

PetAnon: A Privacy-Preserving e-Petition System Based on Idemix

*Kristof Verslype Jorn Lapon Pieter Verhaeghe
Vincent Naessens Bart De Decker*

Report CW522, October 2008



Katholieke Universiteit Leuven
Department of Computer Science
Celestijnenlaan 200A – B-3001 Heverlee (Belgium)

PetAnon: A Privacy-Preserving e-Petition System Based on Idemix

Kristof Verslype Jorn Lapon Pieter Verhaeghe
Vincent Naessens Bart De Decker

Report CW522, October 2008

Department of Computer Science, K.U.Leuven

Abstract

Electronic petition systems support participation in the democratic decision-making process. However, current petition systems often have serious drawbacks in reliability and anonymity. This paper presents PetAnon, a privacy-preserving petition system that tries to tackle the shortcomings of existing systems. A proof-of-concept implementation is presented that uses the Belgian eID card in a bootstrap procedure, after which users are allowed to sign petitions anonymously, while certain constraints may be imposed.

Keywords : Petition, Anonymity, Privacy, Security, Electronic Identity
MSC : Primary : D.4.6, Secondary : K.4.1.

PetAnon: A Privacy-Preserving e-Petition System Based on Idemix

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

October 2008

Abstract

Electronic petition systems support participation in the democratic decision-making process. However, current petition systems often have serious drawbacks in reliability and anonymity. This paper presents PetAnon, a privacy-preserving petition system that tries to tackle the shortcomings of existing systems. A proof-of-concept implementation is presented that uses the Belgian eID card in a bootstrap procedure, after which users are allowed to sign petitions anonymously, while certain constraints may be imposed.

Keywords: *Petition, Anonymity, Privacy, Security, Electronic Identity*

1 Introduction

In a petition, opinions of people are collected and processed. In paper-based petitions the collection and processing takes a lot of time and effort. Electronic petition systems (e-Petition), however, offer several benefits with respect to the paper-based petitions. e-Petitions enable users to sign petitions anywhere at any time and now reach wider sections of society. Moreover, automatic processing of opinions is faster and less error-prone.

On the other hand, electronic petition systems introduce new problems. Some systems may return unreliable results as for instance a user may sign a petition more than once. Other systems use personal information to prevent multiple signing. However, these systems are not privacy friendly, which is important in petitions where a political, religious, etc. opinion is expressed.

This paper presents PetAnon, a privacy-preserving petition system using Idemix anonymous credentials. PetAnon combines good privacy properties with reliable results.

PetAnon allows petition organizers to provide potential signers with multiple choices. Today, many petitions only have one version: 'in favour'. PetAnon allows to address only a subset of the population (e.g. all women older than 30) and invites the signers to reveal some personal properties such as gender and zip code. Hence, additional statistical information can be collected by the petition organizer. However, this information does not compromise the signer's anonymity. Everyone can verify the correctness of the petition and the petition signer can withdraw or change his vote.

The paper is structured as follows. Section 2 introduces the main technologies used. Section 3 presents the functional and threat model. Sections 4 and 5 discuss the protocols and give an evaluation. Section 6 generalises a technique used in the protocols. Section 7 discusses the implementation and we conclude with a summary and future work.

2 Technologies

This section discusses the basic technologies used in PetAnon, namely anonymous credentials, commitment schemes and provable one-way functions.

2.1 Anonymous Credentials in Idemix

Anonymous credentials [6, 2] allow anonymous yet accountable transactions between users and organisations. Moreover, selective disclosure allows the user to reveal only the information necessary for the purpose. One of the anonymous credential systems currently being developed is Idemix [5, 4]. Except when identifiable attributes are revealed, multiple Idemix credential shows are unlinkable.

The relevant functions for this paper are:

- $U \xrightarrow{\leftarrow} I : cred \leftarrow \text{issueCred}(\text{attributes})$. Issuer I issues a credential to U . The credential contains *attributes*, not necessarily known by I .
- $U \xrightarrow{\leftarrow} V : proof \leftarrow \text{showCred}(cred, \text{properties})\{msg\}$. U proves to verifier V possession of a valid credential *cred*. U can selectively reveal credential attributes or properties thereof. U may decide to sign a message *msg* with his credential, creating a provable link between the proof and the message.

2.2 Commitment schemes

A commitment scheme [13, 8] allows a party to commit to a value, while keeping it secret. The commitment hides the value towards the verifier. Later, the committer can decide to open the commitment, showing the actual value to the verifier or prove properties of the committed value.

The following (simplified) methods are relevant in this paper:

- $(comm, openInfo) \leftarrow \text{commit}(\text{attribute})$. A new commitment containing a single attribute is generated as well as the opening info required to prove properties about the commitment. The attribute and *openInfo* are known to U .
- $U \rightarrow V : \text{prove}(\text{properties}, comm, openInfo)$. User U proves properties of the commitment to verifier V using the corresponding *openInfo*.

2.3 Provable one-way functions

We define a provable one-way function $out \leftarrow f(in)$ as a one-way function such that the party knowing in , can easily prove that he knows an in such that $out = f(in)$ in a zero-knowledge proof. Multiple arguments are possible.

3 Functional and Threat Model

This section first discusses the roles and interactions in PetAnon. We continue by summing up the assumptions and by presenting the attacker model and conclude with the relevant e-Petition requirements.

3.1 Roles and Interactions

We distinguish a single, central registration service R , which could be a government instance, a set of petition organizers $\{P_1, P_2, \dots, P_k\}$ and a set of users $\{U_1, U_2, \dots, U_n\}$. Each U and P has an identifying certificate; $cert_U$ (the Belgian eID card in PetAnon), respectively $cert_P$. $cert_U$ contains a user identifier and personal attributes such as gender and zip code. A user may possess multiple $cert_U$ s as long as they all contain the same identifier

We now elaborate on the formal interface of the different (inter)actions. The bold entities initiate the interaction by calling the method.

- $\mathbf{P} \rightleftharpoons R: (SK_{pet}; cert_{pet}) \leftarrow \text{registerPetition}(cert_P, name, desc, period, props_P, choices)$. A new petition must first be registered by P to R , resulting for P in a petition certificate $cert_{pet}$, which contains the petition's name,

description and validity period, the different vote options as well as some cryptographic information (cfr. section 4). An organizer can restrict the population of potential voters (based on voter’s properties) and/or can also invite voters to prove additional properties for statistics about the voting behavior (e.g. women vs. men). These properties are described in $cert_{pet}$, which is made public. P can organize multiple petitions.

- $\mathbf{U} \rightleftharpoons \mathbf{R}$: $cred_U \leftarrow registerVoter(cert_U)$. U requests as a potential voter a $cred_U$ credential, which allows him to sign petitions. The credential will include personal properties of U . These properties are either derived from $cert_U$ or from a database that can be linked to that user.

- $\mathbf{U} \rightleftharpoons \mathbf{P}$: $(receipt; record_{sign}) \leftarrow signPetition(cred_U, cert_{pet}, choice, props_U)$. U signs one version ($choice$) of a petition specified by $cert_{pet}$ using $cred_U$. $props_U$ specify the properties he must and is additionally willing to disclose. Both $choice$ and $props_U$ are derived from $cert_{pet}$. Signing a petition results in a new petition record $record_{sign}$ that is made publicly available by P . Each petition thus consists of petition records. U receives a $receipt$.

- $results \leftarrow countPetition(cert_{pet}, record_{sign}[])$. After retrieving the petition certificate and the petition records, the result can be computed. Only valid records are taken into account.

- \mathbf{U} : $true/false \leftarrow verifyOwnVote(cert_{pet}, receipt, record_{sign})$. After retrieving the petition’s $cert_{pet}$ and U ’s vote record, U can verify whether his vote is correctly stored and matches the receipt.

- $\mathbf{U} \rightleftharpoons \mathbf{O}$: $withdrawSignature(cred_U, name, receipt.voteNr)$. During the voting period, when the user has second thoughts, he can withdraw (and later possibly redo) his vote.

3.2 Security and Privacy assumptions

The following (non-trivial) assumptions are made.

- **Secure connections.** Connections preserve integrity and confidentiality. U , P and R authenticate before starting an interaction, but when U interacts with P (i.e. $signPetition()$ and $withdrawSignature()$), a different anonymous connection for each protocol execution guarantees unlinkability of actions and U ’s anonymity at network level.
- **User bound sign credentials.** A sign credential is bound to one U . In Idemix, sharing can be discouraged by including a valuable user secret in the credential, or by requiring the cooperation of a smart card.

- **Large anonymity set.** A sufficiently large user set guarantees the signer’s anonymity, even when personal properties are disclosed.

3.3 Attacker Model

We describe the attacker model against which PetAnon will be evaluated. Attacks can be grouped in petition forgery or vote linking.

Petition forgery by a malicious U (1), P (2) or R (3), coercion (4) or impersonation (5).

1. *Malicious U .* U signs the same petition multiple times by either obtaining multiple sign credentials, or by hiding for P that his sign credential has already been used.
2. *Malicious P .* P modifies, forges, or deletes records in a specific petition; P moves records to another petition; P convinces U to sign another petition; or P aborts the execution of the sign protocol.
3. *Malicious R .* A compromised R issues faked or modified sign credentials or petition certificates and can revoke petition certificates. Malicious Us , Ps and R can collude.
4. *Coercion.* U is forced to express a particular opinion.
5. *Impersonation.* An entity fools P by performing the `signPetition()` or `withdrawSignature()` method in the name of someone else; i.e. while not possessing the corresponding sign credential.

Signature linking to the signer’s id (1) or to another signature (2).

1. *To the signer’s id.* A set of colluding entities can link a sign record to a specific user.
2. *To another signature.* P can link votes of the same U on different petitions to each other. Because each P can manage many petitions, there is no difference with a set of colluding Ps .

3.4 Requirements

Based on the interactions and the attack model, the PetAnon requirements are discussed. They are grouped into security and privacy requirements.

Security requirements

- S1 A user can sign a certain petition only once.

- S2 A petition can address only a subset of the potential signers; therefore, the signer may be required to prove that he belongs to this subset.
- S3 Each citizen must have the right to sign a petition if he belongs to the targetted population, independent of his preferred voting choice.
- S4 During the voting period, a user can revoke, and potentially change, a prior vote.
- S5 A user can verify the correct inclusion of his signature in the petition database.
- S6 No one is able to change, modify or delete petition records. Moreover, everyone can verify the correctness of the petition results.

Privacy requirements

- P2 Petition signatures cannot be linked to a user. Moreover, signatures of the same user for different petitions cannot be linked.
- P3 On request of the petition organizer, the user may or may not release additional personal properties such as age interval and gender.

4 Protocol

The implementation of the methods in section 3 are discussed in more detail.

4.1 Register a petition

In `registerPetition()`, P provides R with the petition specification. The cryptographic data is a unique provable one-way function $y \leftarrow f_{pet}(x_1, x_2)$, determined by R , with two attributes. We implemented this function using the discrete logarithm assumption: $y \leftarrow g^{x_1} h^{x_2} \bmod n$, where n and the description of the groups G and H are public parameters of R and thus the same for each petition. For every petition two bases $g \in G$ and $h \in H$ are chosen by R and included in $cert_{pet}$. Before issuing $cert_{pet}$, R needs to check the uniqueness of the petition's name.

4.2 Get a petition sign credential

In `registerVoter()`, U authenticates to R using his $cert_U$ (step 1 of protocol 1.a), which reveals all the personal data contained in that $cert_U$, including U 's identifier id_U and personal attributes.

Every citizen is only allowed to have one sign credential. This is first checked by R . If the user did not register before (2a), the user generates

a secure random number (2a.1), puts it in a commitment (2a.2), which is sent to R , and proves that he knows the committed value (2a.3). U can also prove other properties such as the length of the committed value to R . R also generates a random value (2a.5). These two random values will be used to prevent signing the same petition multiple times, as we will explain later. The opening info of the commitment is encrypted using an integrity preserving symmetric encryption scheme (2a.4) and a user secret (i.e. the symmetric key). The resulting cipher c is sent to and stored by R .

U will need a new sign credential in case of loss or expiry of the previous one. Then (2b), the two random values (one encrypted) are retrieved from R 's storage and will be reused (2b.1). The user receives c from R and decrypts it, resulting in the commitment opening info which is required for issuance of a new credential.

Now, the sign credential can be issued (3). It contains the two random values and a subset of the attributes (or properties thereof) that were extracted by R from $cert_U$ or from a database that is linked to the user's identity. Note that R never gets hold of U 's secure random number. The user needs to have the opening info corresponding to the commitment during the issuance protocol.

Finally, U stores the credential (3), and R the commitment, the second random number, and U 's identifier (4). This will allow R to check whether a user already has been issued a sign credential and to issue new ones.

4.3 Signing a petition

In `signPetition()`, with the petition's one-way function and the two random values contained in the vote credential, U generates his petition specific nym (1), and sends it to P , together with the description of the personal properties $props_U$ U will reveal and his vote *choice* (2).

The interactive Idemix show protocol is executed (3): U proves the selected properties, as well as that the petition specific nym for that user is correctly formed based on the random values contained in the credential. Also the user's vote choice is signed anonymously, together with a timestamp whose necessity will be explained later.

If the owner of the nym has not yet signed the petition (4), the protocol continues by generating a vote number. This is a reference to the petition record that is being generated and is the previous vote number incremented by 1. The vote number, the hash of the proof and the U 's nym are signed with the petition secret key and stored by P together with that signature. The resulting record is made public. The signature is sent to and stored by

U , together with the vote number and the hash of the proof. This signed triplet is in fact U 's receipt.

4.4 Withdrawing a signature.

In `withdrawSignature()`, U proves to P ownership of the corresponding petition specific nym and thereby signs a message, stating that he wants to withdraw his vote: $\mathbf{U} \rightleftharpoons \mathbf{O}$: $withdrawProof \leftarrow \text{showCred}(cred_U, nym\ cred_U)\{\text{"withdraw: " + } voteNr + \text{"; " + } timestamp\}$. Steps 6-8 in protocol 1.b do not change (only "*withdrawProof*" replaces "*proof*"). Thus, a record, stating that the U withdrew his vote, is generated, replaces the previous proof and is made public.

4.5 Counting the petition results.

`countPetition()` consists of verifying for each record the proof and the respective signature (with P 's $cert_{pet}$).

4.6 Verifying your vote.

`verifyOwnVote()` can be done by U after having retrieved his record which was signed by P and whose $voteNr$ is also included in U 's receipt.

Notes. Credentials can be revoked, but no implementation supports this at the moment. This could be useful in case U 's $cred_U$ was stolen. Short validity periods for $cred_U$ credentials can be chosen to alleviate this.

U could add comments to his petition signature by adding them to the data that is anonymously signed in step 3 of protocol 1.b.

5 Evaluation

The evaluation of the protocols discussed in the previous section will be given, based on the requirements and attacker model in section 3.

S1. For each petition, U can only be known under a single nym, even when U requests a new vote credential. If that nym is already included in a petition record, the vote is cancelled by P . If P fails to do this, this abuse can be detected by anyone verifying the petition.

Table 1: Protocols for PetAnon

(1.a) Retrieving an anonymous credential

(1)	$U \rightarrow R$: <code>authenticate($cert_U$)</code>
(2a)	R	: <code>if (!hasValidCred($cert_U.Id$))</code>
(2a.1)	U	: <code>$secureRand \leftarrow \text{genSecureRand}()$</code>
(2a.2)	U	: <code>$(comm, openInfo) \leftarrow \text{commit}(secureRand)$</code>
(2a.3)	$U \rightarrow R$: <code>$comm, \text{prove}(\{secureRand \mid comm == \text{commit}(secureRand)\}, comm, openInfo)$</code>
(2a.4)	$U \rightarrow R$: <code>$c \leftarrow \text{enc}(\text{userSecret}, openInfo)$</code>
(2a.5)	R	: <code>$rand_R \leftarrow \text{genRand}()$</code>
(2b)	R	: <code>else</code>
(2b.1)	R	: <code>$(comm, rand_R, c) \leftarrow \text{retrieveCredInfo}(cert_U.Id)$</code>
(2b.2)	$U \leftarrow R$: <code>$c, comm$</code>
(2b.3)	U	: <code>$openInfo \leftarrow \text{dec}(\text{userSecret}, c)$</code>
(3)	$U \rightleftharpoons R$: <code>$cred_U \leftarrow \text{issueCred}(comm.secureRand, rand_R, \text{subset}(cert_U.attributes))$</code>
(4)	R	: <code>$\text{store}(cert_U.Id, comm, rand_R, c)$</code>
(5)	U	: <code>$\text{store}(cred_U)$</code>

(1.b) Signing a petition

(1)	U	: <code>$nym \leftarrow cert_{pet}.f_{pet}(cred_U.secureRand, cred_U.rand_R)$</code>
(2)	$U \rightarrow P$: <code>$nym, props_U, choice$</code>
(3)	$U \rightleftharpoons P$: <code>$proof \leftarrow \text{showCred}(cred, props_U \ \&\& \ nym \sim cred)\{choice, \text{timestamp}\}$</code>
(4)	P	: <code>if (petitionSigned(nym)) abort()</code>
(5)	$U \leftarrow P$: <code>$voteNr \leftarrow \text{getVoteNr}()$</code>
(6)	$U \leftarrow P$: <code>$sig_{vote} \leftarrow \text{Sig}(SK_{pet}, (voteNr, \text{Hash}(proof), nym))$</code>
(7)	P	: <code>$\text{publish}(voteNr, nym, proof, sig_{vote})$</code>
(8)	U	: <code>$\text{store}(sig_{vote}, \text{Hash}(proof), voteNr)$</code>

S2, P3. U will need to prove certain personal properties (if the voter's set is restricted); he may or may not disclose additional properties.

S3. P is always able to prevent the completion of the `signPetition()` (or `withdrawSignature()`) protocol. Although this abuse is hard to prove, it is very noticeable and will result in loss of trust in that P .

Secondly, we must guarantee that no two U 's possess the same nym (which would also compromise S6). The nym for a specific petition is calculated as $nym \leftarrow f_{pet}(secureRand, rand_R)$, where $secureRand$ should be randomly chosen by the user. It is infeasible to prove in zero-knowledge that this value was indeed randomly chosen and not already chosen by an-

other U . Therefore, R also chooses a random value $rand_R$. If both values are used as input for f_{pet} , and sufficiently large, the probability that the same nym is generated for different users is neglectable.

S4. Only U , i.e. the owner of the nym included in the vote record, can generate a new proof in which he proves possession of that nym and states that he withdrew this vote (cfr. `withdrawSignature()`). Newer proves should replace older ones (cfr. timestamp), allowing U to withdraw or change his vote as many times as he wants.

S5. U can detect and prove whether his vote was modified or deleted based on the receipt signed by P .

S6. The publicly available records can be verified by anyone. To change a vote or the corresponding properties, the proof must be modified, which can only be done by the owner of the corresponding nym that is also included in the record. If U executed multiple times `signPetition()` and/or `withdrawSignature()`, P has several valid records w.r.t. a specific nym and can simply choose which one to publish. Therefore, timestamps are included in the proofs (and thus also in the receipts and records). If P does not publish the most recent record, U can prove this with a more recent receipt. Alas, this type of abuse cannot be detected by other parties.

The `voteNr` is incremented for each new signature; P can thus only delete the most recent votes, which can be quite visible and which is always provable by the record owner.

If an entity tries to add forged votes, it must be able to generate multiple nyms, which is only possible if a compromised R issues multiple sign credentials to the same entity. R must thus be trusted not to do this.

If an external attacker wants to delete a vote, he must perform the `withdrawSignature()`. Therefore, he must proof possession of the corresponding `nym`, which is infeasible.

P is unable to use a vote for one petition for another petition as the nyms used in the proofs are petition specific.

U can be coerced; the receipt can be used by U as a proof for the coercer. The latter will then know `voteNr`, allowing him to detect a vote change by that U . Although this attack cannot be prevented, it will be very hard for a coercer to have a real impact on the petition. If the coercion only happens for a short time, U can change his vote afterwards.

In conclusion, P can delete records, or replace them by older vote records owned by that same user, but this is very visible and even provable by the user who casted the vote. Coercion or vote buying cannot be prevented, which makes PetAnon unsuitable for eVoting.

P1. If no identifying attributes are revealed, U remains anonymous to P (Idemix property). Moreover, privacy is preserved in case of collusion of R and P as one of the two values needed to calculate the nym is only known to U . Evidently, R can know a lot about U .

P2. To sign a petition, U authenticates anonymously. Besides some statistically interesting data such as gender, only a nym is revealed. This nym is different for each petition. Based on the properties of secure one-way provable functions, it is impossible to link two nym of the same user, even if the petitions are organized by the same P .

6 Generalization of the Nym Construction.

In the PetAnon system, a different, certified, provable one-way function is issued for each petition and each sign credential contains two sufficiently large random values, wherof one unknown to R . This offers a means for unlinkability of petition signatures of the same U for different petitions on the one hand, and unlinkability of the petition's signature and the real user's id, which is known to R . These two types of unlinkability are computationally secure, even if all the involved parties (excluding U) collude.

This construction can be generalized. It allows service providers to limit per service usage by the same user, while the usage by that user of the different services cannot be linked and while the user is also ensured that he will remain anonymous. This holds even if the credential issuer and (all) the service provider(s) are the same entity or collude. If each service can have a set of n one-way functions, U can, by simply selecting an unused one-way function, access the service maximum n times without enabling the service provider to link the different visits.

The anonymity can be made conditional by introducing in the credential a user identifier. When the credential is shown, a verifiable encryption of that identifier is generated and sent to the service provider. The 'verifiability' property allows the user to prove that the encrypted content equals the identifier in the credential and that a third (e.g. legal) party can decrypt it.

7 Implementation

7.1 General set-up

The prototype consists of two applets and two server systems. The first applet is used to obtain a new credential, the second to vote and to verify. PetAnon uses the Belgian eID card to authenticate with the registration server and to retrieve identity information of the owner. However, other technologies could be used. The personal attributes embedded in the credential in our test environment were age, zip code and gender.

7.2 XML Descriptions

Three PetAnon specific types of XML descriptions were defined. P first composes a (quite straightforward) *petition description*, which contains the petition name, extended textual description, validity period, vote options and the personal properties that may or must be shown by U to P (defines e.g. the age intervals), as well as the voting options. This XML description is provided by P to R when registering a petition and its hash is included in the X.509 $cert_{pet}$ certificate to prevent modification and is also published. U has an auto-generated *credential description*, describing the values in his $cred_U$. Together with the petition description and with help of U , the exact properties that will be released to P are determined, as well as U 's vote choice and is put in a dynamically generated *show description*.

To show an Idemix credential, an Idemix specific XML description must be generated at the user's side, based on the show description, that also proves the correctness of the nym. This description is sent to P before the Idemix show protocol can be performed.

7.3 Performance

Computational efficiency. Table 2 shows the computational performance, based on a 1.87 GHz processor for a petition where U 's age interval and gender are revealed, while hiding his zip code. P can thus handle 206 2048 bit signatures per hour, which is also the speed at which a U can verify the correctness of the petition. For a petition with 10,000 signers, verification lasts thus more than 14 hours. P is able to process about 3270 1024 bit signatures per hour and the petition can be verified at the same rate.

Especially the greater and smaller than relations (used for interval proving) are very heavy. Signing a petition and revealing your exact age and

	1024 bit			2048 bit		
	I or P	U	Total	I or P	U	Total
<i>Retrieving sign cred</i>	0.6s	0.5s	1.1s	1.7s	1.7s	3.4s
<i>Signing a petition</i>	1.1s	1.1s	2.2s	5.1s	5.4s	10.5s
<i>Record verification</i>	-	1.1s	1.1s	-	5.1s	5.1s
<i>Checking own vote</i>	N/A	1.1s	1.1s	N/A	5.1s	5.1s
<i>Withdraw vote</i>	1.1s	1.1s	2.2s	5.1s	5.4s	10.5s

Table 2: Illustration of the computational performance of the PetAnon system on an Intel(R) Core(TM) CPU T5600 @ 1,83GHz for 1024 bit and 2048 bit keys for classical signature and Idemix

	1024 bits	2048 bits
Raw Idemix proof	5.3KB	9KB
XML show description	1.0-2.0KB	1.0-2.0KB
Nym	128 byte	256 byte
Classical signature	128 byte	256 byte
Vote choice	< 256 byte	< 256 byte

Table 3: Record storage cost in the PetAnon system.

gender only lasts about 1.7s (1.0s by U , 0.7s by P) for 1024 bit security and is reduced to 0.4s (0.2s by U and 0.2s by P) for 2048 bit security.

In reality, servers will have more powerful processors; e.g. 2GHz quad core, which is a combined 8GHz, thus seriously increasing the capacity of the issuer and petition organizers. Parallelism and more efficient Idemix implementations can reduce the overall protocol execution time. Tests showed that 88.5% of the computational cost of the Idemix show protocols (for 2048 bit security) comes from modulo operations on big numbers. More efficient (e.g. hardware) implementations can seriously enhance this.

Storage efficiency. Table 3 illustrates for the same petition the sizes of the data in a record, which thus requires about 11-12KB or 7-8KB for 1024, respectively 2048 bit security, i.e. 110MB or 70MB for 10,000 votes. More concise XML notations and Idemix proofs can reduce this. Simpler requirements result in smaller proofs; when only one’s age and gender are revealed, the raw Idemix proofs are reduced to 1.0 till 1.3KB.

Summary. The feasibility of PetAnon will highly depend upon the complexity of the revealed properties and on the latency caused by the underly-

ing mix network. The final version of this paper will present more elaborate measurements taking into account connection and authentication costs.

8 Related Work

The main difference with voting protocols ([1], [7], [10], ...) is the voting behavior repudiability requirement, which is less relevant in petition systems. Less attention has been devoted to e-Petition systems.

Scotland has experimented with e-Petitions. In [9], the government distributed (login,PIN) tuples to schools, which in turn distributed these to their pupils. [11] says about a second system: “[...] *manual moderation of signatures [...] has been preferred.*”. These systems clearly do not suffice.

Recently, another e-Petition based on the Belgian eID card and Idemix was developed [3], but nor a description of the underlying protocols, nor a thorough evaluation was provided.

An Idemix based anonymous reviewing system [12] 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.

9 Conclusion and Future Work

The Idemix based anonymous petition system PetAnon was proposed. The Belgian eID card, that is in casu not privacy friendly was used as a bootstrap. We showed that it is possible to offer anonymity combined with reliable results for e-Petitions while at the same time, PetAnon is much more versatile than other petition systems. A prototype was developed and revealed that it can be quite demanding w.r.t. computation and storage.

We showed how unlinkability of a signature with other signatures of that user, as well with the user’s identity can be guaranteed, while the user can only vote once. This construction was extended and generalized.

Integration of Idemix credentials on (eID) smart cards would make the registration phase superfluous. Therefore, Idemix efficiency improvements will likely be necessary. Measurements using mix-networks are required.

References

- [1] J. Benaloh and D. Tuinstra. Receipt-free secret-ballot elections. In *STOC '94: Proceedings of the twenty-sixth annual ACM symposium on*

- Theory of computing*, pages 544–553, New York, NY, USA, 1994. ACM.
- [2] S. Brands. *A technical overview of digital credentials*. 1999.
 - [3] M. Kohlweiss C. Diaz, H. Dekeyser and G. Nigusse. Privacy preserving electronic petitions.
 - [4] J. Camenisch and E. Van Herreweghen. Design and implementation of the idemix anonymous credential system, 2002.
 - [5] J. Camenisch and A. Lysyanskaya. An Efficient System for Non-transferable Anonymous Credentials with Optional Anonymity Revocation. In *EUROCRYPT '01: Proceedings of the International Conference on the Theory and Application of Cryptographic Techniques*, pages 93–118, London, UK, 2001. Springer-Verlag.
 - [6] D. Chaum. Security without identification: transaction systems to make big brother obsolete. *Commun. ACM*, 28(10):1030–1044, 1985.
 - [7] D. Chaum. Secret-ballot receipts: True voter-verifiable elections. *IEEE Security and Privacy*, 2(1):38–47, 2004.
 - [8] I. Damgard, T. Pedersen, and B. Pfitzmann. Statistical secrecy and multi-bit commitments, 1996.
 - [9] A. Macintosh and A. Whyte. Electronic democracy and educating young people.
 - [10] T. Moran and M. Naor. Receipt-free universally-verifiable voting with everlasting privacy. In *CRYPTO 2006*, volume 4117 of *Lecture Notes in Computer Science*, pages 373–392. Springer, 2006.
 - [11] A. Macintosh N. Adams and J. Johnston. E-petitioning: Enabling ground-up participation. In *Challenges of Expanding Internet: E-Commerce, E-Business, and E-Government*, pages 129–140, Poznan, 2005. IFIP.
 - [12] V. Naessens, L. Demuyck, and B. De Decker. A fair anonymous submission and review system. In *Communications and Multimedia Security*, pages 43–53, 2006.
 - [13] T. Pedersen. Non-interactive and information-theoretic secure verifiable secret sharing. In *CRYPTO '91: Proceedings of the 11th Annual International Cryptology Conference on Advances in Cryptology*, pages 129–140, London, UK, 1992. Springer-Verlag.