Security and Privacy Improvements for the Belgian eID Technology

Pieter Verhaeghe, Jorn Lapon, Bart De Decker, Vincent Naessens and Kristof Verslype

Abstract The Belgian Electronic Identity Card enables Belgian citizens to prove their identity digitally and to sign electronic documents. At the end of 2009, every Belgian citizen older than 12 years will have such an eID card. In the future, usage of the eID card may be mandatory. However, irresponsible use of the card may cause harm to individuals.

Currently, there exist some privacy and security problems related to the use of the eID card. This paper focuses on solutions to tackle these problems. A new authentication protocol is introduced to substantially reduce the risk of abusing the single sign-on authentication and privacy friendly identity files are proposed to improve the citizen's privacy.

1 Introduction

Belgium has introduced an electronic identity card [1, 2] in 2002 as one of the first countries in Europe. The Belgian government aims at completing the roll-out by the end of 2009. At that time, each citizen will be the owner of an eID card. The card enables individuals to prove their identity digitally and to sign electronic documents. The Belgian eID card opens up new opportunities for the government, their citizens, service providers and application developers.

It is clear that many application developers benefit from this evolution. Today, integrating eID technology for authentication purposes is a real hot topic in Belgium. However, the usage of the eID card involves a few security and privacy hazards. Still,

Pieter Verhaeghe, Bart De Decker and Kristof Verslype
Katholieke Universiteit Leuven, Department of Computer Science, Celestijnenlaan 200A, 3001 Heverlee, Belgium, e-mail: `firstname.lastname@cs.kuleuven.be`

Jorn Lapon and Vincent Naessens
Katholieke Hogeschool Sint-Lieven, Department of Industrial Engineering, Gebroeders Desmetstraat 1, 9000 Gent, Belgium, e-mail: `firstname.lastname@kahosl.be`

most citizens are unaware of these pitfalls, which is disturbing, since the usage of the card is highly encouraged both by the government and the industry.

This paper first explains the Belgian eID card technology in section 2 and outlines security and privacy hazards related to the usage of the card in section 3. Next, a new authenication protocol ***auth*** (using the eID card) is presented and privacy-friendly identity files are introduced in section 4 and evaluated in section 5. Finally, the paper draws conclusions and describes directions for future research.

2 Belgian eID Technology

This section gives an overview of the current Belgian eID technology. A more elaborate description can be found in [1, 2].

2.1 Contents of the Belgian eID card

Private information such as the owner's name, birthdate and -place, address, digital picture and National Registration Number is stored in three separate files: an identity file, an address file and a picture file. The files are signed by the National Registration Bureau (NRB). The National Registration Number (NRN) is a unique nation-wide identification number that is assigned to each natural person.

Two key pairs are stored on the eID card. One key pair is used for authentication, the other is used is for signing. The (qualified) e-signatures are legally binding. The public keys are embedded in a certificate which also contains the NRN and the name of the card holder. The private keys are stored in a tamper-proof part of the chip and can only be activated (not *read*) with a PIN code. Authentication is single sign-on, i.e. the PIN code is only required for the first authentication. For signing, a PIN code is needed for each signature[3].

2.2 Belgian Public Key Infrastructure

The certificates on the eID card are part of a larger hierarchical infrastructure, the Belgian Public Key Infrastructure (be-PKI) [4]. The hierarchy is illustrated in figure 1. The citizen's signature and authentication certificates are issued by a *Citizen_CA* which is certified by the *Belgium_Root_CA*. Other governmental CAs such as the *Card_Admin_CA* and *Government_CA* also have certificates issued by the *Belgium_Root_CA*. The former can update the eID card. The latter certifies the National Registration Bureau (NRB) which signs the identity and address file and offers other services in the public sector. The *Belgium_Root_CA* has two certificates. The first is a self-signed certificate, that allows for offline validation of the signature and authen-

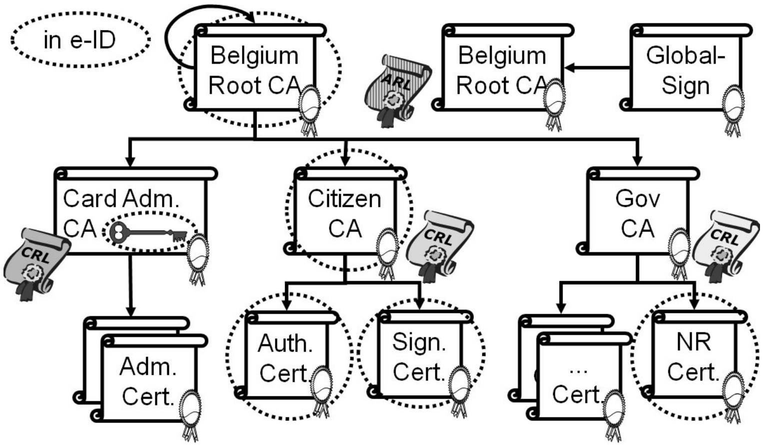


Fig. 1 Belgian Public Key Infrastructure

tication certificates on the eID card. The second certificate is issued by GlobalSign. The latter is typically known to popular applications (such as browsers) and allows for the automatic validation of electronic signatures. The PKI provides Authority Revocation Lists (ARL) and Certificate Revocation Lists (CRL) [5] that keeps the serial numbers of the revoked certificates.

2.3 *Official middleware*

The cryptographic functionalities in the Belgian eID card are accessed through middleware [6]. Applications typically interact with the card via a simple API [7] offered by this middleware. If a document needs to be signed, the middleware passes a hash of the document to the card. Similarly, a hash of the challenge is passed to the card for authentication purposes. When an application wants to authenticate or sign a document with the eID card, the middleware requests the user to enter his PIN code. The middleware can also verify the validity of the certificates (using CRL or OCSP). It is important to note that the use of the official middleware is not mandatory. Several alternatives, developed by different companies, are available.

3 Security and Privacy Hazards

This section elaborates on security and privacy hazards related to the usage of the eID card. Abuse of the single sign-on authentication mechanism and unrestricted release of the card holder's personal data are the major threats.

3.1 Single Sign-On authentication

The card holder only needs to enter his PIN code for the first authentication. As long as the card is not removed from the reader, SK_{auth} remains activated. Hence, when the user browses to multiple sites that require eID authentication, authentication is performed transparently except for the first site. This implies that users are unaware that identity information (i.e. the authentication certificate) is transferred to these sites. Moreover, a trojan horse can secretly log in to these sites and collect or even modify the citizen's private data.

Some websites already use the eID card to set up a mutual authenticated HTTPS connection.

3.2 Unrestricted release of personal data

The identity, address and picture files on the card are not PIN-protected. As soon as the eID card is inserted in a smart card reader, these three files can be read by any application.

Usually, the official middleware will intervene and request the user's consent to access these files. However, a program can directly access the card and collect these files. This is especially problematic when children use their eID card to login at a "secure" chat box. As the identity file can be copied to another smart card, identity theft is quite easy if authentication is not requested (e.g. to get access to the municipality's rubbish dump).

3.3 Other threats

Since the certificates contain the NRN, all actions performed by the same citizen can be linked. The date of birth and gender of the individual can also easily be derived from the NRN.

Another threat is related to the default settings of the middleware. When the citizen inserts his eID card in the card reader, the authentication and signing certificates are stored in persistent memory by default. This behaviour can be disabled. However, some applications will fail to authenticate with the eID card (e.g. Internet Explorer).

A malicious application can easily deceive the user and let the eID card sign a different document than the one intended. The current user interface of the middleware does not show which document is actually signed.

A full list of threats can be found in a technical report [8].

4 Improving the security and privacy properties of the eID card

This section presents a number of improvements to increase the security of the eID technology and to make it more privacy friendly. Some solutions can be realised in software. However, it would be better to incorporate them in the card. Others require a modification of the card.

4.1 A new authentication protocol: auth

Since single sign-on authentication can easily be abused by trojan horses, traditional client authentication (HTTPS) should be replaced by a new protocol.

4.1.1 Requirements

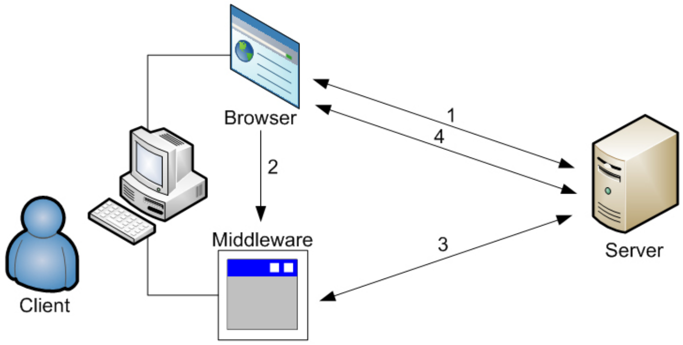
- The protocol should prevent trojan horses to authenticate secretly in the citizen's name and in addition require the user's consent for every authentication.
- The protocol should tackle man-in-the-middle attacks.
- The protocol should handle the authentication of sessions between a client and a web server. This makes it easy to integrate with web access.
- The protocol must be usable with every authentication mechanism that is based on a challenge and signature scheme. It should be compatible with all kinds of eID cards that allow for authentication based on a challenge-response protocol.
- The protocol only requires the presence of the eID card for a short time. After successful authentication, the card can be removed from the card reader.

4.1.2 Description and implementation

Figure 2 shows the message flow of the ***auth*** protocol. First, the user browses to a web page containing an ***auth***-URL. The URL has the following syntax:

```
auth://SPhost/path/auth-web-service#sessionID&types
```

A new HTTP session is started by the web server when this web page is requested. Each session has an unique number (*sessionID*). This number can be either stored in a cookie on the user side or can be propagated in the URL. When clicking on the ***auth***-URL, the module in the middleware that handles the ***auth*** protocol is executed. The middleware will pop-up a warning window that shows the (domain) name of the other party, it will invite the user to select an authentication means from the list of supported *types* (e.g. an eID card) and to give his consent for the authentication. Sometimes, the user will have to activate the authentication means by entering a PIN code or a password. The middleware module will set up a separate HTTPS



1. There exists an HTTP session between client and server.
2. User clicks on ***auth*** URL.
3. User authenticates to server and gives a reference of the HTTP session.
4. User can access private content in existing HTTP session.

Fig. 2 Connection flow of ***auth*** protocol

connection with the web server and ask for a challenge. Validation of the server certificate is important when setting up this connection to prevent man-in-the-middle attacks. Next, extra information is appended to the challenge (the type of authentication means, the domain name of the SP, the *sessionID* and a *middleware_secret*) and the aggregate is signed with the authentication key. The *middleware_secret* is a secret message which is programmed in the middleware code and hidden through code obfuscation. The generated signature is sent together with the certificate chain of the used “authentication” keypair to the web server. The middleware module on the server verifies the signature and the certificate chain. Then the server needs to check the equality of the IP addresses used by the client in the browse-session and in the authentication session. If all the checks pass, the browse-session is converted into an authenticated session and the user can browse to private pages on the website. The session is active until the user logs out or the session times out. Table 1 contains a detailed description of the ***auth*** protocol. The pop-up message described in (1.b.12) can be avoided by including a Javascript program in the HTML-page with the ***auth***-URL, which continuously polls for the termination of the ***auth*** protocol.

4.1.3 Modifying the card

The ***auth*** protocol can be implemented by only adapting the current middleware. However, the security can be increased by having the card implement the ***auth*** protocol. Currently, the eID card receives a challenge from the middleware and signs it after the PIN is entered. If other parameters like *sessionID* and *SPhost* can also be sent to the card for authentication, the card can compose the message that has to be signed (like in 1.b.5). The *middleware_secret* can then be omitted, since the card will only sign when the user has given his consent (OK button or PIN code).

(1.a) Browse to login page

(1)	U → SP	:	HTTPS request
		:	Header: [GET /login.html — Host: *SPhost*]
(2)	SP	:	startSession(*sessionID*; IP_{client}; *challenge*; $< now + timeout >$)
(3)	U ← SP	:	HTTPS response
		:	Header: [Set-Cookie: session=*sessionID*; Expired=$< now + timeout >$; secure]
		:	Body: [HTML page with hyperlink to “auth://*SPhost*/auth.php#*sessionID*&*types*”]

(1.b) Authentication over a new connection

(1)	U	:	Request user’s consent with pop-up window
		:	[Do you want to authenticate with this authentication *type* to *SPhost*?]
(2)	U	:	if($user\_input ==$ “yes”){select authentication means and activate it} else {abort}
(3)	U → SP	:	HTTPS request
		:	Header: [POST /*path*/auth.php?session=*sessionID*&type=*typeID* — Host: *SPhost*]
		:	Body: [“getChallenge”]
(4)	U ← SP	:	HTTPS response
		:	Body: [*challenge*]
(5)	U	:	$signature = \text{sign}_{SK_{auth}}(< type \parallel challenge \parallel SPhost \parallel sessionID \parallel$
		:	$middleware\_secret >)$
(6)	U → SP	:	HTTPS request
		:	Header: [POST /*path*/auth.php?session=*sessionID* — Host: *SPhost*]
		:	Body: [response=*signature*&authCertificate=$CertChain_{auth}$]
(7)	SP	:	if(IP_{client} != lookupIP(*sessionID*)) abort
(8)	SP	:	if(validateCertificate($CertChain_{auth}$) == false) abort
(9)	SP	:	if($\text{verify}_{PK_{auth}}(signature; < type \parallel challenge \parallel SPhost \parallel sessionID \parallel$
		:	$middleware\_secret >) ==$ false) abort
(10)	SP	:	setSessionAuthenticated(*sessionID*; *signature*; $< now + timeout >$)
(11)	U ← SP	:	HTTPS response
		:	Body: [“OK”]
(12)	U	:	Show pop-up to user that authentication is performed successfully.

(1.c) Browse to private content

(1)	U → SP	:	HTTPS request
		:	Header: [GET /private/index.html — Host: *SPhost* — Cookie: session=*sessionID*]
(2)	SP	:	if(isAuthenticated(*sessionID*) $==$ false) abort
(3)	U ← SP	:	HTTPS response
		:	Header: [Set-Cookie: session= *sessionID*; Expired=$< now + timeout >$; secure]
		:	Body: [$<$ content of requested page $>$]

Table 1 The ***auth*** protocol

4.2 Privacy friendly identity files

With the current eID card, it is only possible to disclose the entire identity, address and picture files. Otherwise, the server cannot check if the signature of the NRB on the identity file is valid. This section introduces more privacy friendly identity files (PFID-files). The concept allows for releasing only the necessary personal attributes.

4.2.1 Hashed attributes

The PFID-file contains for each attribute the hash of its value. Hence, it is not possible to extract personal information out of the hash values without knowing the attribute value. To reduce brute-force or dictionary attacks, the hash function is randomized by adding an attribute specific random number to the plaintext value: hash($ATTRIBUTE \parallel rand_{ATTRIBUTE}$). The PFID-file is certified (signed) by the National Registration Bureau (NRB). To disclose certified personal attributes, the card releases the signed PFID-file together with the plaintext values and the attribute specific random numbers of these attributes: *ATTRIBUTE*, $rand_{ATTRIBUTE}$. The card should request the user's consent before releasing personal data. This consent could be given by entering a PIN code or pressing the OK button on a card reader with a separate PIN pad. The other party can verify the values by calculating the hashes and comparing them with the values in the PFID-file. Optionally, the user's consent could be overridden (no PIN code required) after proper authentication by privileged service providers. The latter can be useful for border control, police, emergency services, etc.

4.2.2 Encrypted attributes

Some attributes can be encrypted in the identity file. Instead of just storing the National Registry Number (NRN) as attribute value, the NRN can be encrypted with a symmetric key only known by the government or by another trusted third party (TTP). The enciphered NRN serves as a unique pseudonym and can - in case of abuse - be deanonymized by that TTP during a legal investigation.

4.2.3 PFID-files on eID card

A PFID-file contains no personal information and does not need to be protected. However, in order to be sure that released personal attributes really belong to the card owner (and are not simply copied from another card), it is necessary to have that owner authenticate to the service provider. The service provider then needs to verify whether the PFID-file and the authentication certificate refer to the same holder and whether the certificate is still valid.

The attribute specific random number can be calculated from a master random number: $rand_{ATTRIBUTE} = \text{hash}(masterRandom \parallel ATTRIBUTE)$; and can be calculated at runtime. Hence, the card only needs to store (1) the plaintext values of the personal attributes, (2) the master random number, (3) the signature of the NRB on the PFID-file and (4) the PIN-code for the user's consent. However, this requires that the card can calculate the hash at runtime. When no hash function is available, more storage is needed to store all the attribute hashes. To implement PFID-files, the API must be extended to pass the list of requested attributes to the card.

4.2.4 Multiple domains

With only one PFID-file, multiple actions of the same citizen can easily be linked. To reduce linkability, multiple PFID files can be created and signed by the NRB. Each file is assigned to a domain and should only be used in that domain: e.g. “GOVERNMENT”, “COMMERCIAL”, “MEDICAL”, etc. Linkability is then only possible within one domain. A similar technique is used in the German eID card[9]. That card also works with domains.

To build multiple PFID-files, the hash values must be different in order to prevent linkability. A unique domain is concatenated to the master random value and the attribute name to calculate the random value (see table 2):

$$rand_{(DOMAIN,\,ATTRIBUTE)} = hash(MasterRandom \| DOMAIN \| ATTRIBUTE)$$

For encrypted attributes, the domain name precedes the actual attribute value. Hence, the ciphertext is different for each domain. The ciphertext of the domain and the NRN can be considered as a domain specific pseudonym for the card holder.

Attribute values	PFID-file for *DOMAIN*
Nym$_{DOMAIN}$	encrypt$_{K_{NRB}}$($DOMAIN$ || NRN)
Name	hash(Name || $rand_{(DOMAIN,Name)}$)
Surname	hash(Surname || $rand_{(DOMAIN,Surname)}$)
Street	hash(Street || $rand_{(DOMAIN,Street)}$)
Zip code	hash(Zip code || $rand_{(DOMAIN,ZipCode)}$)
Municipality	hash(Municipality || $rand_{(DOMAIN,Municipality)}$)
Birth location	hash(Birth location || $rand_{(DOMAIN,BirthLocation)}$)
Birth date	hash(Birth date || $rand_{(DOMAIN,BirthDate)}$)
Hash photo	hash(Hash photo || $rand_{(DOMAIN,Hashphoto)}$)
...	...
	signature of the National Registration Bureau on the PFID-file for *DOMAIN*

Table 2 Overview of privacy friendly identity file

To link PFID-files to card owners, different authentication certificates are necessary each referring to its own Nym$_{DOMAIN}$. Each certificate corresponds to its own keypair: ($SK_{auth_{DOMAIN}}$, $PK_{auth_{DOMAIN}}$). The only difference between the certificates is (1) the subject field, (2) the public key and hence (3) the signature of the certificate. The subject value is the domain pseudonym. Hence, each certificate can be linked to the corresponding PFID-file for each domain.

4.2.5 Storage requirements

If only one PFID-file is used on the eID card, the additional storage space that is required compared to the current eID card is very small. An additional master random value needs to be stored.

When using multiple domains, more space is needed on the card. However, the extra space per domain is quite small. For each PFID-file, an extra Nym$_{DOMAIN}$, other encrypted attributes and the NRB's signature per PFID-file must be stored. Also, room for an extra authentication keypair and the signature on the domain specific authentication certificate must be provided.

4.2.6 Updating PFID-files

Sometimes, the PFID-files need to be updated (e.g. if a citizen moves to a new address). This implies that the signatures of all PFID-files need to be updated. The NRB must build the new PFID-files by asking the master random value from the eID card. This is done after mutual authentication. This step is also needed with the current eID card to update the address file. As the NRN cannot be changed without replacing the eID card (because it is printed on it) the pseudonyms for the domains will be the same. Hence, the authentication certificates do not need to be updated.

4.3 Other improvements

Currently, a citizen has to trust the middleware on his machine when using his eID card. To prevent that trojan horses mislead the citizen, smart card readers with a pinpad and LCD screen should be preferred to those without these features. The card can communicate via the LCD screen with the user (e.g. show hash value of the document to be signed, show the personal information that will be disclosed, etc.), the user is able to give his consent for or abort an operation, and PIN codes cannot be intercepted through key logging.

5 Evaluation

The ***auth*** protocol tackles the single sign-on problem by requesting user's consent before signing a challenge. However, with the current eID card, the PIN is only necessary for the first authentication. It would be better if future versions of the Belgian eID card no longer implement single sign-on. Moreover, to increase the security, the card should implement the ***auth*** protocol itself.

The ***auth*** protocol includes some countermeasures against man-in-the-middle attacks. The client checks the certificate chain of the server and will only trust certificates issued by a configured set of CAs. Moreover, *SPhost* must be included in the certificate of the server. Finally, the client side will only connect to the service provider with the DNS name *SPhost*.

An attacker could forward an ***auth***-URL of his own browser session to another user and ask him to authenticate on his behalf. This authentication will only succeed if the external IP address (known to the server) of the victim is the same as that of the attacker (e.g. if they are behind the same proxy of NAT).

The implementation of the ***auth*** protocol can easily be integrated in every browser. Moreover, other types of authentication means (with other eID cards) are possible. The use of HTTPS as communication channel ensures that the authentication messages are protected against tampering and eavesdropping. In comparison with eID client authentication over HTTPS in a browser, the implementation of the ***auth*** protocol has its own trust policy for server certificates. Hence, the application can enforce that the certificate of the web server contains the DNS name. Otherwise, it aborts the authentication. In browsers, users can ignore this exception.

The ***auth*** protocol has as well advantages for the server side. In the current setting, a reverse proxy[10] is needed to implement the correct OCSP validation in the web server of the client's "authentication" certificate. Since most webmasters do not have access to the configuration files of the web server (e.g. shared hosting), they cannot use HTTPS with eID client authentication at this moment. Installing a web service and the middleware module is sufficient for the ***auth*** protocol to authenticate the client and does not require any configuration changes to the web server.

After a successful authentication with the ***auth*** protocol, the user can remove his eID card from the cardreader. This reduces the risk that trojan horses abuse the eID card.

The new PFID-files prevent that unauthorized applications have unrestricted access to the personal information stored on the eID card. Moreover, the user is no longer obliged to release all personal attribute values. The personal attributes of the identity and address file can be merged. Each domain has a separate PFID-file to prevent linkabilities between separate domains. However, linkabilities in the same domain are still possible. The PFID-files are signed by the National Registration Bureau. Hence, the server can easily check the integrity of the attribute values.

The PFID-files are not PIN protected. Copying the files from an eID card to another smart card is possible. Hence, the verifying party cannot be sure if the identity information on the eID card corresponds to that of the owner of the card. An authentication is necessary to be sure about the identity of the user. Using only identity files as a basis for access control to applications, buildings, etc. can imply a serious security threat.

The master random number is only stored on the card. The NRB only needs this value if new PFID-files must be updated. The NRB does not need to record this master random number for every citizen. This value can be retrieved from the card after proper authentication.

6 Conclusion and future work

The usage of the Belgian eID card implies some privacy and security hazards. The proposed solutions in this paper try to tackle the most important threats. However, these improvements cannot solve all the problems with the current eID card. Although it is quite easy to implement some of the proposed improvements in order to improve the security of the current eID card. To prevent abuse of the eID cards in the near future, the Belgian government should deploy a more secure and more privacy-friendly version of the eID card as soon as possible.

The main contribution of this paper is the ***auth*** protocol and the privacy friendly identity files. The ***auth*** protocol tackles single sign-on authentication and trojan horses at the client side. The PFID files only disclose the necessary personal information and reduce linkabilities.

As a first step, the paper proposes to implement the ***auth*** protocol in software. The next step would be a new eID card that implements PFID-files and the ***auth*** protocol itself.

Although the storage efficiency of PFID-files is good, the current eID card has not enough free storage available to accomodate several domains. Hence, a smart card with more persistent storage will be necessary.

The privacy of the card holder can even further be enhanced by using anonymous credentials.

Acknowledgement

This research is partially funded by the Interuniversity Attraction Poles Programme Belgian State, Belgian Science Policy and the Research Fund K.U.Leuven, the IWT-SBO project (ADAPID) ”Advanced Applications for Electronic Identity Cards in Flanders” and the IWT-TeTra project (e-IDea) ”Development of secure applications with the Belgian electronic identity card”.

References

1. D. De Cock, C. Wolf, B. Preneel, in *Sicherheit 2005: Sicherheit - Schutz und Zuverlässigkeit, Beiträge der 3rd Jahrestagung des Fachbereichs Sicherheit der Gesellschaft für Informatik e.v. (GI)*, *Lecture Notes in Informatics (LNI)*, vol. LNI P-77, ed. by J. Dittmann (Bonner Köllen Verlag, Magdeburg,DE, 2006), *Lecture Notes in Informatics (LNI)*, vol. LNI P-77, pp. 298–301
2. D. De Cock, K. Wouters, B. Preneel, in *European PKI Workshop: Research and Applications*, *Lecture Notes in Computer Science*, vol. 3093, ed. by S. Gritzalis, S.K. Katsikas, J. Lopez (Springer-Verlag, Samos Island,GR, 2004), *Lecture Notes in Computer Science*, vol. 3093, pp. 1–13
3. M. Stern, *Belgian Electronic Identity Card content*. Zetes, CSC, 2nd edn. (2003)
4. G. Ramlot, *eID Hierarchy and Certificate Profiles*. Zetes, Certipost, 3rd edn. (2006)
5. Belgian certificate revocation list. `http://status.eid.belgium.be/`
6. P. Andries, *eID Middleware Architecture Document*. Zetes, 1st edn. (2003)
7. J. Rommelaere, *Belgian Electronic Identity Card Middleware Programmers Guide*. Zetes, 1st edn. (2003)
8. B. De Decker, V. Naessens, J. Lapon, P. Verhaeghe, Kritische beoordeling van het gebruik van de Belgische eID kaart. Report CW524 (2008)
9. Advanced security mechanisms for machine readable travel documents extended access control (eac) and password authenticated connection establishment (pace). Technical Guideline TR-03110 (2008)
10. SSL Authentication Reverse Proxy. `http://eid.belgium.be/nl/Achtergrondinfo/De_eID_technisch/`