

# Weaknesses in BankID, a PKI-substitute Deployed by Norwegian Banks

EuroPKI 2008

Kristian Gjøsteen

Department of Mathematical Sciences, NTNU

June 17, 2008

- 1 Problem Statement and Background
- 2 Protocol Flaws
- 3 Cryptographic Flaws
- 4 Privacy Flaws
- 5 Concluding Remarks

# Problem Statement

BankID is a system developed and deployed by Norwegian banks to solve the following challenging problem:

## Problem

Provide secure login/identification and e-signatures over the internet.

Applications include access control for health records and other sensitive information. The security requirements are therefore very high, mandating extreme care in design and implementation.

The banks have released no detailed information about the BankID system to the public.

# Overview of BankID – login

The user name is entered,

**Innlogging**

Logg inn som:

- Privatkunde
- Bedriftskunde

Velg innloggingstype:

- BankID**  
Dersom du har BankID, kan du logge deg på portalen her.
- DigiPass / Kodekort**  
Dersom du har DigiPass eller kodekort, kan du logge deg på portalen her.

---

Fødselsnummer / brukernavn:

# Overview of BankID – login

The user name is entered, then the one-time code and password.



The image shows a screenshot of the BankID login interface. At the top left, the text reads "Identifisering" and "SpareBank 1 NettBank". At the top right, there is a "BankID" logo. The main area contains two input fields: "Sikkerhetskode:" and "Personlig passord:". Below these fields are three buttons: "OK", "Endre passord", and "Avbryt".

**Identifisering**  
SpareBank 1 NettBank

**BankID**

Sikkerhetskode:

Personlig passord:

OK    Endre passord    Avbryt

## Problem

Provide secure login/identification and e-signatures over the internet.

A typical user

- does not understand URLs;
- does not understand SSL; and
- reveals passwords and one-time codes to anyone who asks.

A good system provides security for typical users.

## Problem

Provide secure login/identification and e-signatures over the internet.

A typical user

- does not understand URLs;
- does not understand SSL; and
- reveals passwords and one-time codes to anyone who asks.

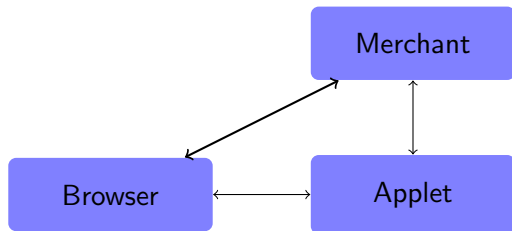
A good system provides security for typical users.

## Observation

BankID does not provide security!

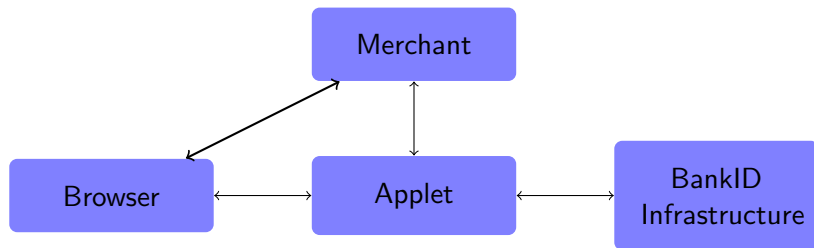
# Overview of BankID II

The password entry page contains a signed Java applet. The applet runs an authentication protocol with the merchant and redirects the browser to the user's account page.



# Overview of BankID II

The password entry page contains a signed Java applet. The applet runs an authentication protocol with the merchant and redirects the browser to the user's account page.



The BankID Infrastructure safe-guards the user's private signing key, and does public and private key operations on the applet's behalf.

# Threat Model II

## Problem

Provide secure login/identification and e-signatures over the internet.

## Reduced Security Claim

BankID provides security, provided the user ensures that he only interacts with the real BankID applet.

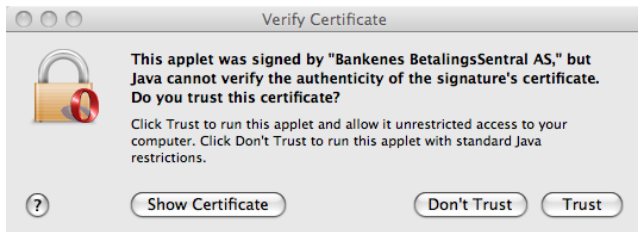
# Threat Model II

## Problem

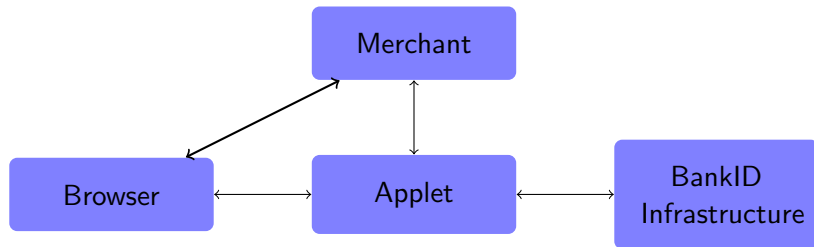
Provide secure login/identification and e-signatures over the internet.

## Reduced Security Claim

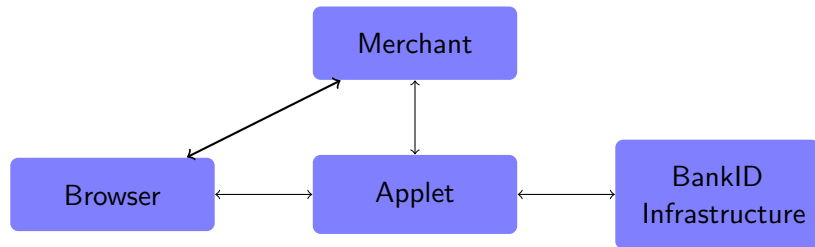
BankID provides security, provided the user ensures that he only interacts with the real BankID applet.



## Previous work: session hi-jack



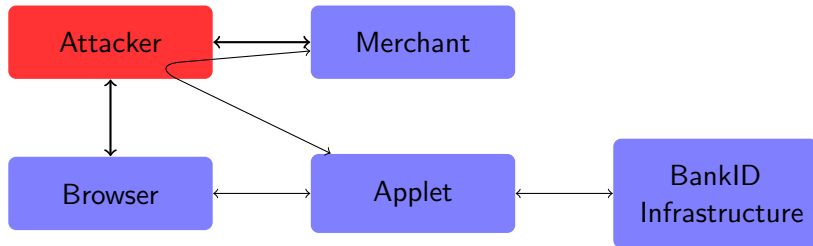
## Previous work: session hi-jack



Observation – Hole et al. (2007)

The applet did not verify its communication parameters.

## Previous work: session hi-jack



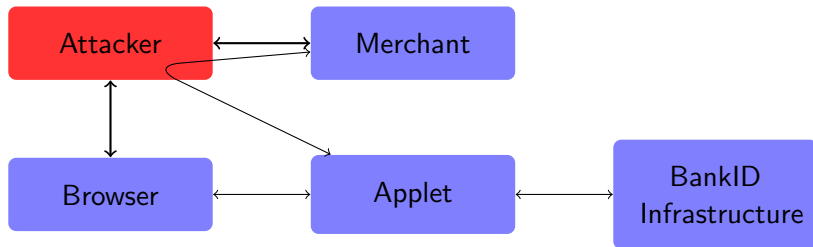
Observation – Hole et al. (2007)

The applet did not verify its communication parameters.

Consequence

A combined phishing/man-in-the-middle attack was possible.

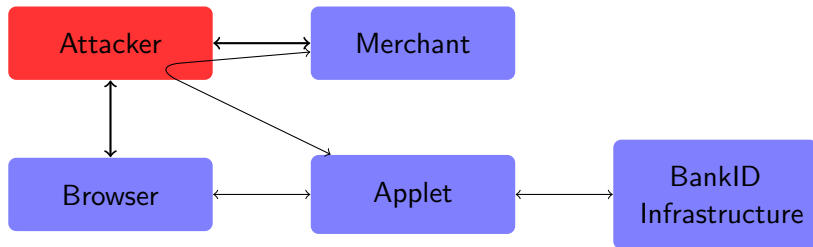
# Communication Parameters Verification Turned Off



## Observation

The code to verify the communication parameters was not missing, but **disabled**.

# Communication Parameters Verification Turned Off



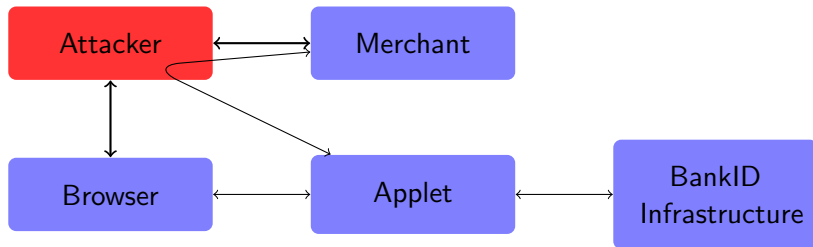
## Observation

The code to verify the communication parameters was not missing, but **disabled**.

## Question

Why, oh, why??

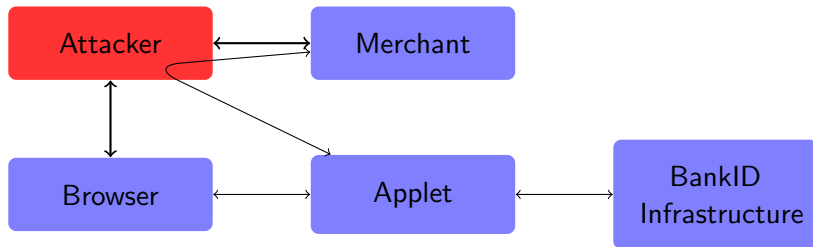
# Communication Parameters Verification Flawed



## Observation

The verification code did not check if SSL should be used or not.

# Communication Parameters Verification Flawed



## Observation

The verification code did not check if SSL should be used or not.

## Consequence

A man-in-the-middle attack was possible, even when the verification code was enabled.

- 1 Why use complicated and unreliable hostname and ip address verification, when SSL provides reliable cryptographic authentication?

# Questions

- ① Why use complicated and unreliable hostname and ip address verification, when SSL provides reliable cryptographic authentication?
- ② Who expects more attacks along these lines to be found?

# Questions

- ① Why use complicated and unreliable hostname and ip address verification, when SSL provides reliable cryptographic authentication?
- ② Who expects more attacks along these lines to be found?

Recall: the BankID infrastructure safe-guards the user's signing key.

# Questions

- ① Why use complicated and unreliable hostname and ip address verification, when SSL provides reliable cryptographic authentication?
- ② Who expects more attacks along these lines to be found?

Recall: the BankID infrastructure safe-guards the user's signing key.

- ③ What's to keep insiders from abusing the user's key?

# Questions

- 1 Why use complicated and unreliable hostname and ip address verification, when SSL provides reliable cryptographic authentication?
- 2 Who expects more attacks along these lines to be found?

Recall: the BankID infrastructure safe-guards the user's signing key.

- 3 What's to keep insiders from abusing the user's key?

## Observation

There can be no security unless

- 1 the infrastructure uses tamper-resistant hardware correctly; and
- 2 trusted employees are honest.

# Pseudo-Random Number Generation

This is a common cryptographic idiom:

*public-key-encrypt(k) + symmetric-encrypt(k, message),*

where **k** is a random symmetric key.

## Fact

Random numbers are difficult to generate on a computer.

# Pseudo-Random Number Generation

This is a common cryptographic idiom:

*public-key-encrypt(k) + symmetric-encrypt(k, message),*

where  $k$  is a random symmetric key.

## Fact

Random numbers are difficult to generate on a computer.

## PRNG

A pseudo-random number generator outputs numbers that look random to anyone who doesn't know the input seed.

# Pseudo-Random Number Generation

This is a common cryptographic idiom:

*public-key-encrypt(k) + symmetric-encrypt(k, message),*

where  $k$  is a random symmetric key.

## Fact

Random numbers are difficult to generate on a computer.

## PRNG

A pseudo-random number generator outputs numbers that look random to anyone who doesn't know the input seed.

## Observation

A seed is still needed!

# BankID seed generation

The applet uses Java's self-seeding PRNG implementation. For certain Java versions, the applet generates the seed itself.

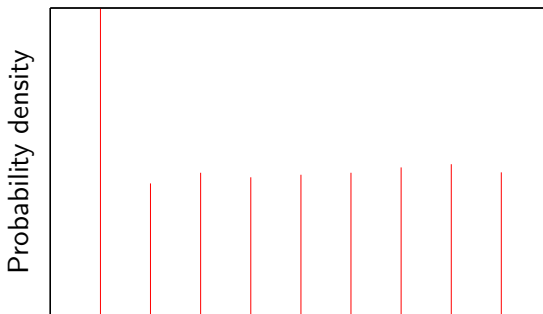
# BankID seed generation

The applet uses Java's self-seeding PRNG implementation. For certain Java versions, the applet generates the seed itself.

Analysing a random number generator is in general hard,

# BankID seed generation

The applet uses Java's self-seeding PRNG implementation. For certain Java versions, the applet generates the seed itself. Analysing a random number generator is in general hard, but sometimes very easy.



Probability density plot for approx. 7500 samples.

# BankID seed generation

The applet uses Java's self-seeding PRNG implementation. For certain Java versions, the applet generates the seed itself.

## Consequence

Insider attacks against users with certain Java versions may be possible.

PKCS#1 is a standard for doing public key encryption and digital signatures based on RSA.

- Version 1.5 was published in 1993 and broken in 1998.
- Version 2.0 was published in 1998 and lists easily implemented countermeasures for version 1.5.

PKCS#1 is a standard for doing public key encryption and digital signatures based on RSA.

- Version 1.5 was published in 1993 and broken in 1998.
- Version 2.0 was published in 1998 and lists easily implemented countermeasures for version 1.5.

## Observation

BankID uses the padding from 1.5. We cannot find any countermeasures implemented.

PKCS#1 is a standard for doing public key encryption and digital signatures based on RSA.

- Version 1.5 was published in 1993 and broken in 1998.
- Version 2.0 was published in 1998 and lists easily implemented countermeasures for version 1.5.

## Observation

BankID uses the padding from 1.5. We cannot find any countermeasures implemented.

## Consequence

Insider attacks against any user may be possible.

The BankID infrastructure gathers too much information about its users:

- When a user logs in to a website, the BankID infrastructure is told which website this is.
- When a user signs a document, the BankID infrastructure gets a hash of the document.

## Consequence

BankID functions as a moderately effective surveillance system.

The BankID infrastructure gathers too much information about its users:

- When a user logs in to a website, the BankID infrastructure is told which website this is.
- When a user signs a document, the BankID infrastructure gets a hash of the document.

## Consequence

BankID functions as a moderately effective surveillance system.

## Remark

Standard cryptographic tools, such as blind signatures, have been developed for this purpose.

## Concluding Remarks

- BankID in its current form clearly does not satisfy reasonable security requirements for eID solutions.

## Concluding Remarks

- BankID in its current form clearly does not satisfy reasonable security requirements for eID solutions.
- BankID does not satisfy minimal privacy requirements for an eID solution.

# Concluding Remarks

- BankID in its current form clearly does not satisfy reasonable security requirements for eID solutions.
- BankID does not satisfy minimal privacy requirements for an eID solution.
- A serious failure in quality assurance during the development of BankID has resulted in a number of serious design and implementation mistakes, some of them quite surprising.

# Concluding Remarks

- BankID in its current form clearly does not satisfy reasonable security requirements for eID solutions.
- BankID does not satisfy minimal privacy requirements for an eID solution.
- A serious failure in quality assurance during the development of BankID has resulted in a number of serious design and implementation mistakes, some of them quite surprising.
- Any eID candidate solution should undergo thorough external expert review, preferably in public.