

A Mobile and Reliable Anonymous ePoll Infrastructure

Pieter Verhaeghe¹, Kristof Verslype¹, Jorn Lapon²,
Vincent Naessens², and Bart De Decker¹

¹ Katholieke Universiteit Leuven, Department of Computer Science,
Celestijnenlaan 200A, 3001 Heverlee, Belgium

`firstname.lastname@cs.kuleuven.be`

² Katholieke Hogeschool Sint-Lieven, Department of Industrial Engineering
Gebroeders Desmetstraat 1, 9000 Ghent, Belgium

`firstname.lastname@kahosl.be`

Abstract. This paper illustrates and scans the limits of the use of anonymous credentials (e.g. Idemix) on smart phones to preserve the user's privacy. A prototypical application with strong privacy requirements, ePoll, will be presented in detail. To ease the implementation of such applications, a specialized identity management framework has been developed. A first prototype of the ePoll application was built for workstations. Later it was ported to a smart phone to evaluate the performance of anonymous credential protocols in this setting.

Keywords: ePoll, mobility, anonymous credentials.

1 Introduction

Mobile Internet becomes increasingly popular. The recent generation mobile devices are capable of using third generation GSM (3G) networks, Bluetooth and WiFi. Some phones even have GPS. This connectivity allows for a broad range of new services. Developers can implement their own mobile application to further enrich the user's experience. However, as a mobile device is more used for personal transactions than an average PC, it is more attractive for service providers to make extended profiles of their clients. Moreover, it is easier to track the location of a mobile phone than of a static PC. Hence, the need for protection of the user's privacy increases as mobile phones will be used frequently in online transactions. To illustrate the use of privacy-enhancing technologies on a mobile device, a prototypical application with strong privacy requirements, namely an electronic poll application (ePoll), has been designed and implemented.

Opinion polls support participation in the democratic decision-making process. In poll schemes, the poll organizer is often interested in evaluating the differences in opinions between the distinct groups of voters (e.g. female voters may prefer the first option, while the male voters prefer the second one). Identification of the

voter¹ can provide this information, but is not always reliable (it could have been made up) or identification can discourage voters to participate at all. A reliable and flexible poll system based on anonymous credentials is, therefore, more appropriate. It guarantees the *voter's anonymity* but also the *correctness of disclosed personal information* (such as gender or age group). The current ePoll system bootstraps (during registration) with the Belgian eID card. However, any other unique identification means can be used during registration. The voter will then receive an anonymous voting credential which can be used in several polls; each voter can vote only once per poll and no votes of the same individual can be linked. The voter may have to prove certain properties (e.g. gender = female and age > 18) in case the set of allowed voters is restricted (in this case to female adults) and she may disclose extra attributes (e.g. the ZIP code of her residence) to allow for more varied statistics (e.g. deduce the difference of preferences between residents from Catania or those from Palermo).

The paper is structured as follows. After a description of the underlying technologies in Section 2, the ePoll protocol is presented in detail in Section 3. Section 4 discusses some implementation details of the prototype and the results of porting this protocol to a smart phone. Section 5 concludes the paper.

2 Technologies

2.1 Anonymous Credentials

Anonymous credentials [1,2,3,4,5] allow for anonymous yet accountable transactions between users and organizations. Also, selective disclosure of attributes embedded in the credential gives the user the possibility to prove only the information strictly required for the transaction and keep the rest hidden. Different operations are available. All these operations are interactive protocols:

- $U \rightleftharpoons I: Cred \leftarrow issueCredential(Atts, Properties)$

The issuer I issues a credential $Cred$ to the user U . The credential will contain attributes $Atts$. One of the attributes can be a pseudonym of the user U known by issuer I . The other attributes can be chosen by U , by I or by both parties. In the first and the last case, only U knows the final value of the attribute. The properties can specify the validity period and the number of times the credential can be shown. Some credentials can only be shown once or a limited number of times; if the limit is exceeded, the credential issuer will be able to detect this and eventually identify the culprit.

- $U \rightleftharpoons O: Trscr \leftarrow showCredential(Cred, Properties)\{Msg\}$

The user U shows his credential $Cred$ to organization O . (Note that the credential may have been issued by another organisation I). Additionally, U reveals some $Properties$, involving public values, attributes kept in the credential

¹ This is the case in most of the paper-based polls, where voters have to write down their name and address and possibly some extra information such as their age or gender.

Cred and/or attributes already revealed in previous credential shows and/or commitments. Optionally, a message *Msg* can be signed during the credential show, which provably links the message to the credential show. Features such as conditional anonymity and limited-show are not used in the ePoll application and, hence, will not be explained here. *Trscr* consists of all the messages exchanged during the show-protocol and can be used to resolve disputes. An example of a predicate proved by *U* could be that he is an adult (i.e. $\text{Cred.age} \geq 18$) and a Belgian citizen (i.e. $\text{Cred.citizenship} = \text{"BE"}$). In all credential systems, a “credential show” cannot be linked to the issuing of the shown credential (except when unique attributes are revealed during the show).

2.2 Commitments

A commitment can be seen as the digital analogue of a “non-transparent sealed envelope”. It enables a committer to hide a set of attributes (i.e. non-transparency property), while at the same time preventing him from changing these values after commitment (i.e. sealed property). The committing party can prove properties of the attributes embedded in the commitment.

- $(Com, OpenInfo) \leftarrow \text{commit}(Attribute(s))$

A new commitment *Com* is generated as well as a secret key *OpenInfo* containing, among others, the attributes embedded in *Com*. This key can be used to prove properties about the attributes.

- $\mathcal{P} \rightarrow \mathcal{V}: \text{comProve}(Com, \text{pred}(Attribute(s)))$

The public input to this protocol is both a commitment *Com* and a boolean predicate *pred* that defines *com*’s attribute(s). If \mathcal{V} accepts the proof, \mathcal{V} is convinced that \mathcal{P} knows *OpenInfo* belongs to *Com*, and that *Com*’s attributes satisfy predicate *pred*. Note that *comProve* is an interactive protocol and that \mathcal{P} needs *OpenInfo* to succeed.

3 Protocol

This section discusses the ePoll setting and protocols needed to build a flexible poll system as described in the introduction.

3.1 Requirements

The requirements are split in user requirements and poll requirements.

Poll requirements

- *O1. Uniqueness.* A user can express his opinion in a poll only once.
- *O2. Eligibility.* A poll may wish to address only a subset of the potential signers; therefore, the signer will have to prove that she belongs to this subset.

- *O3. Statistics.* On request of the poll organizer, the user may release additional personal properties such as her age group and/or her gender. However, this disclosure is optional. This additional information allows the poll organizer to derive more significant statistics (e.g. opinion per age group or per gender, etc.).

User requirements

- *U1. Right to participate.* Each citizen must have the right to express her opinion in a poll if she belongs to the targeted population.
- *U2. Accuracy.* Everyone can verify the correctness of the poll results. This implies that an attacker cannot forge the poll results and that each signer can verify whether her opinion is included in the final results.
- *U3. Anonymity.* Participants are anonymous. Moreover, opinions of the same user cannot be linked to an individual or to each other.

3.2 Roles

Different roles can be identified. A **user** U needs a *user credential* $cred_U$ issued by an **issuer** I in order to participate in a poll. A **poll organizer** O is the entity taking the initiative to launch a poll. Such a poll must be published by a **poll server** S . If the poll organizer has the appropriate infrastructure, it is possible that O coincides with S , but in most cases, S will be a separate entity which can run multiple polls of different poll organizers, potentially next to other services. A **central authority** A approves and certifies a poll by issuing a *poll certificate* $cert_{poll}$ to the poll server. Each user's signature for a particular poll is published by the poll server S , allowing everyone to verify the correctness of the poll. Each signer receives a *receipt* as a result of signing the poll. This receipt can be presented to the poll authority in case abuse is diagnosed (e.g. the voter cannot find her signature in the poll's list of signatures). The authority can then revoke the poll certificate and possibly nullify the poll. Note that A may coincide with I .

3.3 Design

This section discusses the different protocols required in the poll infrastructure. The following assumptions are made: (1) each issuer I has an X.509 certificate $cert_I$ which contains security parameters required for the random number generation protocol (cfr. next par.) (2) each poll authority A and each poll organizer O have an X.509 certificate to authenticate towards their clients and (3) each user U has some means to uniquely authenticate to I (physical appearance, eID card, etc.).

We further made the following trust assumptions. A credential issuer is trusted to only issue credentials with correct attribute values. No additional trust is required in the credential issuer. For instance, he cannot vote in the credential

holder's name. A poll authority is trusted to generate the poll's one-way function without hidden trapdoors. For instance, if g_1 is base for poll P_1 , and g_2 is base for poll P_2 , the poll authority does not know $\log_{g_1}(g_2) = a$ (see further). Hence, g_2 may not be chosen as g_1^a ; otherwise, the poll authority is able to link votes by the same user for P_1 and P_2 . If more than 1 poll authority is used, the set of poll authorities are trusted not to collude so that a subset of them do not know $\log_{g_1}(g_2)$ for any two different polls.

Retrieving a user credential. User U and issuer I mutually authenticate. I uses his X.509 certificate and U uses whatever authentication means that is available and acceptable. Next, U and I run the interactive secure random number generation protocol, based on the security parameters in $cert_I$, resulting in a secret random value r_U for U and a commitment to r_U for I . I issues to U an anonymous credential $cred_U$ containing the secret r_U and a set of user attributes which are necessary to prove membership of a poll's voter set or to disclose additional information (e.g. date of birth, ZIP code, gender, etc).

Setting up an ePoll. The poll organizer O composes and signs the poll's specification $spec$. It specifies the alternatives that can be selected by a signer, as well as the required and desired voter's properties to disclose. The $spec$ is sent (via the poll server S) to the poll authority A for approval. As a (positive) result, A will issue a specific poll certificate $cert_{poll}$, which contains the $spec$, as well as a freshly generated one-way function with which a voter generates the poll's specific unique pseudonym (cfr. next par.). A single poll server can organize many polls concurrently.

Signing a poll. (See table 1). The poll server S authenticates twice to the voter U (first using its poll server certificate (1) and then using the poll specific certificate $cert_{poll}$ (2)) to assure the user that it is an accredited poll server that is authorized to organize that poll. Now U selects the alternative of her choice (3) and the properties she is willing to disclose (4). Then, the user generates her poll specific pseudonym nym (5) based on the user's secret r_U embedded in her credential and using the one-way function which is specified in the poll specific certificate $cert_{poll}$. Next, the user signs the alternative of her choice and the poll's identifier with her anonymous credential and proves the selected personal properties and the correct generation of nym (6). The resulting anonymous signature (*proof*) is assigned a number (*voteNb*) (7), is countersigned by S (8) and finally published (9). A receipt is generated for the user (9-12), allowing her to file a complaint when her record has been wrongfully removed from the poll's results. The receipt is a signed verifiable encryption of the poll specific nym , the number and the hashed proof. S proves to U that the correct values are encrypted. This receipt is kept by U in case of a later dispute (14).

Poll closure. A poll closes when the poll certificate expires. The poll server then has to sign the complete set of all poll records and submits this signature to the poll authority A and to the poll organizer O .

Table 1. Signing a poll in the ePoll system
$$U \stackrel{\Leftarrow}{\Leftarrow} P: \text{signPoll}(cert_R, cert_{poll}; cred_U; SK_{poll})$$

- (1) $U \leftarrow S$: `authenticate(SK_S ; $cert_s$)`
- (2) $U \leftarrow S$: `authenticate(SK_{poll} ; $cert_{poll}$)`
- (3) U : `choice := choose($cert_{poll}.alternatives$)`
- (4) U : `props := setProps($cert_{poll}.showProps$)`
- (5) $U \rightarrow S$: `nym := $cert_{poll}.f(cred_U.r)$`
- (6) $U \stackrel{\Leftarrow}{\Leftarrow} S$: `proof := showCred($cred_U$; $nym = cert_{poll}.f(cred_U.r)$
 \wedge props){choice, $cert_{poll}.id$ }`
- (7) $U \leftarrow S$: `voteNb := getVoteNb()`
- (8) S : `sigpoll := sign(SK_{poll}^{sig} , [$voteNb$, nym , $props$, $choice$, $proof$])`
- (9) S : `publish($voteNb$, nym , $props$, $proof$, $choice$, sig_{poll})`
- (10) S : `hash := Hash($proof$)`
- (11) $U \leftarrow S$: `v := $venc_A(nym, voteNb, hash)$`
- (12) $U \stackrel{\Leftarrow}{\Leftarrow} S$: `PK{(): $v.nym = nym \wedge v.voteNb = voteNb \wedge v.hash = hash$ }`
- (13) $U \leftarrow S$: `sigv := sign(SK_{poll}^{sig} , v)`
- (14) U : `store(v , sig_v)`

Poll verification. The user can easily verify his own poll record by regenerating his poll specific nym and by downloading and verifying the corresponding record. If the record has disappeared, the receipt (v, sig_v) is submitted to the poll authority A who can decide to intervene, e.g. by revoking the poll server's certificate and nullifying the results. It is also possible to verify a random selection of the records or all the records.

3.4 Evaluation

The protocols will be evaluated against the requirements of section 3.1.

- **O1.** For each poll, U is only known under a unique per-poll nym . If that nym has already cast a vote (i.e. a record with that nym is already stored in the poll database), the new vote is rejected. It is possible to adapt the protocol to allow for modification of votes; in this case, the voter first cancels her previous vote.
- **O2, O3, U1.** U will need to prove certain personal properties (if the voter set is restricted); she can decide whether or not to disclose additional properties.
- **U2.** U can detect and prove whether her vote was deleted by generating the poll specific nym again and checking if that nym is included in the poll. If this is not the case, the user's receipt – which is a proof – is sent to A . It is not possible to forge someone's vote without possessing the victim's poll credential. Anyone can verify the correctness of the votes by verifying the publicly available records. $voteNb$ is incremented for each new record; S can thus only delete the most recent votes, which can be quite visible and is always provable by the record owners (via the receipt). It is not

possible to vote multiple times (with different nyms) for the same poll, unless a compromised I issued multiple credentials with different secret random values to the same voter. Hence, I must be a trustworthy party. S is unable to transfer a vote from one poll to another poll since the signature contains a proof that the poll specific nym is generated using the poll's one-way function and the user's secret r_U .

- **U3.** If the disclosed attributes are not taken into account, the system is perfectly anonymous since per poll pseudonyms of the same user are unlinkable. A prerequisite is that different one-way functions f do not introduce linkabilities (e.g. there is a linkability if $f_1(x) = g_1^x \bmod n$ and $f_2(x) = g_2^x \bmod n$ and $\log_{g_1}(g_2)$ is known). Moreover, privacy is preserved even when multiple parties collude since the input to calculate the nym is only known to U . Since the receipt looks like a signed random number, it is possible for people who did not vote to request from the poll server a fake receipt. This eliminates the possibility to use a receipt as a proof of participation. Coercion and vote selling cannot be prevented, since a signer can always (be forced to) regenerate her per poll nym and prove possession thereof to anyone, which also proves ownership of a poll record.

3.5 Extensions

Change of opinion. The `signPoll`-protocol can be extended to allow a voter to change his vote. If the user wants to change his vote, she has to sign a `cancel` message. Hence, the ePoll server has evidence that the previous vote may be canceled.

Immediate or delayed record publication. The `signPoll`-protocol can be modified to hide the chosen alternatives until the poll is closed. In this case, the voter signs a verifiable encryption of the selected alternative instead of the alternative itself and provides a proof that a valid alternative has been submitted. When the poll closes, the poll organizer publishes the key to decrypt these opinions. A secret sharing scheme can be used to share this key between multiple parties (e.g. poll organizer and poll authority), in order to avoid that one entity has access to the results before closure time.

Recovery, Renewal and Theft. When the user loses her poll credential or when it is compromised, the old credential needs to be revoked and a new credential needs to be issued. However, this credential should contain the same secret value r_U of the previous credential. Otherwise, the uniqueness property is no longer satisfied. Therefore, the r_U should be securely backed up by the user. Also, the issuer I should keep a copy of a commitment of r_U . Then, I can issue a new credential with the same r_U -value. Note that this technique can also be applied to allow the user to have multiple poll credentials at the same time (e.g. on mobile phone, PDA, and desktop computer).

When r_U or its commitment is permanently lost, a new r_U^* needs to be generated. All credentials with the old r_U -value will be revoked and a new credential

with r_U^* will be issued. However, to enforce the uniqueness property, the user will only be allowed to vote for polls that begin after the credential's issuing time.

Questionnaire or quiz. The solution can easily be extended to a questionnaire. Here, several multiple-choice questions must be answered and anonymously signed with the credential. Also, the results do not necessarily need to be published.

Since the presented solution is based on anonymous credentials, it is possible to make each vote deanonymizable if the voter agrees. Hence, the presented solution can be used to organize quizzes. Only the winners of the quiz will eventually be deanonymized.

3.6 Scalability

Multiple issuers. Based on a secret in the user's credential, a user can generate a per-poll pseudonym and prove possession thereof. This implies that multiple issuers are possible in the system if one of the following conditions is fulfilled: (1) the different issuers each serve another partition of the population (e.g. at the level of nations), (2) each poll specifies exactly one issuer that must have issued the credentials that can be used to sign the poll or (3) the issuer is able to check whether the user has already registered her secret number r_U with a TTP, which has to keep a commitment to r_U . If this is the case, the commitment is retrieved by the issuer and used to embed r_U in the new credential.

Multiple poll certifiers or poll servers. Multiple poll certifiers can be allowed; however, it does not make sense to have multiple poll certifiers for a single poll. Multiple poll servers S can serve a single poll if there is a shared synchronized clock. Now, votes will also be timestamped. If a user signs the same poll multiple times, only the most recent record will be retained in the results and the other votes will be discarded. Each S needs its own poll certificate for the same poll, all of them will contain the same one-way function and poll specification.

Poll verification. In order to verify large polls, it suffices to randomly select a subset of the valid poll records (i.e. a sample) and compare the resulting statistics based on the sample match to the results given by the poll server. If H_0 is the published statistical poll data and μ is the statistical data based on the sample, the verifier has to calculate $Pr[\mu_{\leq} | H_0]$, i.e. the probability to obtain sample results that are at least as extreme as μ if H_0 is correct. If this probability is too small for a sufficiently large sample, the published poll results are considered as incorrect and a complaint can be filed with A .

4 Implementation

In this section, the implementation of the ePoll application is discussed. First, the framework [6] is presented that is used to build the application. Next, the results of the prototype are shown.

Identity Framework. Applying privacy enhancing technologies is not trivial and each technology has its own interface and peculiarities. Hence, it will cost an application developer a lot of effort and time to make his applications privacy friendly, which increases the probability that (1) the privacy concerns will be omitted or (2) if they are not omitted, the privacy is still compromised due to a bad use of the privacy preserving technologies. Moreover, the user is often not able to keep track of set of personal information has been disclosed to whom.

Our framework offers components to control in a fine-grained way the data that has been disclosed to each party. A uniform interface manages (anonymous) connections. Another uniform interface offers the application developer the possibility to use alternative credential technologies. This makes it very easy to switch to a more efficient or a more privacy-preserving technology.

The framework mainly consists of a set of managers and corresponding handler classes. Currently, the following managers exist in the framework: `Connection-`, `Credential-`, `Persistence-` and `PrivacyMaganer` which respectively take care of connections, credentials, storage of data and privacy preferences. Each manager has a list of corresponding handler instances which implement a specific technology (e.g. X.509 certificates in a `X509CredentialHandler`, TCP socket connections in a `TCPConnectionHandler`). The implementation of the handler components can be packaged in providers. These providers can be plugged in the framework. Hence, a developer can fine-tune the framework towards the technologies he wants to apply in his application. The framework has already been tested by building prototypes in multiple domains (e.g. eTicketing, ePoll, eHealth, eAuction, etc.).

Prototype. A prototype of the ePoll protocol is implemented using the framework that is described in the previous section. The prototype uses the Belgian eID card [7] to enforce that every person gets exactly one ePoll credential. Both the framework and Idemix are implemented in Java. Java applets are used in a website. They run the registration and voting protocols. A Java web server (Tomcat) is used at the server side. Figure 1 shows a screenshot of the vote form on the ePoll website. The screenshot demonstrates how the user can select the option to sign and which extra properties will be disclosed. The first (grayed out) property (*age* > 18) must be proven. Users can optionally prove to which age group they belong.

Smart phones based on the Google Android operating system are able the run the Java code of the framework and Idemix. Most standard Java classes are available on Dalvik, a virtual machine which runs the Java platform on Android mobile devices. Hence, it was quite easy to port the ePoll application to a smart phone. Only the user interface needed to be redesigned because Android uses other classes for the GUI than Java applets. There is no smart card reader available to read out the Belgian eID card. Hence, the registration part of the protocol is executed on a PC with a smart card reader. If an alternative registration method is selected that can be used on smart phones (e.g. authentication with the SIM card), then registration can be performed directly on the smart phone. The retrieved anonymous credential is stored in a local file which can be

1. Fill in the poll

What is your general impression of the workshop?

- Excellent
- Good
- Fair
- Poor

Did you feel the length of workshop presentations was too long, just right, or too short?

- Too long
- Just right
- Too short

2. Load Credential

3. Required and optional proofs

- age > 18
- age in [18 - 25 [

Fig. 1. Screenshot of the vote form on the ePoll website

transferred to the smart phone. No implementation modifications are needed at the server side to support the ePoll protocol on smart phones since the ePoll protocol and the available connection technology in the framework (TCP sockets) are unaltered. Moreover, the ePoll application on the mobile device can communicate over WiFi, 3G, GPRS networks, etc.

Performance. Signing a poll, without revealing attribute values (e.g. gender = “female”), using a smart phone takes about 5 seconds. This test was done on a Motorola Milestone with 256 MB RAM and a Texas Instruments OMAP3430 ARM-processor which has a clock speed of up to 550 MHz. The same protocol on a workstation (with an Intel Core 2 Duo T7100 1.8 GHz processor) takes about 1 second.

Some more performance results are available in Table 2. The execution time was measured on the mobile phone (i.e. the client side). The time includes the

generation of the signature and proofs on the mobile, communication between the mobile and the server over a wireless connection (WiFi) and the verification of the signature and proofs on the server side. The tests were done with an Idemix credential that can contain up to 15 attributes.

If more attribute values are revealed during the `showCred` method, less exponentiations need to be calculated. Hence, the execution time decreases as more attributes are revealed. When `greaterThan`-proofs are used (e.g. `age > 18`), a lot more exponentiations are calculated in the Idemix proof. The additional time needed for a `greaterThan`-proof is around 4 seconds as can be derived from Table 2.

Table 2. Execution times of the `showCred` method of Idemix

Proof	Execution time
No additional proofs	5.102 seconds
1 attribute revealed	5.023 seconds
3 attributes revealed	4.934 seconds
5 attributes revealed	4.824 seconds
7 attributes revealed	4.663 seconds
1 <code>greaterThan</code> proof	8.884 seconds
2 <code>greaterThan</code> proofs	13.599 seconds
3 <code>greaterThan</code> proofs	16.934 seconds

5 Related Work

Many Voting Protocols are published [8,9,10]. An important feature of these protocols is the behavior repudiability requirement, which is less relevant in opinion poll systems. Less attention has been devoted to ePoll and e-Petition systems. Recently, an e-Petition based on the Belgian eID card and Idemix was developed [10], which presents similarities with our ePoll solution. However, it does not allow to renew lost or expired user credentials, nor does it allow the user to have different credentials on multiple devices. Our solution is ported to mobile devices with acceptable performance.

An Idemix based anonymous reviewing system [11] has similar requirements: a reviewer can anonymously review a paper, but only once. However, after each review, the reviewer's credential must be replaced, making the construction less efficient and flexible.

6 Conclusions and Future Work

This paper presented an application with advanced privacy and anonymity requirements. The ePoll protocols are described in detail. Individuals can only vote once per poll and votes of the same user (for different polls) can not be

linked. Moreover, the set of acceptable voters can be restricted based on personal properties. Also, a poll organizer can request the voters to disclose additional properties in order to extract more significant poll statistics. The disclosed properties are guaranteed to be true (i.e. they are certified by the issuer of the voting credential). A concrete implementation of the ePoll application has been realized using Idemix anonymous credentials. We used an identity management framework which helps the application developer by hiding the low level details of the different PET technologies. Finally, the application has been ported to smart phones (with Google Android operating system). Porting was easy, since the identity management framework can run unmodified on the mobile device, and only the GUI needed some adaptations. Tests showed that users can participate in ePolls using their mobile devices. Also, the performance – although slower than on a PC – was still acceptable.

References

1. Brands, S.: Technical overview of digital credentials (1999), <http://citeseer.ist.psu.edu/brands02technical.html>
2. Brands, S.: Rethinking Public Key Infrastructures and Digital Certificates: Building in Privacy. MIT Press, Cambridge (2000)
3. Camenisch, J., Van Herreweghen, E.: Design and implementation of the idemix anonymous credential system (2002), <http://citeseer.ist.psu.edu/camenisch02design.html>
4. Camenisch, J., Lysyanskaya, A.: An Efficient System for Non-transferable Anonymous Credentials with Optional Anonymity Revocation. In: Pfitzmann, B. (ed.) EUROCRYPT 2001. LNCS, vol. 2045, pp. 93–118. Springer, Heidelberg (2001)
5. Chaum, D.: Security without identification: transaction systems to make big brother obsolete. ACM Commun. 28(10), 1030–1044 (1985)
6. De Decker, B., Lapon, J., Layouni, M., Mannadiar, R., Naessens, V., Vangheluwe, H., Verhaeghe, P., Verslype, K.: Adapid deliverable D12: Framework II (2009), https://www.cosic.esat.kuleuven.be/adapid/docs/Adapid_D12.pdf
7. The belgian electronic identity card, <http://eid.belgium.be/>
8. Kiayias, A.: An internet voting system supporting user privacy (2006), <http://www.scientificcommons.org/42582861>
9. Adida, B., de Marneffe, O., Pereira, O., Quisquater, J.-J.: Electing a University President Using Open-Audit Voting: Analysis of Real-World Use of Helios. In: Jefferson, T.M.D., Hall, J.L. (eds.) Electronic Voting Technology Workshop/Workshop on Trustworthy Elections. USENIX (2009)
10. Diaz, C., Dekeyser, H., Kohlweiss, M., Nigusse, G.: Privacy preserving electronic petitions (2008), <http://www.cosic.esat.kuleuven.be/publications/article-1053.pdf>
11. Naessens, V., Demuyne, L., De Decker, B.: A fair anonymous submission and review system. In: Leitold, H., Markatos, E.P. (eds.) CMS 2006. LNCS, vol. 4237, pp. 43–53. Springer, Heidelberg (2006)