of ballots in the output batch. The mixers are prevented from cheating, since they have to guess correctly which ballots the scrutinizers select, and then only change the y_is and z_is that are either all in, or all not in, the selected set. The chance of guessing this for more than just a few ballots is clearly very small.

6 Analysis

In the following, we study some selected parts of the election protocol that we believe are important to the scheme's security and usability. Due to lack of space, we only provide an initial analysis.

6.1 Properties of the Election Protocol

We briefly discuss the security properties [15] of the proposed scheme.

Legitimacy: only registered voters may vote, and only once. The scheme assures legitimacy, since each voter must register for a public-private key pair prior to the election. The private key is used to sign a challenge given by the RBB administrator A, who verifies the voters' signatures with the corresponding public key certificates. We assume that each voter has a dedicated area at the RBB that is overwritten each time the voter re-votes, thus protecting against double-voting.

Ballot secrecy: only the voter should know how she voted. The scheme assures ballot secrecy, since the voters encrypt their own votes. Any

device or other party cannot therefore learn the voters' choices. A mix net is utilized to anonymize each voter's ballot, since the scrutinizers and the RBB administrator A initially know the correspondence between encrypted ballots and voters.

Individual verifiability: the voter should be able to check that her vote is accurately recorded for tabulation. The scheme assures individual verifiability, since the voters can use the scrutinizers to unambiguously confirm their posting to the RBB. Each voter's signed hash prevents A from cheating, since a posting is only valid if there is a correspondence between the hash of the encrypted ballot and the signature.

Universal verifiability: the final tally should be verifiable by any third party. The scheme assures universal verifiability, since verification information is posted to the PBB, enabling public verification of the mix net and the decryption of the encrypted ballots.

Accuracy: the final tally should reflect the true count of all legitimate cast votes. The scheme assures accuracy, since chain encryptions are utilized, ensuring that a ballot only will be correctly decrypted if it has been processed in correct sequence, and by all the designated participants.

Receipt freeness: a voter cannot by herself prove that she voted in a certain way. The scheme assures receipt freeness, since the voters do not have read access to the RBB, and hence cannot show which vote that was encrypted. Before transition to the PBB the ballots at the RBB are re-encrypted. The purpose of the re-encryption is to cause a permutation of the ballots to prevent v from showing a receipt for her vote, as all participants have read access to the PBB.

6.2 The Role of the Scrutinizers

The scrutinizers play a critical role in the scheme. Their main goal is to increase the voters' trust in the election process by guaranteeing correct

operation of the election. However, this raises some practical challenges. We limit our analysis to study how the scrutinizers best can ensure the voters of correct ballot posting to the RBB. How the scrutinizers convince the voters, is essential for the voters' understanding and trust in the correct operation of the election. Ideally, all scrutinizers should be involved in the confirmation of the voters' ballots at the RBB. The voters can then be sure that their ballots are correctly posted, if at least one scrutinizer is honest. We do not think, however, that this is feasible in practice.

A more realizable solution is to let v pick a scrutinizer which she trusts to verify the ballot. A voter v can be quite certain of correct status of her ballot if the index value matches the value received from administrator A. The voter should also have the possibility of asking more than one scrutinizer, to double confirm, and to protect against a possible collusion between a dishonest scrutinizer and A. Each voter should be able to decide how many scrutinizers she needs to ask to be convinced of correct operation. Any irregularities in the answers from the scrutinizers should be reported to the election officials.

6.3 Error Handling

A detailed description of possible errors and recovery strategies must be carried out before the scheme is used. We only focus on how robust the scheme's construction is in terms of error handling. What if the index value returned to v by the scrutinizers is different from the value returned by the administrator A? Clearly, v will not be convinced of correct posting to the RBB. Bits could have been flipped while passing over the network or errors could have been introduced by a careless or cheating administrator. When will most errors occur and be discovered? Note that the scheme has two distinct phases. In the first phase, the voters post their ballots to the RBB and confirm correct ballot posting with help from the scrutinizers. The voters can also change their vote at any time during the first phase. At an official time announced in advance, the second phase starts by moving the ballots from the RBB to the PBB. The ballots go through the mix net, before being decrypted and tallied. A nice feature of this design, is that it is possible to catch errors before the second phase starts. In the event of errors, these can be found and fixed, without having to cancel the election. This is in contrast to many election schemes that verify correct procedure at the end of the election, making error recovery more troublesome. Note that significant

errors introduced in the mixing and tallying phase will be harder to recover from. Errors are, however, more probable in the first phase where more critical information is passed over the public network. In phase two, only information to and from the PBB will be exposed.

6.4 In Combination with a Traditional Election Model

Any remote electronic voting scheme should be integrable with the traditional election model, that is, voting at poll places. Many voters still lack the computer skills or the necessary equipment to participate in an online election. We utilize the fact that our scheme consists of two distinct phases to propose a solution for the combination of both election models. Let the voters cast ballots in the first phase until a pre-announced time. After the first phase ends, a day of traditional poll place election is held in-between phase one and two of the online election. A list of voters that voted in the traditional election is made and delivered to the scrutinizers. The scrutinizers jointly remove ballots at the RBB associated with voters who voted in both elections. The robustness of the election is strengthened through this combined solution. In the case of any serious errors, we have a solid backup solution.

6.5 Mixing in Practice

We mention some practical requirements to prevent information leakage during mixing. Disclosed information could be used for vote buying and coercion purposes. An important restriction is that the mix administrators do not use their own systems for mixing: it would then be easy for an administrator to store pairs of input and output values from the mix. We assume that special purpose mix machines are operated in protected areas, while being supervised by election officials. The calculations inside the machines should not be visible to anyone, but be based on parameters entered by the mix administrators. Correct operation of the mix machines can at any time be verified by running through test batches (for example an Elgamal encrypted input batch that is decrypted in the mix). It is inevitable that the operators of the mix will see some input and output values during work with the mixing and verification. However, the mixers cannot take with them any printouts or any other stored data when they leave the areas for mixing. In practice,

it will therefore be very difficult to participate in any form of vote buying or coercion.

Practical organization of electronic elections is a large challenge in itself, and we have only mentioned some rough ideas for the mix phase that especially should have strong security requirements. We conclude that practical security requirements are necessary, regardless of how well a protocol is constructed. There will always be additional reasons to protect the election, like for instance terrorist attacks.

6.6 Coercion

A common criticism against remote electronic voting schemes is the vulnerability to voter coercion. However, this is not a threat specific to Internet-based voting, traditional poll place elections are also exposed. Many poll place election are vulnerable to chain voting [16, p. 373], or the use of cameras to prove what a voter votes. Even so, remote electronic elections are potentially more sensitive, due to the many ways an adversary can verify voter behavior. The information being sent to the voters can be misused by an adversary to control the voters.

To get an idea of how vulnerable our scheme is to coercion, we consider three scenarios where a coercer threatens a voter. The voter (i) is forced to vote while the coercer watches, the voter (ii) is pressured into revealing her private key, and the coercer (iii) colludes with a dishonest scrutinizer to improve voter control.

(i) The first scenario represents little danger, since it is difficult to prevent a voter from re-voting at a later time. The coercer could stay with each voter till the deadline of the remote voting phase, but that would restrict the number of voters the coercer would be able to influence. In addition, in Section 6.4 we stressed that an electronic election should be combined with a traditional poll place election. A coerced voter in the scenario given above, can utilize the fact that a voter's ballot in the traditional election overwrites the vote given in the electronic election.

(ii) With her private key no longer secret, the voter will effectively have lost her vote. The voter could report that her private key has been compromised to the election officials, in order to obtain a new key pair. However, care must be taken to prevent a coercer from finding

out. The coercer could find out by trying to re-validate his vote posted to the RBB. A countermeasure could be to always return a positive acknowledgment if a verification request is made from an old key pair.

(iii) A coercer and scrutinizer collusion potentially gives access to all information on the RBB. The coercer could force voters to encrypt a ballot listing a specific candidate and verify that this value is posted at the RBB. This scenario is a real threat, but the probability of large scale coercion will be very small. Note that the dishonest scrutinizer could communicate with the coercer either during the ongoing election or after the election. Since the scrutinizers have conflicting political interests they will be controlling each other. We assume that it is nearly impossible for the dishonest scrutinizer to communicate during the election with the coercer, without being revealed by the other scrutinizers. Operation of a collusion after the election would be harder to detect. However,we assume that a scrutinizer will not be able to remember more than a couple of values on the RBB, as the scrutinizers are not allowed to carry with them any stored data when leaving the area of the RBB (similar to the security requirements discussed in Section 6.5).

7 Conclusions and Future Work

In this paper, we propose a new receipt free, remote electronic voting scheme for large scale elections. A new probabilistic multiple-key encryption function is used to carry out chain encryptions. An encryption chain is formed by the voter encrypting her ballot, followed by the mix administrators who each re-encrypt the encrypted ballot to the next step in the chain, while the talliers decrypt the last element in the chain. A ballot will only be correctly decrypted if it has been processed in correct sequence, and by all the designated participants. The integrity of the election is secured through the locked chain of encryptions which prevents insertion of valid ballots after the initial voting phase, where each step of the encryption chain is verifiable. Posting of ballots and RBB administrator A's re-encryption are only verified by the scrutinizers, to prevent a voter from showing how she has voted. The scrutinizers can be trusted to monitor and verify the election process, since they represent different candidates (or parties). Hence, a collaborative effort to cheat would be highly unlikely. Mixing of ballots at the PBB and the

tallying phase are universally verifiable.

Vote buying and voter coercion are theoretically possible, but minimized by several countermeasures: the scheme's receipt freeness property, the ability to re-vote in the remote election, the possibility of overwriting the electronic vote in a traditional poll place election, and practical security requirements. We note that the possibilities for vote buying and coercion could nearly be eliminated if the mix net was not publicly verifiable. Even though the trust model of the scrutinizers should be sufficient to verify the mix net, public scrutiny of the mix net is important to gain the voters' trust in the scheme. We believe that the integrity and usefulness of a remote electronic election is most important, and vote buying and coercion on a strictly limited scale can be tolerated, as long as the impact on the election outcome will be negligible.

We have only given an initial analysis of some selected parts of the scheme. Future work will include a more formal and in-depth analysis, in particular we want to consider:

1. **Error handling**: We need to carry out a threat analysis of different errors that could occur during the election phase and describe a detailed scheme for the handling and recovery of these errors. The probability of significant errors that make error recovery impossible should be minimized. One challenge is how to treat errors that occur during initial ballot encryption. These errors will not be detected before the tallying phase and cannot be mapped back to the voter. A possible solution is to let the voters cast dummy votes, to verify that encryption is carried out without errors, before the real vote is cast. Another scenario which should be studied further, is the voter as a possible adversary. A voter supporting a party with no chance of winning, could try to compromise the election process.

2. **Role of the scrutinizers**: The scrutinizers monitor and verify each step of the election protocol. We need to establish a clearer model for the verification of ballot postings and response to each voter. A method for double checking the RBB before transition to the PBB has not been specified. We cannot require from every voter that she states the hash of the posted ballot to the scrutinizers. A rough idea is to first check for the hashes received from the voters. For the cases in which no matching hashes are found, a database of voter certificates

could be utilized to verify the correspondence between posted hashes and signatures at the RBB.

3. **Vote buying and coercion**: A more systematic analysis of scenarios that could occur should be carried out. Note that the RBB administrator A potentially has access to all the correspondences between ballots at the RBB and PBB, while the scrutinizers have a 50 percent chance (given that they randomly pick half of the ballots when verifying A's re-encryption) of showing that a particular ballot at the RBB is not changed before the transition to the PBB. Further requirements to limit access to the RBB should be determined to prevent the scrutinizers and A from storing significant amounts of data from the RBB.

We do not think that the implementation of remote electronic elections is something that should be rushed. There are many uncertainties in the underlying network infrastructure that need to be analyzed and solved before a large scale remote electronic election should be deployed. Currently, there are initiatives to create an alternative to the Internet that is more robust and secure [18].

Acknowledgments

The authors would like to thank Håvard Raddum, Kjell Jørgen Hole, Lars-Helge Netland, Thea Peacock and the anonymous referees for many helpful comments.

References

[1] A. Gumbel, *Steal This Vote: Dirty Elections and the Rotten History of Democracy in America*, Nation Books Jul. 10 2005.

[2] T. Campbell, *Deliver the Vote: A History of Election Fraud, an American Political Tradition–1742-2004*, Carroll & Graf Publishers 2005.

[3] R. G. Niemi and H. F. Weisberg, "Controversies in Voting Behaviour," CQ Press, 2001.

[4] S. Parker, "Shaking voter apathy up with IT," The Guardian, 11 Dec. 2001.
http://society.guardian.co.uk/modlocalgov/story/0,7999,616636,00.html.

[5] Caltech-MIT Voting Technology Project, "Voting, What Is, What Could Be?," 2001.
http://www.vote.caltech.edu/Reports/2001report.html.

[6] A. Rubin, "Security Considerations for Remote Electronic Voting over the Internet," ;Login: The magazine of Usenix & SAGE, Feb., 2001, vol 26., pp. 20–28.
http://www.usenix.org/publications/login/2001-02/pdfs/rubin.pdf.

[7] J.C. Benaloh and D. Tuinstra, "Receipt-Free Secret-Ballot Elections (extended abstract)," in 26th ACM STOC, pp. 544–553, 1994.

[8] M. Hirt and K. Sako, "Efficient Receipt-Free Voting Based on Homomorphic Encryption," Proceedings of Advances in Cryptology – EUROCRYPT'00, pp. 539–556.

[9] A. Juels and D. Catalano and M. Jakobsson, "Coercion-Resistant Electronic Elections," 2005, ACM Workshop on Privacy in the Electronic Society.

[10] D. Chaum, "Untraceable Electronic Mail, Return Addresses, and Digital Pseudonyms," communications of the ACM, 1981.

[11] C. Adams and S. Lloyd, *Understanding PKI*, Addison-Wesley 2002.

[12] http://www.e.govt.nz/resources/research/international.html.

[13] R. E. Newman, I. S. Moskovitz and P. Syverson, "Metrics for Traffic Analysis Prevention," 2003, CHACS.

[14] T. Elgamal, "A Public Key Cryptosystem and A Signature Scheme Based On Discrete Logarithms," IEEE Transactions and Information Theory, vol 31, no. 4, pp. 469–472, Jul. 1985.

[15] P. Y. A. Ryan and T. Peacock, "Prêt à Voter: a Systems Perspective," Technical Report Series, CS-TR: 929, 2005.

[16] J. P. Harris, *Election Administration in the United States*, The Brooklings Institution, 1934.

[17] M. Jakobsson, A. Juels, and R. L. Rivest, "Making Mix Nets Robust For Electronic Voting By Randomized Partial Checking," 2002, proceedings of the 11th USENIX Security Symposium.

[18] D. Talbot, "The Internet is Broken," Technology Review, Dec. 19 2005. `http://www.technologyreview.com/InfoTech/wtr_16051,258,p1.html`.

Paper IV: Vulnerabilities in Online Banks

Vulnerabilities in Online Banks

Thomas Tjøstheim and Vebjørn Moen

Abstract

This paper describes some attacks on online banks that authenticate each customer through the use of a unique user identifier and a Personal Identification Number (PIN). Many user identifiers contain structure which make them easy to generate on a computer. Given a generated set of identifiers it is possible to do a brute-force attack on the PINs. A general attack model is described and some example attacks against a Scandinavian online bank are discussed.

Keywords: online bank, brute-force attack, DoS attack.

1 Introduction

Online banks have thrived with the explosive growth and availability of the Internet. A wide variety of services are offered to the customers. Paying a bill, checking the account balance, or applying for a loan can now be comfortably done from one's own home or office. However, the new possibilities introduced with Internet banking have also resulted in new security challenges. It is difficult to create both user friendly and secure Internet banking solutions. Can customers really trust that an attacker will not be able to break into their accounts?

Online banks claim that they are secure as they have many security features like firewalls, Public Key Infrastructure (PKI), Intrusion Detection Systems (IDSs), money auditing systems, and Secure Socket Layer (SSL). The average customer seems to be satisfied with the level of security in Internet banks. However, security is complex and not some magic potion that you add to your system to make it secure. Security should be considered from the start of the system development phase. A careful analysis of the environment is necessary to determine the needed security services and to determine

how to implement these services correctly. Analysis of a security protocol is very difficult, due to the many ways an attacker can take advantage of the protocol environment.

In this paper we show that online banks authenticating each customer through an N-digit PIN in combination with a structured user identifier are vulnerable to both brute force and Denial of Service (DoS) attacks. There are at least three online banks in Norway that use or have used this form of customer authentication. However, the authors have decided not to explicitly name any banks, as this has been requested from one of the banks, and the fact that some of the banks are still vulnerable to the attacks described in this paper.

The rest of this paper is organized as follows: Section 2 presents the general attack model, Section 3 describes structure and generation strategies when Social Security Numbers (SSNs) and account numbers are used as unique identifiers, Section 4 discusses some example attacks on a real Internet bank in Scandinavia, and Section 5 concludes the paper.

2 Attack model

A common way of authenticating customers in online banks is to require a unique identifier for each customer together with a secret that only the customer knows. This section describes a general attack model against online banks where each customer has a unique user ID and an N-digit PIN as their secret. The PIN can be either static or dynamic. A static PIN stays the same while a dynamic PIN is changed for each login; it can for example be generated by a PIN calculator.

A customer gains access to an account only by entering a valid user ID and PIN combination. Typically, there will be a limit on how many times (often three or five) a wrong PIN can be entered for a given customer's user ID. The objective of this limit is to prevent a brute-force attack against the customer's PIN. A customer will temporarily lose access to the account if the limit is exceeded and must contact the bank in order to receive a new PIN.

2.1 Brute-force attack

Figure 1 depicts the attack model for the brute-force attack. The generated set of user IDs will contain a subset of the user IDs belonging to the customers

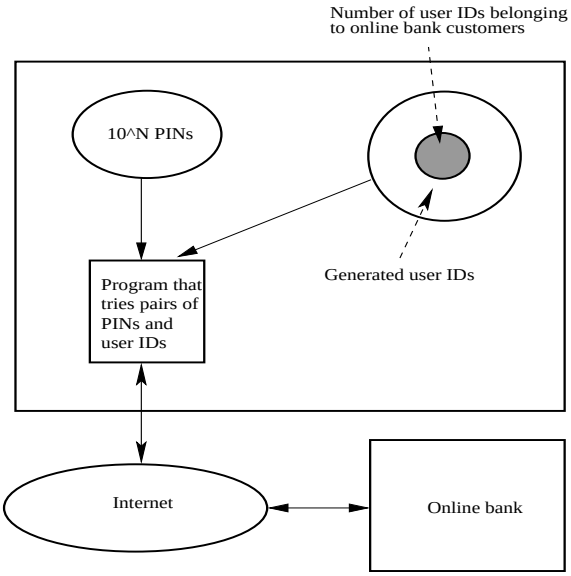

Figure 1: Attack model

of an online bank (we will see examples of how to generate such sets in Section 3). A program logs into the online bank's web pages and automatically enters different user ID and PIN pairs. One can observe that it does not matter if the PIN is dynamic or not. The only difference is that if a PIN is dynamic the attacker would have to attack the account at the moment a valid PIN and user ID combination is found.

Running the attack from only one host is not realistic, as this most likely will be detected by the bank's IDS. A distributed attack could be done by for example spreading a virus that contains the brute-force program in addition to having a control program that gets feedback from the zombie machines. Furthermore, it is possible to avoid the IDS by spreading the attack over several days in order to hide the number of tried logins in the anticipated natural traffic from legitimate users accessing the bank.

The probability of accessing at least one account is

$$\begin{aligned} P(\text{At least one}) &= 1 - P(\text{none}) \\ &= 1 - (1 - P_{success})^Y \end{aligned} \tag{1}$$

where Y is the number of online bank customers in the generated set of user

IDs. The anticipated number of cracks is

$$\mu = Y \times P_{success}. \tag{2}$$

The probability that an attacker cracks a random customer's account is

$$P_{success} = \frac{X}{10^N}. \tag{3}$$

Here, X is the maximum number of allowed wrong entries for a PIN, and N is the number of digits in the PIN.

Given a generated user ID, an attacker must first consider whether it is valid and belongs to an online bank user, or not. An attacker would ideally like to maximize the amount of customer user IDs in the initially generated set. This would give the most effective and *silent* attack.

2.2 DoS attack

If an attacker can acquire *many* user IDs, there is also a possibility of doing a distributed DoS attack against the bank's customers. The login scheme of online banks simplifies an application layer based DoS attack. An attacker can temporarily shut down access to accounts by entering X incorrect PINs for each valid user ID.

2.3 Combined DoS and brute-force attack

The probability in (3) assumes a combined DoS and brute-force attack. It can be discussed if the attacker is better off guessing $X - 1$ times since customers are not denied access this way, and it might take longer before the bank's IDS detects the attack. However, if the attacker controls a network of "zombie" machines, it is very difficult to both identify the attacker and stop the attack from all the machines in the controlled network. The chaos created by the combined attack could also be an advantage for the attacker. For more information on how to execute a distributed DoS attack please consult [1].

3 Generating user IDs

This section describes two real cases of user IDs being used in online banks. We will study structure and generation strategies for Norwegian SSNs and

account numbers. The arguments that apply to the Norwegian user IDs are similar for other countries. In particular, we have verified that the same generation strategies, with minor modifications, can be applied to both Swedish and Danish user IDs.

3.1 Norwegian SSNs

Norwegian SSNs are not confidential. Given a reasonable documented need, the SSNs can be requested from a national register. Many public institutions like hospitals, banks, and tax authorities have legal access to people's SSNs, but it can be difficult for private persons to argue the need for many SSNs. Therefore it might be better for an attacker to generate a set of SSNs. Most SSNs have a specific structure which make them easy to generate.

3.1.1 Structure of Norwegian SSNs

The Norwegian SSN [2] consists of 11 digits: $x_1x_2x_3x_4x_5x_6i_1i_2i_3c_1c_2$.

$x_1x_2x_3x_4x_5x_6$ is the birth date of the individual on the form *ddmmyy*.

$i_1i_2i_3$ is called the individual number and is used to separate people born on the same date. The national register distributes SSNs in the order they receive birth messages. They start with the highest available valid individual number for that day and proceeds downwards for each new birth message. The individual number is based on the century the person is born in, as shown in Table 1. It is also possible to distinguish boys from girls by looking at i_3, which is odd for boys and even for girls.

c_1c_2 are control digits that are calculated as weighted sums of the first 9 and 10 digits, respectively.

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

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

If either c_1 or c_2 is calculated to be 10 (mod 11) the SSN is discarded, and if c_1 or c_2 is equal to 11 then it is set equal to 0. Let's assume that c_1 and c_2 are approximately independent, then an SSN will be discarded with the following probability:[1]

$$\begin{aligned} p(c_1 \cup c_2) &= p(c_1) + p(c_2) - p(c_1 \cap c_2) \\ &= \frac{1}{11} + \frac{1}{11} - (\frac{1}{11})^2 = \frac{21}{121}. \end{aligned}$$

Individual number	**Year in birth date**	**Born**
500–749	>54	1855–1899
000–499		1900–1999
500–999	<55	2000–2054

Table 1: Correspondence between individual number and birth date

3.1.2 SSN generation strategies

How can an attacker maximize the ratio of customer SSNs in the initially generated set? Four different strategies will be discussed.

Strategy 1: The simplest strategy is to generate SSNs in such a way that all of the online bank's customers are covered. If for example the online bank only has customers in the age group 18–100, one could generate all possible SSNs for this group. With this scenario, all customers are born in the 20th century and are therefore given individual numbers in the range 000–499. Hence, for each day in the year we get 500 possible SSNs, but an estimate of 21/121 will be invalid numbers. The number of possible SSNs for people that are 18–100 is $500 \times 365 \times 83 \times (100/121) \approx 12.5$ million. Let Z denote the total number of customers in an online bank. The proportion of customer SSNs would then be

[1] To control the assumption of stochastic independence, a computer program was written that generated all the possible SSNs for the 20th century and counted the number of discarded SSNs. The results from the computer program gave the probability estimate 21/121 down to the 5th decimal.

$$R_{customers} = \frac{Z}{12.5 \text{ million}}. \tag{4}$$

The drawback, from the attacker's point of view, is of course the huge number of SSNs that has to be checked. A large portion of the SSNs will neither belong to *real* people nor to online bank users. There is also a high probability for the bank's IDS detecting the attack because of the big workload.

Strategy 2: A strategy for increasing the concentration of customer SSNs is to focus on a specific age group that has a high percentage of online bank customers. For example, 34 % of customers in pure online banks are males in the age group 26–35 [3]. The number of generated SSNs for this particular group would be $250 \times 365 \times 10 \times (100/121) \approx 754,132$. The ratio of customer SSNs would then be

$$R_{customers} = \frac{Z \times 0.34}{754,132}. \tag{5}$$

Strategy 3: However, one still has to generate a lot of SSNs belonging to non-existing people. This problem can be avoided by taking advantage of the chronological assignment of SSNs to newborn and immigrants. Instead of generating all valid SSNs for one day, it is possible to use population statistics to reduce the amount of generated SSNs. As an example, one can look at the period corresponding to males in the age group 26–35. There is an average of about 33,343 SSNs assigned per year for this group [4]. This gives an average of $33,343/365 \approx 91.4$ people per day. Let S be the number of assigned SSNs for a particular day. This number will of course vary from day to day. To get an idea of how S varies one can make the simplifying, but only approximately true assumption, that each day in the year is equally probable for the assignment of an SSN. This gives a binomial probability distribution with $n = 33,343$ (the average for a year) and the probability $p = 1/365$ (ignoring leap years) that a person is assigned an SSN for a particular day. The standard deviation for a random variable V having a binomial distribution is

$$Sd(V) = \sqrt{np(1-p)}. \tag{6}$$

From (6) we have $Sd(S) \approx 9.5$. To get an estimate of how S varies for each day one can construct an interval with ± 3 standard deviations. The

probability that S lies in the interval is

$$P(91.4 - 3Sd(X) \leq S \leq 91.4 + 3Sd(X)) \approx P(62 \leq S \leq 120) = 0.9974,$$

since a a binomial distribution can be approximated with a normal distribution.

If the attacker generates 120 SSNs for each day, then:

$$P(120 \leq S \leq 120 + m)$$

is the probability that the attacker looses between 0 and m SSNs for that day. This probability is limited to ≤ 0.0013 since ± 3 standard deviations are used. The number of generated SSNs would be $120 \times 365 \times 10 = 438,000$. We can assume that almost all of the online bank's customers in the age group 26–35 are covered. An approximated ratio of customer SSNs is then:

$$R_{customers} = \frac{Z \times 0.34}{438,000}. \tag{7}$$

SSNs that do not belong to *real* persons can be minimized if the attacker for example generates 62 SSNs for each day. This way the attacker will loose some SSNs belonging to real people, but will with probability $P(S \leq 62 - m)$ which is ≤ 0.0013 generate between 0 and m too many SSNs for a particular day. The number of generated SSNs would then be $62 \times 365 \times 10 = 226,300$. This will correspond to approximately generating 0.68 (226,300/333,430) of the total number of SSNs for the age group 26–35. If we assume a uniform distribution of online customers among the assigned SSNs, an estimate of the customer ratio is then:

$$R_{customers} = \frac{Z \times 0.34 \times 0.68}{226,300}. \tag{8}$$

Strategy 4: Another possibility is to filter out the SSNs belonging to online bank customers by trying to exploit response information from the bank's web pages. Two filtering examples are shown in Section 4.2.

3.2 Norwegian account numbers

An account number is a unique identifier for a customer's account, and can be generated in much the same way as an SSN. This is not hard when the structure is known.

3.2.1 Structure of Norwegian account numbers

A Norwegian account number [5] consists of 11 digits: $b_1b_2b_3b_4a_1a_2a_3a_4a_5a_6c_1$

$b_1b_2b_3b_4$ indicates which bank the account belongs to. Each bank has a set of serial numbers that identify the particular bank.

a_1a_2 is the type of account, e.g. a salary account or a high interest account. There is no standard for which numbers have to be used, each bank can define its own system.

$a_3a_4a_5a_6$ are digits that uniquely identify a customer's account. When a new account is created the smallest available 4-digit number is chosen.

c_1 is a control digit that is calculated as a weighted sum of the first 10 digits: $c_1 = 11 - (5b_1 + 4b_2 + 3b_3 + 2b_4 + 7a_1 + 6a_2 + 5a_3 + 4a_4 + 3a_5 + 2a_6 \pmod{11})$

However, if c_1 is calculated to be 10 (mod 11), the account number is discarded.

3.2.2 Account number generation strategies

The strategies for generating account numbers are easier than for SSNs. An attacker can find out which serial and account type numbers a particular bank uses. Given one of the bank's serial numbers and an account type, an attacker can generate the next four digits $a_3a_4a_5a_6$ in such a way that it gives a valid account number. Note that there are only $10,000 \times 10/11 \approx 9,090$ valid combinations.

An attacker can also take advantage of the fact that the smallest available account number is always chosen. It is also likely that an attacker can filter out valid account numbers by guessing incorrectly X times for a given account number and observing the response. Given this, it is possible to generate an interval of account numbers that the attacker knows belongs to customers.

4 Example attacks on a real Internet bank

Let Bank B denote the Norwegian branch of a Scandinavian bank that specializes in online banking services. The security solution was changed in

2004. In this section we will look at some theoretical example attacks on the Norwegian bank B, both before and after the change of security solution.

4.1 Bank B before 2004

A customer in bank B is supposedly authenticated by having a valid SSN, a $N = 4$ digit PIN, and a *personal* certificate. A new certificate must be downloaded for each new host used to connect to the bank. Before 2004 a customer downloaded a new certificate by entering a valid PIN and SSN pair. With this scenario an attacker could try to brute force the PIN by attempting to download a new certificate. An attacker had ($X = 3$) tries at guessing the correct PIN.

4.2 Brute-force attack

How does the brute-force attack described in Section 2.1 apply to bank B? Given the first strategy in Section 3.1.2, all SSNs for people in the age group 18–100 are generated. It is realistic to assume that nearly all of B's customers are covered in this SSN generation. In Norway, bank B had more than $Y = 220,000$ customers in 2003. From (1), the following probability can then be obtained

$$P(\text{At least one crack}) = 1 - (1 - \frac{3}{10^4})^{220,000} \approx 1,$$

and from (2), the anticipated number of cracks are

$$\mu = 220,000 \times \frac{3}{10^4} = 66.$$

The ratio of customer SSNs is from (4) $220,000/12.5$ million ≈ 0.018.

The second strategy was to only generate SSNs belonging to males in the age group 26–35. The expected number of B's customers in this age group is $Y = 220,000 \times 0.34 = 74,800$. This gives the following probability:

$$P(\text{At least one crack}) = 1 - (1 - \frac{3}{10^4})^{74,800} \approx 1.$$

The anticipated number of cracks when checking all SSNs one time is

$$\mu = 74,800 \times \frac{3}{10^4} \approx 22.$$

Strategy 3						
Variation	**SSNs per day**	**Age**	**Total SSNs**	**≈ B SSNs**	**Ratio SSNs**	**Cracks**
1	148	18–100	4,483,660	220,000	0.049	66
2	84	18–100	2,544,780	160,180	0.063	48
3	120	26–35	438,000	74,800	0.171	22
4	62	26–35	226,300	50,864	0.225	15

Table 2: Overview of results for strategy 3, when the average number of SSNs ±3 standard deviations is generated each day for the two age groups

From (5), the ratio of customer SSNs is $74,800/754,132 \approx 0.099$.

The third strategy was to exploit the chronological ordering of SSNs combined with the use of population statistics. We apply the strategy to both the age group of 26–35 and 18–100. Table 2 shows some different results when the average number of SSNs ±3 standard deviations is generated per day for the two groups. In particular, the table lists the total number of SSNs generated and the approximated number of B's customers covered.

In Section 3.1.2 a binomial probability distribution was assumed, and the two cases of generating 120 SSNs and 62 SSNs each day for the group 26-35 were considered. In the first case one can expect to almost cover all of the 74,800 customers in B and obtain about the same probabilities as when we generated all the valid SSNs for the same age group. The ratio (7) of customer SSNs would be $74,800/438,000 \approx 0.171$. With 62 SSNs generated each day the following estimate of the ratio of B SSNs is obtained from (8) to be $50,864/226,300 \approx 0.225$.

The birth statistics [4] for men and women in the age group 18–100 show that there is a total of 3,495,131 people (SSNs) and this gives an average of $3{,}495{,}131/83 \times 365 \approx 115.4$ per day. The standard deviation is calculated from (6) to be ≈ 10.7.

From Table 2 we see that the anticipated number of accounts cracked is dependent on the number of SSNs generated. There are different attack variations depending on how the bank will react to the attack. For instance, the most effective attack would be to try and verify the 226,300 SSNs generated with variation 4 in Table 2. Depending on how B would react to the first attack, the same attack could be repeated with about 15 anticipated cracks each time. On the other hand, if the attacker only gets one chance at

verifying the SSNs and the number of SSNs does not matter, then variation 1 in Table 2 yields the best attack.

The fourth strategy was to try to filter out the SSNs belonging to online bank customers. Two different approaches were discovered for the case of bank B:

1. When a valid SSN is combined with a wrong PIN the following error message is returned: "You have entered the wrong SSN or PIN." After three incorrectly entered PINs, and only if this is an SSN belonging to a customer, will the bank respond that the customer has been denied access to the bank. This means that an attacker can guess three times for each SSN, and not only find valid PIN and SSN combinations, but also filter out which of the SSNs belong to B's customers.

2. It is also possible for an attacker to filter an SSN by trying to register a new customer. When registering, bank B only verifies the correspondence between the SSN and the name. Fake email, phone numbers and so on can be entered. Only when entering an SSN that belongs to a customer will a specific error report be sent: "There was en error with registration. Please contact..." Otherwise the person with this SSN will be registered as a new customer. A disadvantage is that an attacker will register a *large* amount of new customers and this will probably be detected. The advantage of this method compared to the first is that an attacker can filter the SSNs before executing the brute-force attack.

4.3 Bank B in 2004

The certificate downloading scheme in bank B was altered in 2004. In addition to entering a valid PIN and SSN combination, a customer must also enter a valid *one-time password* that is sent either to the customers' mobile phone or mailbox. If the password is sent as an SMS it has 15 minutes validity, while a password in the regular mail is valid for 14 days.

The brute-force attack described in Section 4.1 is still possible. An attacker that finds valid PIN and SSN combinations can decide how the one-time password shall be delivered. If the password is delivered by SMS, the attacker could try to *sniff* the password with an interceptor [6]. However, it is much easier and cheaper for the attacker to get the password from the mailbox. Given a valid SSN it was possible to find the matching name and

the address. This could for instance be done by using a Norwegian pension fund web site [7]. This site authenticates members using only SSNs. When a member logs in, the site displays the name and address corresponding to the SSN. If the attacker chooses password delivery by mail, he decides when the password shall be delivered, and will have a good indication of when to steal the mail.

5 Conclusions

This paper shows that online banks authenticating customers through the use of a PIN in combination with an SSN or an account number are vulnerable to both brute-force and DoS attacks at the application level. The degree of exposure to brute-force attacks depends on the number of digits in the PIN. Whether the PIN is static or is changed for each login makes no difference in this case.

In Section 4 it was shown that bank B is vulnerable to a brute-force attack as the PIN only has 4 digits. The PKI solution in bank B is of limited value, since a new certificate and corresponding private key can be downloaded given a valid PIN and SSN combination and the one-time password.

An easy countermeasure against the attacks described in this paper would be to use user IDs that do not contain any structure, so that they would be difficult to generate automatically. The reason the banks have not done this, might be that it simplifies customer handling to use SSNs or account numbers as unique customer identifiers. Another suggestion is a fully functional PKI solution that require customers to meet in person with the Registration Authorities (RAs) when opening an account. This would enable stronger user authetication and possibly solve many of the vulnerabilities in online banks.

Much of the security in online banks relies on IDSs and money auditing systems. The problem with this approach is that it deals more with detection than attack prevention. A combined brute-force and DoS attack is hard to protect against with the current security schemes. The online banks have little other protection than temporarily closing down service for customers. The potential damage to the bank's reputation and the loss in revenue could be substantial.

References

[1] Jelena Mirkovic, Sven Dietrich, David Dittrich, and Peter Reiher "Internet Denial of Service, Attack and Defense Mechanisms." Pearson Education 2005.

[2] Ernst S. Selmer "Personnummerering i Norge: Litt andvendt tallteori og psykologi". Nordisk matematisk tidsskrift 12, Oslo 1964 (in Norwegian).

[3] Monica Hjorth. "En kartlegging av nettb@nkkunders holdninger". Hovedfagsoppgave i Informasjonsvitenskap, Universitetet i Bergen, November 2002 (in Norwegian).

[4] See `http://statbank.ssb.no/statistikkbanken/` (in Norwegian). Last visited August 29th 2005.

[5] See `http://www.ecbs.org/Download/TR201/norway.pdf`. Last visited August 29th 2005.

[6] See `http://www.spylife.com/digital_interceptor.html`. Last visited August 29th 2005.

[7] See `http://www.pensjonskassa.no/elaan/Elaan` (in Norwegian). Last visited August 30th 2005.

Paper V: Case Study: Online Banking

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.

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

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

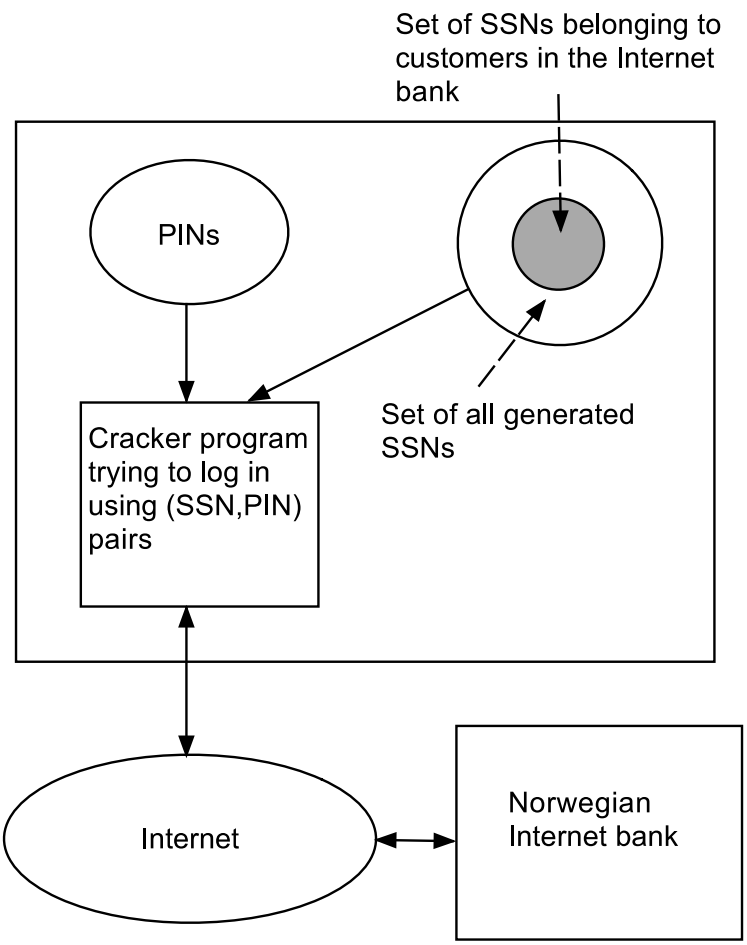

Figure 1: Model of a simple brute-force attack on a Norwegian Internet bank.

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 P(access at least one account) ≈ 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 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.

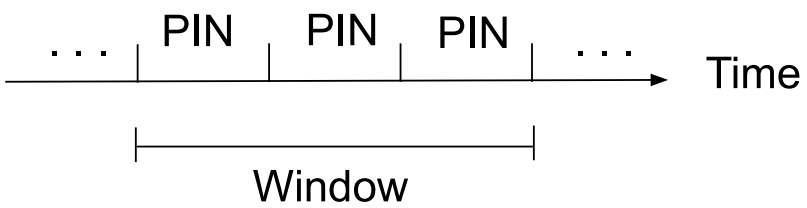

Figure 2: Window containing 3 PINs.

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

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

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 re-opened 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 be-

tween 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.

Paper VI: Next Generation Internet Banking in Norway

Next Generation Internet Banking in Norway

Kjell J. Hole* Thomas Tjøstheim Vebjørn Moen
Lars-Helge Netland Yngve Espelid
André N. Klingsheim

Abstract

The Norwegian banking industry has introduced a new security infrastructure for web applications, including Internet banking. The infrastructure, called BankID, has the potential to increase the security of today's web applications and facilitate new business opportunities. The authors consider BankID from the customers' point of view, analyze the risk the customers take when using BankID, and discuss how to mitigate the risk.

1 Introduction

Many countries, including Norway, Sweden, Denmark, Finland, Estonia, Austria, Belgium, and Canada, have introduced large-scale security frameworks implementing Public-Key Infrastructures (PKIs). In general, a PKI is a collection of hardware, software, processes, and people providing applications with security services based on public-key cryptography. The Norwegian banking industry has introduced a PKI called BankID. While BankID mostly authenticates Internet banking customers at the time of writing, the Norwegian banking industry wants BankID to become a national ID infrastructure used by government agencies and commercial companies to authenticate individuals and to provide legally binding digital signatures with a high degree of non-repudiation.

*Contact address: Professor K. J. Hole, Department of Informatics, University of Bergen, PB. 7800, N-5020 Bergen, Norway. E-mail: Kjell.Hole@ii.uib.no, Mobile: +47 920 38 164, Web: www.nowires.org.

Two earlier papers [1], [2] analyzed the Norwegian Internet banking and Automatic Teller Machine (ATM) systems. This paper applies elements of risk management [3] to BankID. Since the Norwegian banking community declined to share technical information about BankID, we were forced to evaluate BankID exclusively from the customers' point of view. The evaluation was completed in March of 2007 and was only based on publicly available descriptions of the BankID architecture and design, as well as personal use of the system.

In the remainder of the paper, we first determine how BankID differs from a typical X.509 PKI, before carrying out a risk analysis of the end-user authentication, non-repudiation service, and customer privacy. The risk to the customers is found to be significant. We therefore suggest steps to mitigate the risk.

2 PKI primer

This section introduces a typical X.509 PKI [4]–[7].

2.1 Keys, signatures, and certificates

In general, each end-user/customer in an X.509 PKI generates his own pair of asymmetric cryptographic keys—one *private key* and one *public key*. The public key is available to all end-users in the PKI, while the private key is only known to its owner. To secure the private key, it is stored in an encrypted file on the end-user's computer, or better, on a tamper-resistant smart card. As an example, all end-users can utilize Alice's public key to encrypt a message to her, but only she can decrypt the message because nobody else has access to her private key.

Bob can use his private key to *digitally sign* a message, or document. During the cryptographic signing procedure a value, denoted the *digital signature*, is calculated over the message. Alice verifies Bob's digital signature by applying a cryptographic procedure which takes Bob's public key, the received message, and the signature as input. If somebody tampered with the message after it was signed, then the verification will fail, else Bob must have signed the message since only he had access to the private key.

An X.509 *certificate* binds a public key to an identifier, e.g. a personal name, an assigned user number, or a web address. The identifier points to

the end-user or web site with the corresponding private key.

Commercial PKIs, including BankID, utilize different pairs of keys for digital signatures and encryption/decryption. For simplicity, we do not always differentiate between these pairs.

2.2 PKI architecture

An end-user requests a certificate from a *Registration Authority* (RA) by providing the information required to issue a certificate. The RA verifies the information and sends a certification request, containing the end-user's public key, to the *Certification Authority* (CA). The CA generates the certificate and signs it with its own private key. A web site owner initiates a similar procedure to obtain a web site certificate.

A CA revokes a certificate when the public key should no longer be used. Certificate revocation is frequently caused by the termination of a customer relation. More importantly, the CA must revoke a certificate immediately if the corresponding private key is compromised. The CA keeps track of revoked certificates. Often, a CA maintains a digitally signed *Certificate Revocation List* (CRL) containing unique references to the revoked certificates, the dates they were revoked, as well as the reasons for revocation.

A PKI may have multiple CAs. For simplicity, we consider a strict, two-level hierarchy of CAs containing a single root CA on the zeroth level with a special self-signed certificate, i.e., the signature is generated by the root CA's own private key. The root CA generates and signs certificates to the CAs on level one in the hierarchy. These level-one CAs again issue certificates to the end-users in the PKI.

2.3 Transitive trust model

An entity X *trusts* another entity Y when X assumes it knows exactly how Y behaves. As an example, an end-user trusts a CA when she assumes that the binding between the name and the public key in an issued certificate is correct. If X trusts Y and Y trusts Z, then X also trusts Z, e.g., an end-user trusting a root CA also trusts a level-one CA. If she doesn't trust the root CA she will not use the PKI services.

The root CA is the starting point of all trust in a PKI. In a two-level CA hierarchy, certificate path processing extends trust from the root CA to a level-one CA. Consider an instance where Bob and Alice are issued

certificates from different level-one CAs. To verify Alice's certificate, Bob first builds a path of certificates back to the root CA. The path consists of Alice's certificate, the certificate of the level-one CA that issued her certificate, and the root CA certificate. Using the root CA's public key, Bob then verifies the signature of the level-one CA certificate. Next, he extracts the public key from this CA certificate and uses it to verify the signature of Alice's certificate. Because Bob trusts the root CA, he now assumes that the content of Alice's certificate is correct.

2.4 Authentication

Authentication can be defined as the process of establishing an understood level of confidence that an identifier refers to an end-user or a web site. The authentication is said to be *strong* if the level of confidence is high.

Authentication in a PKI is based on certificates, the corresponding private keys, and a source of revocation information, e.g. a CRL. A simplified authentication protocol illustrates the authentication process. When Bob wants to authenticate Alice, he first asks for her certificate and then uses the CRL to verify that the certificate has not been revoked. He also confirms Alice's ownership of the public key by building and verifying a certificate path to a trusted root CA. Bob now asks Alice to sign a random number, denoted the challenge, with her private key. If Bob verifies the signed challenge using Alice's public key, then he assumes he is communicating with Alice. The same protocol is carried out when Alice authenticates Bob.

The strength of the authentication relies on a well tested authentication protocol, such as the Secure Sockets Layer (SSL) protocol, on how securely the private keys are stored, and on the computational difficulty of calculating the private key corresponding to a public key. To maintain the authentication strength over time, the CAs must update their CRLs often and the lists must always be available to both parties during the authentication process.

2.5 Legal view of non-repudiation

Non-repudiation offers a person protection against a false claim by another person that a communication never took place. The two most commonly discussed types of non-repudiation are *non-repudiation of origin* in which a person cannot falsely deny having originated a message, or document, and

non-repudiation of delivery in which a person cannot falsely deny having received a message.

From a legal point of view, non-repudiation consists of the ability to convince a third party that a specific message originated with, or was delivered to a certain person. Credible evidence is needed to persuade a judge, jury, or arbitrator. Both the quality and the presentation of the evidence determine the level of non-repudiation.

A non-repudiation service utilizing digital signatures can be built on top of a standard PKI. A high level of non-repudiation can be obtained if the basic PKI services are combined with the correct combination of legal and technical non-repudiation protocols. It is also essential that at least one trusted third party collects, validates, time stamps, signs, and stores relevant information [5, Ch. 9], [7, Ch. 4]. The third party must be able to withstand pressure from the communicating parties and present the non-repudiation evidence in an unbiased manner during a conflict.

It is important to note how the burden of proof is on the party wanting to rely on a digital signature. Hence, non-repudiation does not take away a person's legal right to refute a signature. A high level of non-repudiation only implies it will be possible to show, with high probability, that a person digitally signed a particular document even if he later denies it.

3 BankID explained

This section outlines the BankID architecture [8], [9] using the nomenclature established in the PKI primer.

3.1 Certificate types

There are three different types of 'user' certificates. The first type is a personal certificate for regular end-users, the second type is an employee certificate for end-users representing a company or an organization, and the third type is a certificate for web sites. All these certificates follow Version 3 of the X.509 Recommendation [4]. In addition, CAs, RAs, and other entities in the BankID infrastructure have their own certificates.

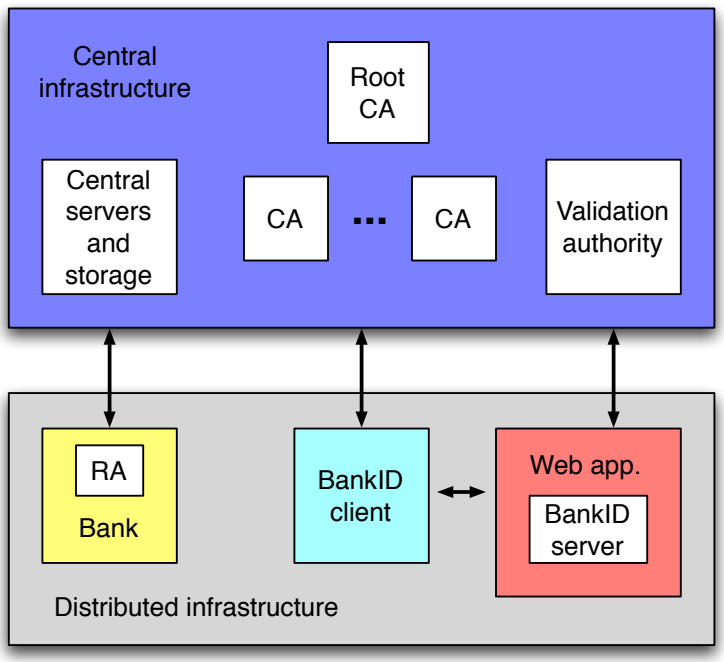

Figure 1: BankID architecture with lines indicating information flow.

3.2 Architectural overview

The BankID architecture is divided into two main parts as depicted in Figure 1. The first part, referred to as the *central infrastructure*, is operated by the Norwegian Banks' Payment and Clearing Centre (also known as BBS). The central infrastructure contains CAs, a certificate *Validation Authority* (VA), central storage facilities for cryptographic keys and certificates, and functionality for digital signing of documents. The *distributed infrastructure* comprises the BankID server, which is part of a web application, the RA at an end-user's bank, and the BankID client running on the end-user's computer.

3.3 Central infrastructure

The central infrastructure includes a two-level hierarchy with one root CA and multiple level-one CAs. The root CA is owned by the Norwegian Finan-

cial Services and Saving Banks Associations. While the level-one CAs are part of the central infrastructure, each level-one CA is owned by a separate bank (or group of banks). A bank uses its CA to issue certificates to its own customers. The root-CA certificate is valid for 26 years, while the level-one CA certificates are valid for 12 years [9].

The VA utilizes CRLs from the level-one CAs to determine if certificates have been revoked. This validation service is available to both the BankID server and client in Figure 1.

3.4 BankID server

A web application utilizing BankID, e.g. an online store or an Internet banking site, runs a BankID server. This server software is available in both the C and Java programming languages, and can be incorporated into the web application. The BankID server stores its certificate and private key in a *Hardware Security Module* (HSM) or an encrypted PKCS #12 file [10]. HSMs also provide web applications with dedicated hardware for processor-intensive cryptographic operations.

3.5 Bank RA

Each bank operates its own RA software, which is likely to be integrated into the bank's customer service software. A new BankID customer requests a certificate from the RA using his Internet browser or by showing up in person at the local branch office. The RA sends a certification request to the central BankID infrastructure using the SSL protocol.

3.6 BankID client

The certification request from the bank RA starts the initialization of a new customer record. The central infrastructure first generates (at least) one public-private key pair. The private key is stored in a secure central database and the public key, as well as the request from the RA, are sent to the CA belonging to the customer's bank. The new certificate generated by the CA is stored on the central infrastructure. The customer downloads a Java applet to her Internet browser each time she wants to use the BankID client. No software or information is stored permanently on her computer.

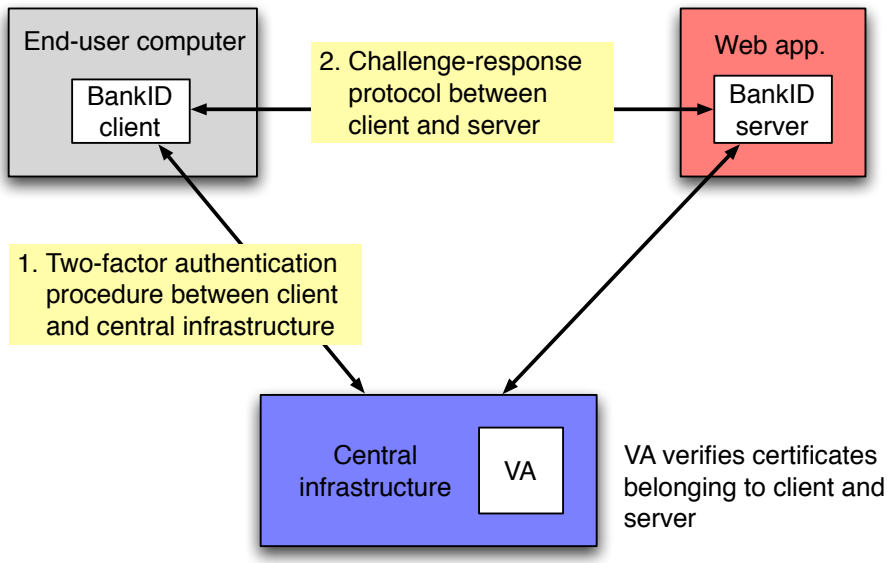

Figure 2: Overview of the BankID authentication procedure.

3.7 BankID authentication procedure

The *mutual* authentication between an end-user and a web application is divided into two parts as depicted in Figure 2. The first part, a two-factor authentication procedure (to be explained), authenticates the end-user to the central infrastructure and ensures that the user controls the access to her own centrally stored cryptographic keys. The second part is a challenge-response protocol between the BankID client and server. The protocol is similar to the one described earlier. All communications between the entities in Figure 2 are executed inside SSL tunnels.

The first part of the authentication operates as follows. Since an end-user can have multiple BankID certificates issued by CAs belonging to different banks, she first enters her Norwegian Social Security Number (SSN) into the BankID client. The central infrastructure responds with a list of all her BankID affiliations enabling her to choose the one she wants. Next, the client prompts the user for a one-time Personal Identification Number (PIN) which is verified by the central infrastructure. The one-time PIN is taken from a list of PINs supplied by the end-user's bank, or generated by a hardware token, often called a PIN calculator. A fixed PIN is sometimes needed to

activate the PIN calculator. Finally, the client prompts the user for a fixed password used by the central infrastructure to get access to the end-user's private key.

The described procedures are essentially traditional two-factor authentication mechanisms based on something you have (the PIN calculator or list of PINs) and something you know (the fixed password and perhaps a fixed PIN needed to activate a calculator) [6].

The second part consisting of the challenge-response protocol is completed once the central infrastructure has access to the end-user's cryptographic keys. During the protocol execution, the central infrastructure carries out the cryptographic functions for the BankID client. The BankID server uses the VA to verify the certificate from the BankID client and vice versa.

The reader should note that BankID differs from a typical X.509 PKI because the central storage of asymmetric keys and the central execution of cryptographic operations make it necessary for a BankID client to transmit a one-time PIN and a fixed password over the Internet to the central infrastructure.

3.8 Non-repudiation

BankID aims to provide both non-repudiation of origin and delivery. Unfortunately, nearly all information about the legal and technical non-repudiation protocols and storage of non-repudiation information are kept secret. It is known, however, that signed documents contain the signed data, the signatures of the parties, and the results of the VA requests at the time of signing [8]. BankID again differs from a typical X.509 PKI since no trusted third party is used to establish non-repudiation information for dispute resolution [8], [9].

4 Risk of authentication service

It is convenient to define *risk* as the possibility of suffering harm or loss. There are several types of risks [3]. On one hand we have pure risk associated with natural accidents such as fires, floods, hurricanes, and earth quakes. Speculative risk on the other hand is related to man-made failures, e.g., terrorism and misuse. We'll only consider speculative risk, denoted 'risk' for simplicity.

Let a vulnerability be a flaw in BankID and let a threat be an adversary with the capabilities and intentions to exploit a vulnerability. The risk taken by BankID customers is a function of the exploitable vulnerabilities and the danger from the threats [3]. This section discusses the risk associated with the BankID authentication procedure and suggests how to mitigate the authentication risk.

4.1 Exploiting the BankID authentication procedure

The BankID client gets access to the cryptographic functionality in the central infrastructure after the end-user has provided the correct SSN, one-time PIN, and fixed password. Experiments have shown that if the customer enters correct SSN and a wrong PIN three times, then the access to the central infrastructure closes down and the customer must contact his bank to reopen the account. (Access is also denied if the end-user first types in correct SSN and PIN and then enters a wrong password enough times.) The described process can be automated. In a simple proof of concept, an executable script starts Internet Explorer 7, downloads the BankID client, and simulates a user logging in with the correct SSN and wrong PIN three times.

Because SSNs have a well-defined structure it is possible to generate a large set of SSNs containing SSNs belonging to BankID customers. (The set may also contain SSNs not belonging to customers. See [1] for details.) A small program can then run through the set of SSNs and close down the customers' accounts as described above. The attack can be spread over many computers to construct a Distributed Denial-of-Service (DDoS) attack at the application layer, which is very difficult to stop if the number of computers is large ($\geq 10,000$) [1], [11]. Unlike DDoS attacks at lower network layers, it is not necessary to transmit a large amount of dummy traffic for a long time to close down the accounts, it is only necessary to try to log into each account a small number of times. Because this efficient DDoS attack on the authentication procedure has the potential to deny all users access to BankID, it can be said that the authentication procedure represents a *single point of failure*.

During the fourth quarter of 2006, about 600,000 of the 2.3 million Internet banking customer in Norway had moved to BankID. If most of the remaining customers also move to BankID during 2007, as envisioned by the Norwegian banking industry, then BankID may well get more than two million end-users in the near future. Because they'll all rely on the same

infrastructure, a successful DDoS attack will have severe economic consequences. Roughly two million end-users will not be able to access their accounts, transfer funds, or pay bills, and some of the online stores using BankID will not be able to sell goods. If BankID also becomes a national ID used by government agencies, then the consequences of a DDoS attack will be even more severe.

It has been argued that the likelihood of a DDoS attack against an online banking system in Norway is small because there is no group of people with strong enough motivation to carry out an attack. On the other hand, BankID will become a national "monoculture" [12] for online banking, which makes it easy to attack two million customers at the same time. As long as the BankID authentication procedure is not changed, it will not be difficult to repeat a DDoS attack at the application layer. While traditional crackers may be reluctant to attack BankID because they fear the inevitable investigation by Norwegian government agencies concerned with national security, other groups wanting to attack the Norwegian financial system to create chaos may find BankID a tempting target.

Examples of potential threats are terrorist groups in strong opposition to the Western society in general, and Norway in particular. Cyber terrorists [13] are both able and willing to disrupt critical information structures to cause harm in order to advance their own political or religious agendas. Other terrorist groups without the needed skill set can hire or coerce crackers to carry out attacks. The danger of terrorist attacks in Europe, the increasing usage of BankID, and the increasing number of DDoS attacks against businesses in Norway reported by the Norwegian National Security Authority, lead us to conclude that the likelihood of a DDoS attack against BankID will grow during the coming years.

Observation 1 *Because the authentication procedure in BankID utilizes SSNs and denies an end-user access after a few wrong login trials, it is particularly vulnerable to DDoS attacks—just like the authentication procedures in the other Norwegian Internet banking systems. The potential DDoS attacks represent a growing risk to end-users and web site owners.*

4.2 Strength of end-user authentication

Let us consider the strength of the end-user authentication in BankID consisting of a traditional two-factor authentication procedure followed by (part

of) a request-response authentication protocol as depicted in Figure 2. Assume that an adversary has obtained an end-user's PIN calculator or list of PINs. Furthermore, the adversary knows the end-user's fixed password, and—if needed—the fixed PIN used to activate the calculator. The adversary can then download the BankID client and give it access to the cryptographic functionality at the central infrastructure. The client can now complete the request-response authentication protocol with the BankID server. Hence, if the two-factor authentication is compromised, then the authentication of the end-user is compromised. Because BankID-member banks utilize different two-factor authentication mechanisms, the exact strength of the end-user authentication varies between the banks.

Observation 2 *The end-user authentication in BankID is no stronger than the two-factor authentication used in many older Internet banking systems.*

Surprisingly, the fixed password provided by the end-user during the first part of the BankID authentication procedure doesn't always strengthen the end-user authentication. One particular BankID-member bank, which provides each end-user with a PIN calculator requiring an activating PIN, lets any end-user ask for a new password by simply providing his SSN and a one-time PIN. The new password is displayed in the user's browser. Hence, if an adversary gets hold of the calculator and the activating PIN, he can also obtain a new valid password without knowing the old password!

4.3 Phishing/man-in-the-middle attacks

Each time an end-user downloads the BankID client, he also downloads HTML code containing parameters to the Java applet. While the applet itself is signed, the HTML code is not signed and changes to the parameters will not be detected by the user's Internet browser. Two parameters specifying URLs can be altered to make the BankID client communicate with the BankID server and central infrastructure through a proxy (server) controlled by a cracker. The proxy can be realized by software running on some computer.

To initiate this Man-in-the-Middle (MitM) attack, a phishing attack may first be used to trick a BankID customer into downloading modified HTML code together with the unaltered BankID client. When the customer starts a new session, the BankID client then connects to the proxy, which again

connects to the BankID server and central infrastructure. We have developed proof-of-concept code to show that the proxy can steal the session after the user has authenticated himself to the central infrastructure and the BankID server.

MitM attacks are a well established form of deceit. In 2006, several Norwegian Internet banks, not based on BankID, were victims of MitM attacks. The likelihood of MitM attacks on BankID will increase as BankID gets more users and the system becomes a national monoculture.

Observation 3 *Combined phishing/MitM attacks can be used to steal sessions initiated by BankID customers because it is possible to change the addresses to which the BankID client connects.*

4.4 DDoS/phishing attacks

A DDoS attack closing down BankID customers' accounts can be followed by a phishing e-mail attack to let the same customers "know" how they again can use their accounts. It is clear from Observation 1 that it is particularly easy to execute a DDoS attack to close down accounts. An e-mail can then notify each customer about the account closure, which the customer can easily verify, and then entice the customer into entering a one-time PIN and a password at a fake banking site to open the account. The attacker must either call the bank to try to reopen the account, or wait for the bank to do it, and then use the stolen one-time PIN and fixed password to access the account before the legitimate owner.

No such combined DDoS/phishing attack has yet been reported in the press as far as we know. However, both DDoS attacks and phishing attacks are increasing in frequency, and it may only be a question of time before combined attacks occur.

Observation 4 *BankID is potentially vulnerable to combined DDoS/phishing attacks where customers are tricked into entering one-time PINs and passwords on fake Internet banking sites.*

4.5 Exploiting the BankID server

A web application using the BankID infrastructure incorporates the BankID server software. The private key and the certificate belonging to the web

application are stored in an encrypted PKCS #12 file or an HSM. At least one version of the BankID server does not support the use of an HSM. In this case, the web site owner provides a fixed password to give the BankID server access to the decrypted file. If an outside cracker or rogue insider gets hold of the file and the password, then the web site owner's private key is exposed. An attacker with the private key can pass himself off as the web site owner. The seriousness of this threat is determined by how hard it is to get hold of the PKCS #12 file and how difficult it is to determine the owner's fixed password.

The encrypted PKCS #12 file has a known structure, which makes it possible to locate on a computer. When a cracker has access to this file, he can run a brute-force or dictionary attack to try to recover the accompanying password [14]. On the other hand, "social engineering" or "shoulder surfing" may be all that is needed. The attacker can also try to install a hidden camera or a hardware keylogger to obtain the password.

In the case where a cracker has no physical access to the computer, he may still be able to employ malicious software, or *malware*, to obtain the PKCS #12 file. The cracker can then run a dictionary attack to try to determine the password and obtain the private key. Alternatively, the malicious software can try to both copy the file and sniff the password using a software keylogger.

A cracker can implement a DoS attack by developing self-propagating malware which spreads to many computers and deletes the PKCS #12 files, thus, preventing web applications from utilizing BankID. Since it is only possible to detect malware by scanning for known patterns, the risk associated with the described DoS attack cannot be completely eliminated.

Observation 5 *A BankID server utilizing an encrypted PKCS #12 file to store the private key is potentially vulnerable to well-known password attacks to decrypt the file, and DoS attacks where malware deletes the file.*

4.6 Mitigating authentication risk

From Observation 1, the policy of using SSNs to identify customers and denying them access after a few wrong login trials must be changed because it enables efficient DDoS attacks on the application layer, potentially affecting more than two million customers in the near future. According to Observation 3, transactions should be authenticated for BankID to become

more robust against combined phishing/MitM attacks, and from Observation 4, passwords and PINs should not be transmitted from a BankID client to the central infrastructure since this solution is vulnerable to combined DDoS/phishing attacks. Passwords and PINs should only be used locally by the end-user to give the client access to the end-user's PKI credentials. A new end-user authentication solely based on the end-user's public-private key pair will increase the strength of the authentication beyond what is possible with traditional two-factor authentication. From Observation 5, all web applications using BankID should employ HSMs to store private keys.

5 Risk of non-repudiation service

In the following we first discuss three vulnerabilities in the BankID architecture with the potential to limit the degree of achievable non-repudiation. We then consider how the strength of the end-user authentication influences the degree of non-repudiation. Finally, we discuss how to mitigate the non-repudiation risk.

5.1 No trusted third party

BankID does not employ a trusted third party to achieve non-repudiation despite the fact that this is required by most non-repudiation protocols described in the literature [7]. On the contrary, the BankID-member banks have strong financial relationships with both end-users and merchants owning web sites. In particular, when the web sites are Internet banking sites, the banks own the sites as well as the complete BankID infrastructure, giving the banks a large amount of control over financial operations requiring non-repudiation.

The following scenario illustrates the problem with the non-existent third party. Assume that a BankID customer and his bank have both digitally signed a document. At some later point in time there is a conflict between the bank and the customer during which the bank claims it didn't sign the document. It is then up to the customer to show that the bank did in fact sign. Since the bank controls the Internet banking application and BankID is controlled by the Norwegian banking community, the bank has access to a wealth of technical and judicial information, whereas the customer has only a copy of the digitally signed document on his computer. Furthermore, the

bank has readily access to BankID experts, while the customer has only access to security experts without any inside knowledge of BankID. Consequently, since no third party has collected non-repudiation information to assist the customer during the conflict, the customer and his lawyers will find it very difficult to convince a judge that the bank really signed the document when it denies having done so.

Observation 6 *The non-repudiation service in BankID gives a bank an advantage over its customers during conflicts involving repudiation of digital signatures because the customers cannot rely on help from a trusted third party.*

5.2 Insecure local key storage on BankID server

A high degree of non-repudiation requires that it is very difficult for an outside cracker (or rogue insider) to obtain a web site's private key. Unfortunately, in some cases the private key is stored in an encrypted PKCS #12 file. A cracker may be able to obtain the fixed password used to decrypt the file since the password is vulnerable to dictionary and social engineering attacks. Once the cracker has the private key in the decrypted file he can sign documents without the knowledge of the key's rightful owner.

Observation 7 *Only web sites utilizing HSMs to store and use private keys can support a high level of non-repudiation.*

5.3 Central key storage

An end-user's private key is stored on the central BankID infrastructure controlled by The Norwegian Banks' Payment and Clearing Centre. According to [8], this key is only used inside an HSM. Since private keys must *only* be available to end-users [4, pp. 52, 93, 156] to achieve a high degree of non-repudiation, the central key storage raises several questions. How are the private keys generated on behalf of the customers and placed in the HSM without anyone possibly learning the value of the keys? How can customers provide the fixed password to activate the authentication and signing functionality without anyone else obtaining the password? Is a network connection from a customer made directly to the HSM to avoid MitM attacks from rogue insiders?

During a dispute a bank must provide a judge with convincing answers to these questions. In particular, a bank must explain why a rogue insider isn't able to exploit the HSM's application programming interface, modify existing code, install new malicious code, and/or modify server configurations to get access to keys.

The banks have already been able to convince the Norwegian Post and Telecommunications Authority how only the end-users can grant the central infrastructure access to private keys inside an HSM. However, the authority has refused to share its reasons for accepting central key storage. This is unfortunate since the lack of information makes it very difficult for an end-user (or his lawyer) to challenge a bank's claims about the non-repudiation during a dispute.

Not surprisingly, the Norwegian banking community has also refused to share any information with us about the system they use for non-repudiation. As far as we know, the non-repudiation protocols have not been analyzed by independent security experts or tested in Norwegian courts.

Observation 8 *The security-through-secrecy policy of the BankID-member banks gives them an advantage over their customers during conflicts because the customers and their lawyers have no access to technical information about the non-repudiation service.*

5.4 End-user authentication limits degree of non-repudiation

If the end-user authentication in a PKI is too weak then it is possible for a skilled cracker to steal a user's digital identity. The cracker can then digitally sign a document using the victim's identity. It can be difficult for the unfortunate user whose digital identity was misused, to show that he didn't sign the document. If the PKI offers a non-repudiation service, this will only increase the problem for the unfortunate user because the other signee has access to "credible evidence" showing that the user did sign when in fact it was the cracker who misused the user's identity. Hence, strong authentication of users is needed to achieve a high degree of non-repudiation.

According to Observation 2, the strength of the end-user authentication in BankID is limited by the strength of the two-factor authentication utilized by the BankID-member banks. Furthermore, Observations 3 and 4 point out

that the the two-factor authentication is vulnerable to well-known attacks. As a result, it is possible to steal users' identities.

Observation 9 *The end-user authentication in BankID limits the degree of non-repudiation. The strength of the authentication should be increased before customers digitally sign contracts concerning large-valued assets.*

5.5 Mitigating non-repudiation risk

From Observations 6 and 8, the non-repudiation service gives a bank an advantage over its customers because the customers cannot obtain technical information about the service or rely on help from a third party during a conflict. The Norwegian banks should release information about the technical and legal non-repudiation protocols. In particular, the banks need to publish their dispute resolution procedures. Only then will it be possible for the users to get an understanding of the true risk associated with the non-repudiation service. It follows from Observation 7 that web site owners wanting to use the service must invest in HSMs to store cryptographic keys. From Observation 9, the strength of the authentication should be increased to improve the level of non-repudiation.

6 Privacy risk

The meaning of the term *privacy* depends on the context in which it is used. We define privacy as the right of an individual to decide when and how sensitive personal information should be revealed. To get an understanding of the privacy risk associated with BankID, we'll consider how the system may be used to build customer profiles.

Let us first consider the situation where an end-user wants to authenticate a web site. During the authentication process the BankID client depends on the central infrastructure to verify the web site's X.509 certificate. Because the end-user has to authenticate to the central infrastructure and the certificate contains the web address of the site, both the end-user and the web site are uniquely identified by the central infrastructure.

The CAs in the central infrastructure have access to the personal information provided by all individuals wanting to become BankID end-users. Once they start using BankID, it follows from the above observation that

the central infrastructure can record e.g. where end-users shop and which government agencies they have dealings with.

Because the end-users must enter their SSNs during the authentication process, it is possible to link information from BankID with information in other commercial and governmental systems such as credit reporting agencies and taxation agencies. Hence, assuming BankID becomes the prevalent tool for online authentication of individuals in Norway, it will be possible to build increasingly *detailed profiles* over time revealing business and personal relationships of more than 2 million customers.

The US National Research Council has published a report [15] stating that individuals should know what information about them is stored in a national computer system, how this personal information is made available to third parties, how the information is updated, and most importantly, how they can prevent disclosure of the information. Neither of these requirements are fulfilled by BankID.

Observation 10 *The Norwegian banking community controls an ID system with the potential to build detailed profiles of roughly half of the Norwegian population as long as the BankID authentication utilizes X.509 certificates and SSNs. The BankID customers don't know how their personal information is utilized.*

We remark that the BankID client leaks information. It can be downloaded by anyone with a computer. If you enter the SSN of a BankID customer into the client, then the client will list the banks for which the customer uses BankID for authentication.

6.1 Mitigating privacy risk

The BankID system should be reviewed by independent privacy experts before it is allowed to become a de facto national ID system. Identified weaknesses in the privacy protection should be carefully examined and mitigated. In the long run a new authentication procedure, not using X.509 certificates and SSNs, should be introduced to minimize the system's negative effect on the end-users' privacy.

7 Conclusions

In April of 2007—after the risk analysis was completed—we were informed by a BankID representative that changes had been made to the system to mitigate the risk associated with MitM attacks (Observation 3). These changes may also increase the level of non-repudiation (Observation 9).

At the time of writing, late April 2007, the risk to BankID customers is still significant because (i) BankID is particularly vulnerable to DDoS attacks at the application layer, (ii) combined DDoS/phishing attacks may empty out customers' accounts, (iii), the lack of an independent third party in the non-repudiation service and the banks' security-through-secrecy policy gives them an advantage over their customers during conflicts involving repudiation of digital signatures, and (iv) the customers don't know how BankID utilizes their personal information. We recommend that steps are taken to mitigate (i)–(iv).

7.1 Who should own the remaining risk?

Even after a risk mitigation process is carried out there still is a residual risk associated with BankID. Hence, it is interesting to observe that if an outside cracker or rogue insider is able to empty out an account in a BankID-member bank, then the bank's responsibility is limited to one-hundred thousand Norwegian Kroner (about sixteen thousand US dollars) [9, p. 15].

If a bank is grossly negligent, then the limit doesn't apply. During a dispute it is up to the customer or his lawyers to establish that the bank has been grossly negligent. Experience shows that this is close to impossible since the Norwegian banks have long refused to share any technical information about their systems (see [2] for a further discussion of this point).

Since only the banks can strengthen the security of BankID, they should take the remaining risk and, thus, be liable for any loss caused by crackers and rogue insiders. It is then up to the banks to determine the best balance between investing in better security and simply covering the loss caused by fraud.

7.2 General recommendations

We make four general recommendations to limit the risk customers take when using a national PKI such as BankID. First, traditional secrets, e.g.

PINs and passwords, should not be transmitted from a client to the central PKI infrastructure. Hence, two-factor authentication should only be used locally to give the client access to the end-user's PKI credentials.

Second, if a PKI is found to be vulnerable to well-known attacks such as phishing and MitM attacks, then immediate steps should be taken to mitigate the risk to the end-users. This is of particular importance when the end-users don't have a convenient alternative to the services offered by the PKI.

Third, the non-repudiation service in a national PKI should not be sanctioned by any government before independent lawyers and security experts have evaluated the legal and technical protocols and determined the true level of non-repudiation. The results of such an evaluation should be made public.

Four, no citizen in any country should be forced to use a national ID system before an analysis of the system's privacy implications is made public and any discovered weaknesses in the privacy protection are corrected. Robust safeguards against profile building should be in place before any ID system is accepted by a nation's government.

References

[1] K. J. Hole, V. Moen, and T. Tjøstheim, "Case Study: Online Banking Security," *IEEE Security & Privacy*, vol. 4, no. 2, 2006, pp. 14–20.

[2] K. J. Hole, V. Moen, and A. N. Klingsheim, "Lessons from the Norwegian ATM system," submitted to *IEEE Security & Privacy*.

[3] A. Jones and D. Ashenden, *Risk Management for Computer Security*, Elsevier, 2005.

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

[5] W. Ford and M. S. Baum, *Secure Electronic Commerce*, 2nd Edition, Prentice Hall, 2001.

[6] S. T. Kent and L. I. Millett, Editors, *Who Goes There?*, National Academies Press, 2003.

[7] J. Zhou, *Non-repudiation in Electronic Commerce*, Artech House, 2001.

[8] The Norwegian Banks' Payment and Clearing Centre (BBS), "BankID FOI White Paper," Release 2.0.0, 2006 (in Norwegian).

[9] Bankenes Standardiseringskontor, "Norsk BankID sertifikatpolicy for banklagrede kvalifiserte sertifikater til personkunder," Version 1.1, 2005 (in Norwegian).

[10] RSA Laboratories, "PKCS 12 v1.0: Personal Information Exchange Syntax," 1999.

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

[12] D. Geer, R. Bace, P. Gutmann, P. Metzger, C. P. Pfleeger, J. S. Quarterman, and B. Schneier, "Cyberinsecurity: the cost of monopoly," Sep. 2003;
`www.ccianet.org/filings/cybersecurity/cyberinsecurity.pdf`.

[13] S. C. McQuade, *Understanding and Managing Cybercrime*. Pearson, 2006.

[14] R. E. Smith, *Authentication*. Addison-Wesley, 2002.

[15] S. T. Kent and L. I. Millet, Editors, *IDs—Not That Easy: Questions About Nationwide Identity Systems*, National Academies Press, 2002.