TRANSACTIONS ON DATA PRIVACY 5 (2012) 553–578

# P3ERS: Privacy-Preserving PEer Review System

**Esma Aïmeur*, Gilles Brassard*, Sébastien Gambs**, David Schőnfeld***

*Université de Montréal, Département IRO, C.P. 6128, Succ. Centre-Ville, Montréal (QC), H3C 3J7 Canada.

**Université de Rennes 1 - INRIA, IRISA, Campus de Beaulieu, Avenue du Général Leclerc, 35042 Rennes Cedex, France.

E-mail: {aimeur, brassard}@iro.umontreal.ca, sebastien.gambs@irisa.fr, schonfeld.david@gmail.com

**Abstract.** Even though they integrate some blind submission functionalities, current conference review systems, such as EasyChair and EDAS, do not fully protect the privacy of authors and reviewers, in particular from the eyes of the program chair. As a consequence, their use may cause a lack of objectivity in the decision process. In this paper, we address this issue by proposing P3ERS (for Privacy-Preserving PEer Review System), a distributed conference review system based on group signatures, which aims at preserving the privacy of all participants involved in the peer review process. For this purpose, we have improved on a generic group signature scheme with revocation features and implemented it in order to ensure the anonymity of the submission and the reviewing phases. We argue that P3ERS could contribute to increase objectivity during the conference review process while maintaining privacy of the various participants.

**Keywords.** Conference Review System, Privacy, Group Signature.

## 1 Introduction

Nowadays, peer review is the most commonly used paradigm in the research community to assess and validate novel scientific contributions. This collaborative process has unquestionable merits, but it may sometimes suffer from a lack of transparency and objectivity. For instance, several factors can lead to a bias (often not even consciously) in the decision process, such as knowledge of the author's gender, his prolificacy or the fact that he comes from a famous institution [18]. In order to minimize such biases and to strive for a fair review process, two radically different families of approaches for peer review systems are currently used:

- In *open peer review systems*, the reviewers know the identity of the authors of papers and explicitly sign their reviews, such as on the online academic journals *Philica* [1] and *Biology Direct* [2]. These systems aim at achieving objectivity through full transparency

[1] http://www.philica.com
[2] http://www.biology-direct.com

of the review process, but they also have drawbacks, such as the generation of dissatisfied colleagues who are not happy with the advice provided by the reviewers and also potential damage to the reputation of an author when his paper is not accepted.

- *Anonymous peer review systems*, such as *EasyChair* [3], EDAS [4] and *HotCRP* [5], can be split into two categories, depending on whether they aim at achieving *single* or *double* blind review. In single blind review, the anonymity of the reviewers is preserved from the authors, but not vice versa. The double blind review process goes a step further by trying to ensure anonymity in both directions.

Although the double blind approach increases fairness of the review process, these systems still have some weaknesses. For instance, some entities, such as the program chair and the conference managing system provider, have access to all the information. Furthermore, reviewers may have access to the identity of the other reviewers for the papers that are assigned to them. It follows that there is always a risk of sensitive data disclosure.

Moreover, in addition to the bias introduced during the review process, some of the current peer review systems gather important amount of data about the activities of researchers, which can be damaging to their reputation in case of a privacy breach. Indeed, in his inauguration lecture titled "Big Brother and Little Brother: The Future of Privacy" [6], Mark Ryan compares the current peer review systems (and especially EasyChair) to a "little Facebook" [2]. For instance, EasyChair records all the submissions and activities of the researchers that participate to the conferences it helps organizing, thus raising important privacy concerns. While Facebook creates a huge database composed of the personal information of more than 750 millions users, EasyChair also does it, albeit at a smaller scale, with the history of submissions and reviewing of researchers, starting from 2 conferences in 2002 to more than 4500 in 2011. Overall, since its creation Easychair has hosted more than 19 571 conferences with a total of 724 272 users according to the information available on its website accessed on 31 October 2012. Furthermore, Easychair has recently required that all its users abide to its new terms of use if they want to be able to continue to use the service. Currently, the privacy statement mentions that Easychair will "not sell, distribute, or pass your personal data to any organization" and that "the only persons who have access to this (conference) data are those persons who have access to it as a consequence of their use of EasyChair". However, the terms of use to which the users have to abide also clearly state that "We reserve the right to update the Terms at any time at our discretion without notice to you. [...] Your continued use of the Service shall constitute your consent to such changes and you shall be bound by the updated Terms." Even though we are not saying that this is their current intent, there is a privacy risk that in the future Easychair could silently, yet legally, modify their terms of use and then be free to sell to third parties the information it has collected (or worse). HotCRP avoids the problem of centrally collecting the history of submissions and reviews by requiring the program chair to run it on his own server but "HotCRP version 2.47 was subject to an information exposure where some authors could see PC comments", according to the HotCRP website [5], accessed on 7 April 2012.

In this paper, we address this issue by proposing P3ERS (for Privacy-Preserving PEer Review System), a distributed conference review system based on group signatures that aims at preserving the privacy of authors and reviewers. More precisely, we first identify

[3] https://www.easychair.org
[4] http://edas.info
[5] http://www.read.seas.harvard.edu/~kohler/hotcrp — Anonymity is optional with HotCRP.
[6] http://www.cs.bham.ac.uk/~mdr/slides/pdf/11-inaugural-slides.pdf

the desirable properties that a privacy-preserving peer review system should fulfil. We then describe the group signature scheme that forms the basis of our system and that is used to enforce these desirable properties. In a nutshell, the group signature scheme allows users to sign a message anonymously on behalf of a group. Therefore, a user can demonstrate to a verifier that he belongs to a specific group (thus providing authentication) but without disclosing his identity (thus remaining anonymous within the group). Contrary to most of the standard anonymous credential schemes, the group signature scheme that we use offers the property of anonymity revocation, in the sense that anonymity can be lifted by a dedicated trusted third party, thus revealing the identity of the signer of the message. One of the main ideas of P3ERS is to ensure the privacy of both the authors and the reviewers (and this even from the point of view of the conference provider and the conference chair) by using two different groups of users. In particular, the authors can submit anonymized papers on behalf of the author group to the program chair, who then dispatches the papers according to the declared skills of the reviewer group members in an oblivious manner. In this way, the program chair knows neither the identity of the authors (until a paper is accepted, if it is) nor the correspondence between papers and reviewers.

This paper is organized as follows. First, in Section 2, we define the desirable properties that a privacy-preserving peer review system should fulfil. Afterwards, Section 3 summarizes the related work, including the background on peer review systems and group signature schemes. We then describe in Section 4 the group signature scheme that we use as well as its Java implementation in the form of a generic library. We also briefly compare the performance of this group signature scheme with other related schemes. Section 5 describes the step-by-step architecture of P3ERS. In Section 6, we analyse the security and privacy of our architecture according to a variety of possible threats and we discuss its implementation. In Section 7, we discuss a linkability weakness in our system, which allows the program chair to determine which papers are reviewed by the same reviewer. We conclude with desirable improvements for our system and other open questions in Section 8.

## 2 Desiderata for the Peer Review Process

The peer review process can be divided into several steps. First, a program chair sets up a conference on a particular topic and issues the corresponding call for papers, describing the research topics covered by the conference (*initialization phase*). This initialization phase also involves the selection of a program committee by the program chair. Then, the authors submit their work to the conference via a dedicated website (*submission phase*). Afterwards, each program committee member (henceforth called *reviewer*) expresses his interest in reviewing specific papers (*bidding phase*). Note that this bidding phase is optional and that sometimes the program chair directly matches reviewers with papers that have been submitted. Once the bidding phase is finished, papers are assigned to reviewers, who can then evaluate their quality by grading them and providing constructive criticism (*reviewing phase*). Finally, based on the evaluations received and possibly on the ensuing discussion between reviewers in the case of borderline or problematic papers, the program chair makes the final decision of which papers to accept (*decision phase*).

As mentioned previously, most of the current peer review systems can suffer from a bias during the review process and may potentially lead to important privacy leaks if their databases of researcher activities is ever exposed (for instance due to the exploitation of a

security breach). In order to alleviate these drawbacks, we identify thereafter the desired properties that a privacy-preserving peer review system could fulfil in order to increase the privacy of its users.

- *Blindness*. Several parts of the review process should be anonymous in order for this property to hold:
  1. The authors do not have access to the identity of the reviewers of their papers (*single blind* property).
  2. The reviewers do not know the identity of the authors of the papers that they review (*double blind* property). We speak of a *strong* double blind property if even the program chair does not know the authorship of individual papers.
  3. The program chair does not know the list of members of the author group (unless a submitted paper is accepted thus leading to a de-anonymization of the identities of the authors of this paper). Moreover, the program chair knows the list of members of the reviewer group (since he appointed the program committee), but does not know the assignment of the different papers between reviewers (*triple blind* property).
  4. A reviewer cannot have access to the evaluations of the other reviewers on the same paper except if a discussion on this paper needs to take place (in case of a borderline or problematic paper). In this situation, a reviewer could have access to the comments of other reviewers but perhaps not to their identity.

  Note that most of the current peer review systems, such as EasyChair or EDAS, can achieve properties 1 and 2 (although usually not in the strong form), but fail to meet the rest of these properties, thus not providing full blindness into the review process.
- *Unlinkability*. It is impossible to determine if two papers (resp. reviews) have been written by the same author (resp. reviewer), except if a dedicated trusted third party (the Group Manager in our case) decides to lift their anonymity or if the papers are accepted.
- *Reputation preservation*. A rejected paper is never de-anonymized to preserve the reputation of the author(s).
- *Anonymity revocation*. The anonymity of an author or a reviewer cannot be lifted unless the opening manager decides to do so, perhaps because of suspected wrongdoing or because a paper is accepted.
- *Non-repudiation*. Once the anonymity of an author or a reviewer has been revoked, he cannot deny having written this paper or review.

These properties contribute to protect the privacy of all participants involved in the reviewing process, thus helping to maintain the objectivity of the different actors that can influence the outcome of the review process (*i.e.*, program chair and reviewers). These properties are achieved in P3ERS through the designed architecture and the use of a group signature scheme with anonymity revocation. Of course, it is possible to imagine other alternative architectures and techniques that could also be used to implement these desiderata.

Table 1 presents a comparison between several conference review systems (the list is not exhaustive), including our system P3ERS and *ConfiChair*, another peer review system

Table 1: Comparison between various online review systems. Columns "Single", "Double" and "Triple" correspond to whether or not the system is single blind, double blind or triple blind; "opt" stands for "optional" and column "Free" indicates whether the review system is free of charge or if one has to pay to use it.

| | Free | Deployment | Single | Double | Triple |
|---|---|---|---|---|---|
| **EasyChair** | | On their servers | ✓ | ✓ | |
| **HotCRP** | ✓ | Locally installed | opt | opt | |
| **Web Submission and Review** | ✓ | Locally installed | ✓ | | |
| **CRS** | | On their servers | ✓ | | |
| **Conference Service Mandl** | ✓ | On their servers | ✓ | | |
| **EDAS** | | On their servers | ✓ | ✓ | |
| **ConfiChair** | ✓ | On their servers | ✓ | ✓ | |
| **Our system P3ERS** | ✓ | Distributed | ✓ | ✓ | ✓ |

integrating privacy features [2]. Currently, the program chair of a conference has mainly two possibilities among which he can choose. One possible choice is to place total reliance on an external site such as EasyChair, which stores all the data related to the conference on its servers (submissions, reviews, lists of authors and reviewers...). Apart from the fact that the program chair does not have to install and maintain the conference infrastructure, the main advantage of this solution is that an individual can use the same account for several conferences (playing the role of author and/or reviewer), which improves the usability of the system as long as the user does not mind the erosion of his privacy stemming from the constitution of a dossier about his scientific activities.

The alternative is that the program chair can deploy himself the conference infrastructure on his servers (or the servers of his institution) in order to maintain the confidentiality of the data entrusted to him. This second solution has several drawbacks: the program chair needs to spend a considerable amount of time and energy to set up the system, the institutional server may not be as robust as dedicated conference review websites (for instance, they might crash due to a large amount of last-minute submissions), users need to remember an URL, login and a password for each conference and spend time to become familiar with various interfaces. Moreover, the anonymity of authors and reviewers is still relative, even if the double blind property is implemented, as the program chair has access to all the information concerning the conference. And even if the program chair is honest, the institutional server could be less well protected against hackers than a dedicated conference review website. For all these reasons, we think that an important first step towards creating a more privacy-preserving review system is to distribute it in order to minimize the risks of information disclosure. In short, no entity should have access to full information, but rather the sensitive data should be distributed among the different actors (authors, reviewers, program chair and conference provider), according to the *Division of Trust Principle* [1]: Trust no one, but you may trust two.

# 3 Background and Related Work

In this section, we provide an overview of group signature schemes as well as related work on integrating privacy features into peer review systems.

### 3.1 Group signature schemes

Group signature schemes were introduced in 1991 by Chaum and van Heyst [7] as an extension of digital signature schemes in which individuals can sign a message on behalf of a group rather than directly under their own identity. While it is impossible to trace the exact identity of the signer of a message, the validity of the signature can be checked by using the public group verification key. Group signature schemes are often used as fundamental building blocks for the development of anonymous credential systems. Since the seminal work of Chaum and van Heyst, extensions have been proposed [4, 6, 10, 17, 19, 20] to improve their efficiency or to implement different variants of group signature schemes under various cryptographic assumptions or distribution of roles among entities (see Ref. [5] for a review of some of the existing schemes).

Entities involved in a group signature scheme can be divided broadly into three categories:

1. The *Group Manager*, which is the most important entity, is in charge of delivering and revoking group membership certificates to members. He is also responsible for the potential opening of signatures should the need to de-anonymize the signer of a message arise. Note that protocols in the literature either assume that these three functions are all performed by the same entity or that they are split among three separate ones [21].
2. *Group members*, who can generate group signatures by using their secret keys and their group membership certificates after having registering to the Group Manager.
3. The *Verifier* is the entity responsible for verifying the validity of a group signature (*i.e.*, that the signature was generated by a genuine group member). Note that any entity that knows the public group verification key can play the role of the verifier in the system.

A group signature scheme with revocation features is generally composed of the seven following procedures:

1. `Setup`. This procedure is performed by the Group Manager. It consists in running an algorithm to initialize the public parameters, as well as the public key and secret key of the group signature scheme.
2. `Join`. This protocol is run between a user, who wants to become a group member, and the Group Manager. During this protocol, the user generates his secret signing key at random and receives a group membership certificate; his identity is registered by the Group Manager. This certificate, which does not need to be kept secret, has different roles when combined with various secret keys. First, it is needed by the member, together with his secret signing key, in order to sign a message on behalf of the group. Second, it serves as a link between a signature and the identity of the signer when a signature is de-anonymized. Finally, it is needed by the Group Manager in order to revoke a specific member.
3. `Sign`. This algorithm takes as input a member's secret key together with his group membership certificate, as well as a message. It produces a signature of this message on behalf of the group. We shall lighten the notation in the rest of the paper by considering that the message is only signed with the secret key.

4. `Verify`. This algorithm is used to verify that a particular signature has been produced by a legitimate group member (but without learning his identity). More precisely, the algorithm takes as input the public group verification key, a message and a group signature on this message and returns as output "accept" if the validity of the signature is verified, and "reject" otherwise.

5. `Open`. This algorithm can only be performed by the Group Manager. It takes as input the Group Manager secret key, a message and a signature on this message. It reveals as output the identity of the group member who has performed this signature.

6. `Revoke`. This algorithm also can only be performed by the Group Manager. It results in revoking a particular member from the group. More precisely, the group membership certificate of this user becomes no longer authorized to produce valid signatures, which requires the public group verification key to be updated accordingly. Please do not confuse membership revocation with anonymity revocation, the latter being the task mentioned in point (5) above.

7. `Update`. After the revocation of a member, the remaining group members have to update their group membership certificates in order to produce further valid group signatures. For this purpose, each group member receives the updated group verification key and also updates locally his own certificate by taking into account this new public verification key. Note that the revoked member cannot successfully update his certificate.

## 3.2 Related work

Group signature schemes have been previously used in the design of privacy-preserving auction systems, a research area that shares some similarities with peer review systems. For instance, distributed architectures to achieve secure online auction systems relying on group signature schemes have been proposed [13, 16]. In a nutshell, after registering to an independent Group Manager, each bidder sends anonymously his bid signed under a valid group membership certificate to the system. Only the signature of the winner is sent to the Group Manager, which then runs the `Open` algorithm to reveal the identity of the winner. This selective opening of anonymity protects the anonymity of the other bidders of the auction while ensuring at the same time the property of non-repudiation (*i.e.*, the winner of the auction cannot deny the bidding he has cast).

With respect to the development of a privacy-preserving review system, we are only aware of another related system called ConfiChair, developed by Arapinis, Bursuc and Ryan [2]. The architecture and the aim of ConfiChair are very different from those of P3ERS, as its main objective is to deal with the privacy and security concerns raised by untrusted cloud storage servers. Therefore, the only adversary considered in ConfiChair is the "cloud" itself (in contrast to our work, in which the adversary can potentially be any one of the actors in the review process, including the program chair, but with different goals in mind). ConfiChair ensures the confidentiality of conference data against untrusted servers by encrypting this data with a mechanism using key translations and mixes at the level of the web browser.

# 4 Extension of a Group Signature Scheme

Most of the existing group signature schemes are not used in practice, either because they result in a high computational cost or because they have too strong intrinsic constraints to be widely deployed. For instance, the complexity of group signatures is sometimes dependent on the number of group members [7] or the addition and removal of members cannot be performed in a dynamic manner. Thereafter, we list the properties that a group signature scheme should achieve in order to be useful for implementing P3ERS:

- The scheme must provide constant-size signatures as well as constant-size public group verification keys, and this independently of the number of group members.
- The algorithms `Join` and `Revoke` must be efficient and capable of working in a dynamic manner.

We have searched unsuccessfully for free and open-source implementations of group signature schemes satisfying these requirements. The only two references to a working implementation of group signature schemes with revocation features that we have found [8, 12] are proprietary. However, the scheme proposed by researchers from NEC [12], which is an extension of a protocol proposed by Camenisch and Groth [5], was sufficiently detailed to allow us to implement and extend it to match our requirements. The security of this group signature scheme is based on standard cryptographic assumptions, namely the hypothesis behind RSA (*i.e.*, the difficulty of factoring the product of large prime numbers) and Elliptic Curve Cryptography (*i.e.*, the difficulty of resolving the elliptic curve discrete logarithm problem [3]). Moreover, this scheme meets all the desired properties listed above. (We refer the reader to the original paper for details [12].)

An additional argument for the choice of this scheme is the complete separation of the functionalities provided by the Group Manager. More precisely, these functionalities can be attributed to three different entities:

- The *IssuingManager* is in charge of delivering group membership certificates and maintaining the list of group members.
- The *OpeningManager* is the only entity that has the ability to open a signature, thus revealing the identity of a signer.
- The *RevocationManager* is the entity in charge of revoking a group member, so that it can no longer produce valid signatures on behalf of the group.

We adopt this separation in order to develop a generic library for our group signature scheme, which can be adapted and re-used for various contexts and applications. Moreover, we extend the scheme by introducing an index in the resulting signatures in order to check retroactively their validity. Indeed, when a group member is revoked, the public verification key of the RevocationManager is also updated and all signatures produced before the revocation by this group member are not considered valid anymore by a verifier. Therefore, when generating a signature, the algorithm `Sign` inserts the current index of the revocation list (*i.e.*, the number of revocations since the creation of the group). Moreover, all the different versions of the public verification key are also stored in a public repository, making it possible to check *a posteriori* the validity of a signature in order to be able to

open it and reveal the identity of the signer even if he is no longer an active group member (perhaps because he has been revoked).

The modified group signature scheme we propose requires a pair of public and secret keys for each entity as summarized in Table 2. This nomenclature is used in the remaining of the paper.

Table 2: Summary of the symbols

| | |
|---|---|
| IssuingManager | |
| $ipk$ | public key of the IssuingManager |
| $isk$ | secret key of the IssuingManager |
| RevocationManager | |
| $rpk$ | public key of the RevocationManager |
| $rsk$ | secret key of the RevocationManager |
| OpeningManager | |
| $opk$ | public key of the OpeningManager |
| $osk$ | secret key of the OpeningManager |
| Member | |
| $gmc_i$ | group membership certificate of the $i$-th member |
| $msk_i$ | secret key of the $i$-th member |
| Security parameters and cryptographic function | |
| $K_n$, $K_\ell$ | security parameters of RSA cryptosystem |
| $K_e$, $K_{e'}$ | size of the membership certificate |
| $K$ | security parameters of Elliptic Curve cryptosystem |
| $\mathcal{H}$ | collision resistant hash function |
| $K_c$ | length of the output of the hash function |

The proposed scheme is implemented with the Java programming language. Table 3 shows the performance of the algorithms implemented when executed on a PC with a CPU of 2 GHz (Intel Dual Core processor) equipped with 2 Go DDR2 of RAM. The operating system used is Linux/Ubuntu (11.04) and the language and virtual machine are from the Java release open-jdk-1.6.0. In terms of security parameters, we use $K_n = K_\ell = 1024$ bits, $K_e = 504$ bits, $K_{e'} = 60$ bits, $K = 169$ bits, $K_c = 160$ bits and we rely on the SHA-256 hash function for $\mathcal{H}$.

As it can be observed from Table 3, the performance of our implementation in terms of computational time is not as good as that reported in Ref. [12] for most of the functionalities. This difference is mainly due to the fact that the original scheme was implemented using the C++ programming language, which itself does not require a resource-consuming virtual machine. However, remember that our main goal is not to provide an optimized new group signature scheme but rather to implement an open-source library efficient enough to make possible the development of P3ERS. Indeed, the computational time that we obtain remains reasonable for practical use in most applications, except those that require real-time behaviour. We estimate that our implementation could be sped up with parallel computation, especially for the `Sign` and `Verify` algorithms, in which the computation of intermediate variables is largely independent from one another.

During the development of our implementation, we primarily used standard Java libraries such as `BigInteger` for the representation and the computation of large num-

Table 3: Performance and comparison of the implementation

| Algorithm | Running Time (sec) Our scheme | Running Time (sec) NEC [12] |
|---|---|---|
| SetupIssuingManager | 1.750 | (not specified) |
| SetupRevocationManager | 1.700 | (not specified) |
| SetupOpeningManager | 0.250 | (not specified) |
| Join | 1.200 | 3.2 |
| Sign | 1.300 | 0.135 |
| Verify | 1.20 | 0.112 |
| Open | 0.130 | 0.113 |
| Revoke | 0.002 | 0.030 |
| Update | 0.005 | 0.003 |

bers. However, Elliptic Curve Cryptography is not part of the standard security classes of Java, and therefore we had to use classes `ECPoint` and `EllipticCurve` implemented on an external library developed by Bouncy Castle[7]. More precisely, we used the elliptic curve `secp256`, as recommended by NIST [15].

Before building P3ERS, we have developed some simpler applications, such as an *anonymous comment system*, in order to evaluate the efficiency and the applicability of our implementation of the group signature scheme. The group signature library and the simpler applications developed are freely available as open-source code (under the GNU GPL v2 License) at `http://code.google.com/p/group-signature-java/`.

# 5 Privacy-Preserving Peer Review System

In this section, we describe the architecture of our system and we illustrate how it works through a typical example of conference review process.

## 5.1 Architecture of P3ERS

Our main contribution is the design of a privacy-preserving peer review system satisfying the desiderata listed in Section 2. We name this system P3ERS for Privacy-Preserving PEer Review System. The main objective of P3ERS is to ensure the anonymity of the participants during the review process, mainly through the use of a distributed architecture splitting the information among several entities and the use of a group signature scheme. In order to illustrate how our system works, we consider the fictitious case of a conference called *PrivacyConf* and we introduce the following characters:

- *Alex* is an author who wants to submit a paper to PrivacyConf. First, he has to register by becoming a member of the author group. Once his registration is completed, Alex signs his paper with his group signature key before submitting it. The paper must include a list of keywords taken from a list predefined by the program chair.

[7] `http://www.bouncycastle.org/`

At this point, the entity responsible for storing his submission can verify that it came from someone that belongs to the author group by use of the public group verification key, but cannot learn his identity provided the submission is done through an anonymous channel such as the Tor network [9]. After the review process, the signatures associated with the accepted papers are opened in order to reveal the identity of their authors, but rejected papers are never de-anonymized, thus preserving the reputation of their authors.

- *Randy* is one of the reviewers of PrivacyConf, who has been originally contacted by the program chair (*cf.* next paragraph). Following his acceptation to participate in the program committee and based on the same mechanism used for the authors, Randy becomes a member of the reviewer group. Once this is done, Randy fills out a skill form in which he has to select the keywords that best match his expertise among the same predefined list used by the authors. Later, Randy can bid anonymously on the anonymous papers. Finally, he evaluates the papers that the program chair assigns him anonymously (yet keeping the right to refuse some of them).
- In order to foster objectivity and preserve the privacy of authors and reviewers, *Sacha*, the program chair of PrivacyConf, knows the list of reviewers (his program committee), but has no access to the list of authors nor to the assignment of papers between reviewers. The assignment is based only on the skill forms completed by the reviewers, their preferences at bidding time, and the keywords associated to the papers. However, as in other peer review systems, Sacha may have to deal with the occurrence of conflicts of interest, which is rather challenging in a context where all participants are anonymous. We explain later how we achieve this.

In contrast to ConfiChair [2], which considers the protection against an external adversary, such as the "cloud" (*i.e.*, the conference management system administrator, the cloud service provider, the network administrator,...), we think of the participants themselves as potential adversaries (some can be semi-honest while other can even be malicious). We also rely on a semi-trusted third party embodied by the secured website of the conference provider, which implements our P3ERS architecture, hereinafter simply called *the website*. In our system, no entity has access to all the sensitive information and the anonymity of participants is preserved through the design of the distributed review system architecture.

## 5.2 Description of the reviewing process

Figures 1 and 2 sketch the architecture of our system. We now describe the successive phases of the review process.

**Initialization phase.**

1. Initially, Sacha, the program chair, installs and deploys the group signature application on a server at his institution (*e.g.*, his university). This application consists of two parts: (1) the *reviewer group manager* with the ReviewerIssuingManager, the ReviewerOpeningManager and the ReviewerRevocationManager and (2) the *author group manager* with only the AuthorOpeningManager and the AuthorRevocationManager. Indeed, if the program chair were responsible for delivering himself the membership certificates to the author group, he would have complete access to the

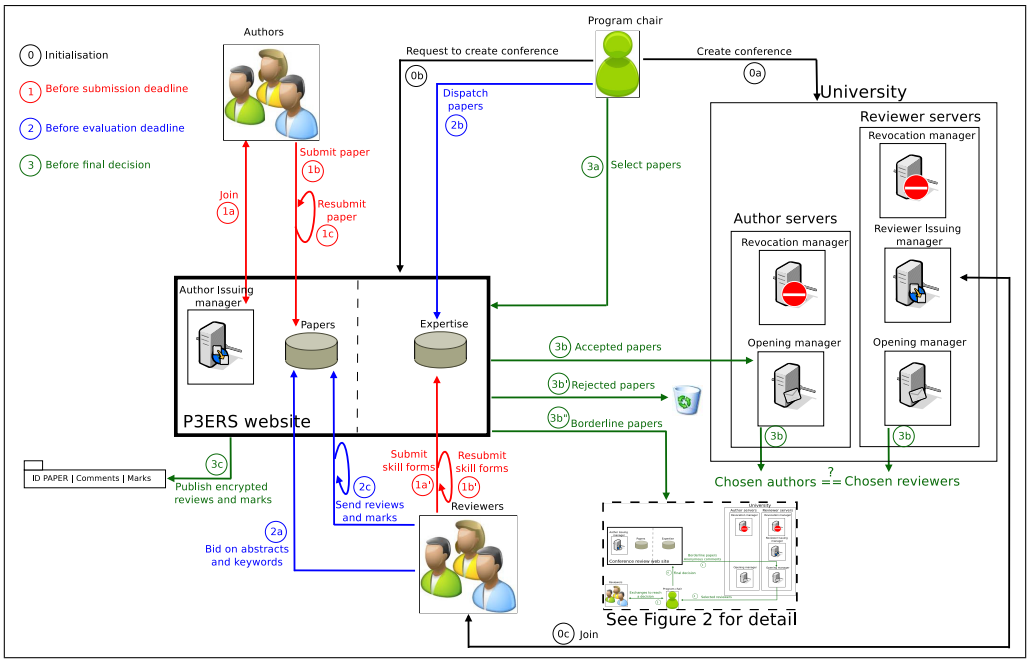


Figure 1: General architecture.

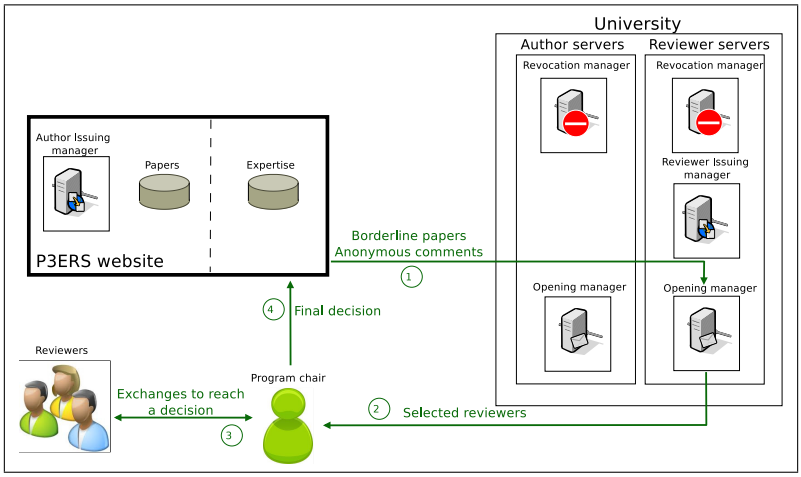


Figure 2: Detail of the architecture (the dashed box at the bottom of Figure 1).

list of authors. In particular, when finally publishing the list of accepted papers, the program chair could form the complementary set of names to disclose the identity of the authors whose papers have been rejected. In contrast, the program chair manages himself the list of reviewers as he is the one responsible for setting up and inviting researchers on this list.

2. The program chair requests from the website (which is a semi-trusted third party) the creation of PrivacyConf. Afterwards, the website verifies that the program chair has correctly initialized both Group Managers through the appropriate secure distributed protocol. In a nutshell, the objective of this protocol is for the website to be convinced of the authenticity the Group Managers set up by the program chair but without learning any additional information (in the spirit of zero-knowledge proofs). If this protocol succeeds, the website runs another distributed protocol in cooperation with the chair to initiate the AuthorIssuingManager.

3. Randy, one of the reviewers invited by the program chair to be on the conference program committee, follows protocol `Join` by interacting with the ReviewerIssuing-Manager installed on the university server. At the conclusion of this protocol, he generates his secret key $msk_{r_i}$ and receives as output his group membership certificate $gmc_{r_i}$ , where index $r_i$ refers to the fact that Randy is the $i^{\text{th}}$ reviewer to register. (When no confusion can arise concerning the reviewer under discussion, we shall lighten the notation and simply write $gmc_r$ and $msk_r$ .) The ReviewerIssuing-Manager stores $gmc_{r_i}$ as illustrated in Figure 3. We assume that the list of reviewers (the program committee) is public and known by the authors in advance. This knowledge is necessary for each author in order to be able to construct his *blacklist* to deal with conflicts of interest.

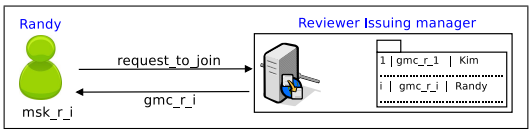


Figure 3: Registration of a reviewer.

4. As shown in Figure 4, Randy, the reviewer, can now complete the form summarizing his skills. He also generates a pair of session keys for the RSA cryptosystem, a public key $pk_r$ and the associated secret key $sk_r$. Then, he sends to the website a message composed of his skill form and his public key $pk_r$ as well as the corresponding signature on this message using his secret group key $msk_r$. After verifying the authenticity of the group signature by calling algorithm `Verify`, the website sends Randy the identifier of the skill form $Enc(id_r, pk_r)$, which corresponds to the encryption under the public key $pk_r$ of a string randomly generated by the website that can be considered as the random identifier of the reviewer form. Randy then saves this identifier $id_r$ for later use.

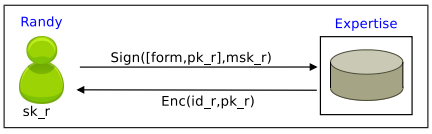


Figure 4: Completion of the skill form.

**Submission phase.**

1. Alex, the author of at least one submitted paper, follows the protocol `Join` with the AuthorIssuingManager located on the website. At the conclusion of this protocol, Alex generates his secret key $msk_{a_j}$ and receives as output his group membership certificate $gmc_{a_j}$ , where index $a_j$ refers to the fact that Alex is the $j^{\text{th}}$ author to register. (Again, when no confusion can arise concerning the author under discussion, we shall lighten the notation and simply write $gmc_a$ and $msk_a$ .) The AuthorIssuing-Manager stores $gmc_{a_j}$ as illustrated in Figure 5. This step is performed only once, regardless of the number of papers submitted by Alex.

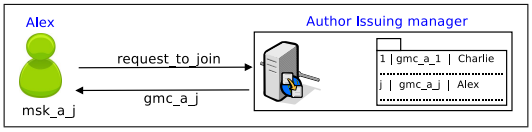


Figure 5: Registration of an author.

2. Since Alex knows the list of reviewers (which is public), he creates a blacklist composed of the names of reviewers with whom he has a conflict of interest (for instance they might be colleagues, co-authors, long-time competitors or even himself). For each paper he wishes to submit, Alex also creates a session pair of keys for an RSA cryptosystem, namely $pk_p$ for the public key and $sk_p$ for the corresponding secret key. The session public key $pk_p$ will be used several times, in particular to encrypt the reviews (comments and marks) of the paper while the corresponding secret key $sk_p$ is kept by Alex and will be used to decrypt these reviews at the end of the entire procedure in case his paper is rejected. Alex then proceeds by anonymously submitting his paper to the website, with its associated blacklist and the session public key $pk_p$, and signs the resulting message with his group signature key $msk_a$. The website verifies the authenticity of the signature by calling the algorithm `Verify`. Afterwards, the website sends back to Alex the paper identifier $id_p$ encrypted with the session public key $pk_p$, which Alex stores for future potential updates of his paper. (We assume that the website can send back this information to Alex without knowing his identity, either by using a bidirectional anonymous channel or by publishing it on a public webpage that any person can browse.) The procedure is illustrated on Figure 6. One should remark that the website only knows that the person that has submitted a paper is a member of the author group, but it cannot identify Alex as being the author due to the use of the group signature[8].

3. Alex has also the possibility to update his submission by sending to the website a message composed of the new version of the paper, the associated blacklist (which could also be updated) and the identifier $id_p$ signed with $sk_p$ (see Figure 7). The website uses $pk_p$ to verify this signature in order to check the authenticity of the request (only the legitimate author knows the secret key $sk_p$ that can be used to sign the paper

[8]In order to avoid a possibility of linking due to timing considerations, it is also important that Alex does not directly submit the paper after having registered as an author. This could be prevented for instance by forcing all the authors to register before a certain deadline and then allowing the submission of papers only after the expiration of this registration deadline.

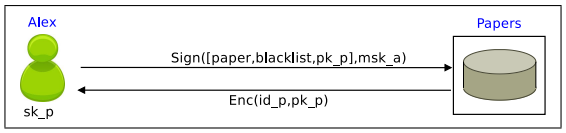


Figure 6: Submission of a paper.

identifier $id_p$) before storing the new version of the paper in its server. As for current conference review systems, Alex can carry out this procedure as many times as he wants before the submission deadline.

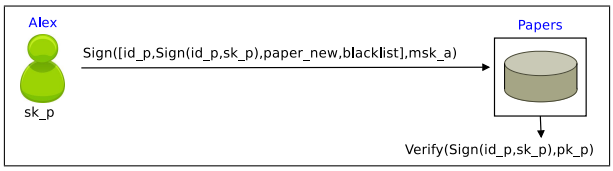


Figure 7: Update of the submission.

**Bidding phase.** Once the submission deadline has passed, the bidding phase starts.

1. The members of the reviewer group are now allowed by the website to access only the titles, keywords and abstracts of the papers submitted by the author group. By taking this information into account, Randy associates a *bidding value* to each paper, which can take any value among {yes, maybe, no}. The case of *value* = no indicates that either the reviewer is not interested to evaluate this paper or that he thinks that he may be in conflict of interest with it. Any paper on which Randy omits to bid will be considered to have *value* = maybe. This process is complicated by the fact that the author of the paper could have blacklisted Randy upon submission, in which case everything should proceed *as if* Randy had chosen *value* = no, except that neither Randy nor the website (nor the program chair) should know whether or not the chosen *value* has been modified.

   It is easiest to achieve this goal if there is a trusted *independent entity* to which Randy can send over a secure channel three messages signed with his secret group signature key $msk_r$:

$$\begin{aligned} & bid &&= Sign([id_r, Sign(id_r, sk_r), id_p, value, \text{“Randy”}], msk_r) \\ & bid_1 &&= Sign([id_r, Sign(id_r, sk_r), id_p, value], msk_r) \\ \text{and} \quad & bid_0 &&= Sign([id_r, Sign(id_r, sk_r), id_p, \text{no}], msk_r) \end{aligned} \tag{1}$$

   Basically, the first message states that the reviewer named “Randy” with skill form $id_r$ wants to bid *value* on the paper with identifier $id_p$. After verification of the group signatures and of the consistency of the three messages (the information that should be the same is indeed the same), the independent entity contacts the website to request the blacklist associated with this paper, giving no other information to the website than $id_p$. Upon receiving the blacklist, the independent entity verifies whether or not

this reviewer (Randy in this case) appears on the list and forwards $bid_0$ to the website if this is the case, or $bid_1$ otherwise. The website stores the information that the reviewer with skill form $id_r$ is ready to bid *value* (or no, whichever is the case) on the paper with identifier $id_p$. The independent entity stores $bid$ in case the need might arise later to prove wrongdoing on the part of Randy. Note that the independent entity does not learn anything about which papers are reviewed by whom because the correspondence between papers and their identifier ($id_p$ in this case) will never be made public, and therefore only a collusion between the independent entity and some other entity can reveal private information (see Figure 8).

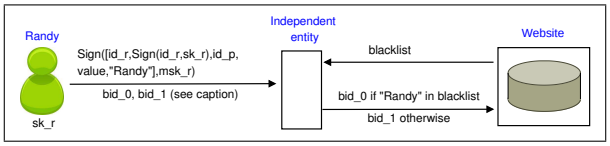


Figure 8: Bidding phase. Messages $bid_0$ and $bid_1$ are specified in Equation (1).

This entire process can be made either simpler or more complicated, depending on the desired security level. A reasonably secure approach consists in using a Java applet on the reviewer side, which is downloaded as a black box from the website with a precompiled blacklist corresponding to the paper under consideration (see the details of the implementation in Section 6.2). This applet takes all three messages discussed above as input and delivers directly to the website the relevant one ($bid_0$ or $bid_1$) after making the required verifications. At the other end of the security spectrum, the reviewer and the website can run a secure two-party computation [14] to simulate the effect of the independent entity without any need for trust into anything else than the cryptographic primitives involved in the process. This last approach, which could be based on the cryptographic primitive known as "oblivious transfer", would likely be more computationally intensive.

**Reviewing phase.**

1. Once the bidding phase has been completed, Sacha, the program chair, carries out the assignment of papers between the reviewers. To help him conduct this task, the website first proposes a semi-automatic assignment computed from the preferences and skills of the anonymous reviewers. One possible algorithm to compute the assignment is the following. First, Sacha decides on the value of parameter $k$, which is the desired number of reviewers that should be assigned to each paper. For each paper, the website looks for the $k$ skill forms, among those for which the bidding value is yes, that best match the keywords associated to the paper. If there are fewer than $k$ such skill forms, the semi-automatic assignment is completed with the best keyword-matching maybe bidding values. If we *still* have fewer than $k$ reviewers, the remaining best-matching skill forms for which the bidding value is different from no are used.

   Once he receives the assignment proposed by the semi-automatic system, the chair can refine this assignment or choose a completely different one. Once all papers have been assigned to their corresponding reviewers (remember that the chair does not

have the knowledge of their exact identities), the website records the assignment. At this point, a message is broadcast to all the members of the reviewer group to ask them to connect to the webpage on which the website has published the results of the assignment process (see Figure 9). Access to this webpage can be limited to the reviewer group by challenging them to sign a random message on behalf of the group before they would be able to access the contents of the webpage. (Note that the website cannot directly send a customized message to each reviewer, as it ignores the correspondence between the skill forms completed and the reviewers.)

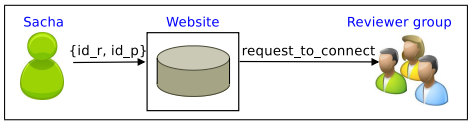


Figure 9: Paper assignment.

2. Randy (the reviewer) connects to the website and sends the message

$$Sign([id_r, Sign(id_r, sk_r)], msk_r)$$

to prove that he has legitimate access the papers associated with the skill form $id_r$. The website returns the papers he needs to evaluate by encrypting them using the public key $pk_r$. Randy can still decide to not evaluate some papers if he thinks that he can guess the identity of the author and that he has a conflict of interest with him. Afterwards, Randy transmits his review on each paper by sending the following message to the website: $Sign([id_r, Sign(id_r, sk_r), id_p, comments, marks], msk_r)$ (see Figure 10). In short, this message states that the reviewer with skill form $id_r$ submits the review $comments$ with the ratings $marks$ (which may include a confidence factor) on the paper with identifier $id_p$. The legitimacy of $id_r$ is guaranteed by its signature under secret key $sk_r$. Until the end of the review phase, Randy may update his review by using a mechanism similar to the one used by authors to update their papers until the submission deadline.

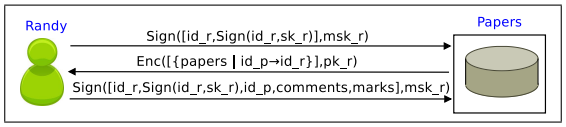


Figure 10: Review submission.

**Decision phase.** This last phase has three possible outcomes for each paper, depending on its evaluation: accepted (b), rejected (b') or under discussion (b").

a) Sacha connects to the website that presents him with a semi-automatic ranking that has been computed based on marks given to the papers by the reviewers. For each paper, Sacha can access the details of the comments and marks before choosing one of the following options for the paper: “accept”, “reject” or “to discuss”.

b) The website forwards the signatures associated with accepted papers to the AuthorOpeningManager and the signatures associated with the comments and marks to the ReviewerOpeningManager located on the university server. The anonymity of the authors and reviewers of accepted papers is now lifted. Therefore, the program chair can contact directly the authors to tell them that their papers are accepted and to send them their comments and marks. However, the program chair must first verify that there is no conflict of interest between the authors and reviewers of accepted paper. This step is important to avoid the possibility that a reviewer has deliberately cheated by reviewing a paper that he has submitted.

b') The website processes each rejected paper by publishing the following message

$$\{id_p, Enc([comments, marks], pk_p)\}$$

encrypted with the public key $pk_p$ of this paper. Since only the legitimate author has knowledge of the corresponding secret key $sk_p$, only him can decrypt the reviews related to his paper (see Figure 11). The website then deletes all information about the paper ($id_p$, files, comments and especially signatures) in order to preserve the anonymity (and also reputation) of the author(s).

b'') The website forwards the signatures of the reviews related to borderline or problematic papers to the ReviewerOpeningManager located on the university server. The anonymity of the reviewers is then lifted so that the program chair and reviewers can discuss and exchange their points of view on the paper before making a final decision. If the paper is accepted (resp. rejected), then the program chair notifies the website to proceed to step (b) (resp. (b')) for this paper. This process is illustrated in Figure 2.

c) At the request of the program chair, the website sends an email to each author to notify them that a decision has been made on their paper. This email tells them that if they have not already received an email from the program chair to the effect that their paper has been accepted, then they are invited to consult the published list of encrypted comments for rejected papers. Note that the website has no way to know which authors have had accepted papers, and furthermore some authors could have both an accepted and a rejected paper, which is why the website has to send this email to all authors.

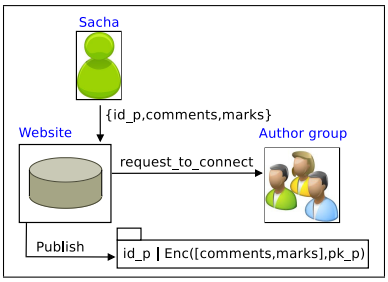


Figure 11: Publication of comments and marks for rejected papers.

# 6 Security Analysis and Implementation

## 6.1 Security and privacy analysis

In this section, we analyse the security and privacy of P3ERS, according to different adversary models personified by the program chair, the website, an author, a reviewer and an external hacker. We assume throughout this analysis that the cryptographic primitives are secure and composable.

- The main objective of the *program chair* is that the conference goes as smoothly as possible. Nevertheless, he might be tempted to try to learn as much information as possible from his interactions with the website. Therefore, the chair is considered to be *honest-but-curious*. We assume that he will not endanger deliberately the conference by cheating. However, he may attempt to infer the identity of authors and reviewers from the messages he receives. In particular, the program chair may try to learn the list of registered authors or the assignment of papers between reviewers. However, the list of authors is carefully preserved by the AuthorIssuingManager located on the website. It is therefore important that the website and the program chair do not collude. Otherwise, this leads to a complete privacy breakdown of P3ERS. However, the program chair can link the assignment of papers between reviewers because there is a direct bijection between the skill forms and the reviewers (*i.e.*, one reviewer has only a single form). This issue is discussed in more details in Section 7.

- The *website* is assumed to be a semi-trusted third party (*i.e*, honest-but-curious). In particular, we consider that the website has no incentives to manipulate the review process. Furthermore, the risk that this would be discovered in the long term cannot be taken by the website because this would ruin its reputation. However, the website may be interested in mining the conference data it has gathered, possibly with the intention of selling it to third parties. In order to protect the privacy of conference participants, it is therefore important to limit the information that can be inferred by the website.

- An *author* can be either honest-but-curious or malicious. For instance, he may want to learn the identity of the reviewers that will be assigned to his paper. This is impossible because once an author has submitted a paper, he no longer has any interaction with the website until the final decision. Moreover, the identity of the reviewers are not attached to the reviews that he receives. A malicious author might want to corrupt the system by sending fake papers on behalf of another author. This threat is prevented by the fact that all submissions are signed with a valid group signature, thus ensuring the property of non-repudiation if anonymity is lifted. Furthermore, rejected papers are never de-anonymized. Therefore, a malicious author has no incentive to submit a "bad" paper on behalf of another author because this would not impact his reputation!

- A *reviewer* can be either honest-but-curious or malicious. For instance, he might try to learn the identity of the authors of papers he reviews. This attack is impossible because no entity in the system knows the correspondence between the identifier of the paper and its author (not even the website) until the final de-anonymization is

carried out by the AuthorOpeningManager located on the university server. Moreover, the list of authors is located on the website and therefore the reviewer has not access to it. A malicious reviewer might want to send fake evaluations on behalf of another reviewer to overrate or underrate a particular paper (for instance a paper he has submitted). This threat is prevented by the fact that all submissions are signed with a valid group signature and therefore anonymity can always be lifted.

- An *external hacker* (unregistered user) cannot perform any attack on the system since all submissions must be signed under a valid group signature. Data that transits over the communication channels cannot provide any information to a hacker who sniffs those channels. He can only have access to public information (public keys of the various managers).

A collusion between an author (Alex) and a reviewer (Randy) has the potential to undermine the security of the system. For instance, Alex and Randy can collude to increase the acceptance probability of Alex's paper. In this situation, Alex does not declare Randy in the blacklist associated to his paper and Randy tries to be chosen as reviewer for this paper in order to register a positive review. Nevertheless, the program chair may notice the collusion if the paper is accepted when the anonymity of authors and reviewers is lifted. Note, however, that this threat is not facilitated by our approach, as it can be attempted in a similar way with a more conventional peer review system. Conversely, if Randy is in personal conflict with Alex, he can try to underrate his work. This threat is more serious because the author and the reviewers of a rejected paper will never be de-anonymized. However, Alex can prevent this by adding Randy to his blacklist.

Thereafter, we briefly describe how P3ERS fulfils the desiderata of a privacy-preserving peer review system (as listed in Section 2).

- *Blindness*. The system ensures the *triple blind* property thanks to the use of the group signature scheme. Indeed, all submissions to the website are carried out via an anonymous channel and under a group signature, thus preserving the anonymity of the source of a message (*i.e*, papers, reviews or other actions). The only entities that can lift the anonymity are the OpeningManagers located on the university server. However, anonymity lifting requires the availability of a signed message, all of which are in possession of the website until papers are accepted (or wrongdoing is suspected). Therefore, the anonymity of the authors and reviewers is protected provided the program chair and the website are not in collusion. From the point of view of reviewers, they do not know the authorship of the papers they review. From the point of view of authors, they only learn that their paper has been reviewed by members of the program committee (this is unavoidable in order to deal with potential conflicts of interest). The program chair himself only knows the list of reviewers (*i.e*, the program committee) but not the list of authors nor how papers have been assigned to particular reviewers.
- *Unlinkability*. The use of a group signature scheme prevents the possibility of linking two different submitted papers to the same author. More precisely, even if an author has submitted two (or more) papers, it is impossible to link these submissions to the same entity unless the anonymity is lifted either because the papers are accepted or wrongdoing is suspected. However, the unlinkability property of reviewers against the program chair does *not* hold because we associate a single skill form to each reviewer. This issue is discussed in Section 7.

- *Reputation preservation*. The reputation of the author(s) is preserved by the fact that the group signatures associated to the rejected papers, which are stored on the website, are never sent to the OpeningManagers located at the university. Moreover, the website erases all traces of rejected papers and their signatures once the decision phase is finished.

- *Anonymity revocation* and *Non-repudiation*. Our improvement on the generic group signature scheme, with its revocation features, ensures unconditional full traceability of the signatures. In particular, a valid signature can always be opened by the OpeningManager, thus revealing (and proving) the identity of the signer, even after he has been revoked from his group (authors or reviewers), should this be the case.

Now, we analyse our proposed system P3ERS in the light of the STRIDE threat model, an acronym for the six threat categories that we list below, which is used extensively to evaluate the security of software systems [11]. After reviewing the model, we describe how well P3ERS holds up to it.

- ***S**poofing identity*. Spoofing occurs when an attacker successfully impersonates an authorized user of a system even though he does not possess the corresponding credentials.

- ***T**ampering with data*. Data tampering occurs when an attacker can modify, add, delete, or reorder data in the system.

- ***R**epudiation*. Repudiation occurs when a user can deny having performed an action while this action has taken place but there exists no proof that the action was indeed performed.

- ***I**nformation disclosure*. Information disclosure occurs when some information is exposed or leaked to an unauthorized user.

- ***D**enial of service*. A denial-of-service attack results in the service or system becoming unavailable for honest users. Denial-of-service attacks are often easy to accomplish, yet it is difficult to guard against them.

- ***E**levation of privilege*. Elevation of privilege occurs when an unprivileged user or attacker gains higher privileges in the system than he was authorized originally.

We evaluate P3ERS using this methodology, again assuming the security of the underlying cryptographic primitives.

- *Spoofing identity*. The architecture of P3ERS is robust against this type of attack. Indeed, only the genuine author or reviewer can actually sign under his own name because he is the only one with knowledge of the corresponding secret key $msk$. Moreover, the website cannot impersonate another participant because it too has only knowledge of the public signature verification keys.

- *Tampering*. In P3ERS, an author might want to tamper with the reviews or papers of other authors, but this attack is not possible because we use a signature mechanism in order to ensure the authenticity and integrity of the data uploaded on the website (*e.g.*, papers, skill forms and reviews). Indeed, each of the data uploaded has a pair of

public and secret encryption keys $\{sk, pk\}$ associated to it, thus ensuring the linkability of the actions performed by one author (respectively a reviewer) when he wants to modify his paper (respectively his skill form and his reviews).

- *Repudiation*. P3ERS is protected against this threat because all submissions (papers, skill forms, evaluations) are signed with a revocable group signature scheme. Therefore, all actions performed by a participant can potentially be traced if the anonymity of his signature on this action is lifted, even retroactively (revoked user).

- *Information disclosure*. This threat can occur during several phases of the conventional peer review process. Even though it is easy to protect the identity of reviewers from the authors using conventional techniques, our approach can do more. For instance, P3ERS keeps secret the list of members of the author group throughout the process (assuming that the website is not in collusion with the program chair).

- *Denial of service*. P3ERS is less susceptible to denial-of-service attacks than traditional systems because only registered authors and reviewers may submit papers and reviews. All the other submissions are immediately rejected following the failed verification of the required group signature.

- *Elevation of privilege*. This type of attack cannot occur in P3ERS because the different entities involved in the review process have non-hierarchical roles and the roles of authors, reviewers and program chair are clearly delimited.

## 6.2 Implementation

An implementation of P3ERS, as described in Figure 12, requires the use of secure and anonymous communication channels between participants.

- A secure channel between the users and the website is needed in order to ensure the confidentiality of all communications. For this purpose we make use of a standard HTTPS connection implementing the "simple TLS handshake" mode.

- An anonymous channel on top of the secure channel is needed to ensure the anonymity of relevant communications. This need is obvious when authors and reviewers submit papers and reviews. Furthermore, the signature algorithm `Sign` requires the verification of the member revocation list on the RevocationManager to update the membership certificate. Thus, anonymity could be compromised by correlations existing between the entity that has accessed public folders and the one that has signed a message. For efficiency and security reasons, we rely on web services and we make them pass through the Tor anonymous communication network to ensure anonymity when appropriate [9].

A prototype of this implementation is currently under development in Java, chosen because the version of Java EE technologies available in August 2012 offers the required components:

- Web technologies to design the website interface through the use of Struts2 framework and its modules (Portlet, JSP, JavaMail),

- the JDBC to access the databases,

- the Java Web Services for communication between the website and the university servers (JAX-RPC).

The chosen application container is Apache Tomcat for its relatively light weight, open-source philosophy and ease of use. We developed the client side using a Firefox extension in order to manage the various keys directly in the user browser in a manner transparent for the user. The Firefox extension does not enable to run directly Java programs and components but we use the functions of the group signature library as applets loaded and executed on the client side.

The unsigned Java applets are based on the sandbox security model, which means that they do not have access to the user's computer and cannot communicate with other servers than the host they came from[9]. We chose to use the applets as black boxes as the Firefox extension enters parameters as input and receives results as output. Note that applets are executed on the client side. As explained in Section 5.2 (bidding phase), applets can serve to implement the independent entity used to manage the conflicts of interest (provided we do not insist on the peace of mind that only secure two-party computation could provide): given the name of the reviewer and the author's blacklist, it can decide whether or not to replace by "no" the reviewer's wish to review a paper without telling anyone that it has done so. We plan to release our implementation of P3ERS as open-source once it is polished and finalized.

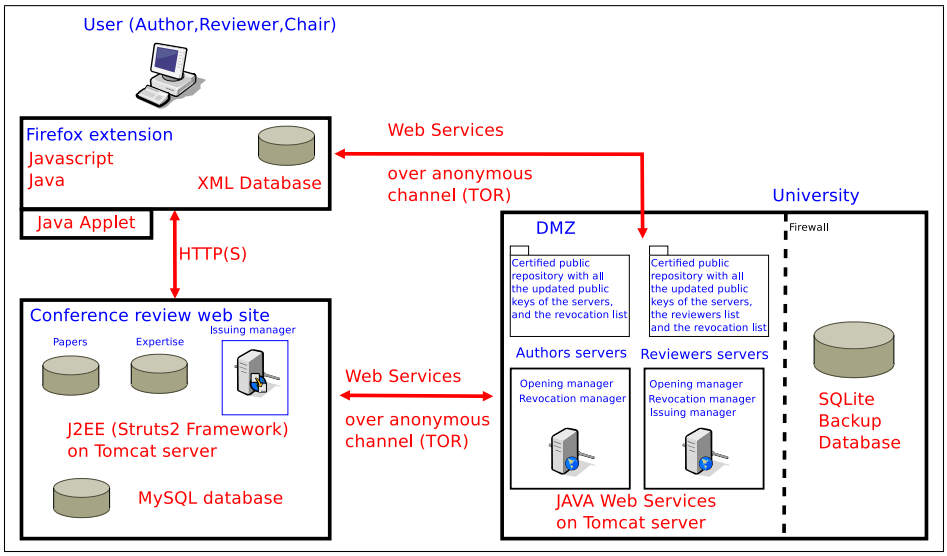


Figure 12: Implementation of P3ERS.

[9] http://download.oracle.com/javase/tutorial/deployment/applet/security.html

# 7 Discussion on the Linkability of Reviewers

In P3ERS, each reviewer is associated with a single skill form containing his areas of expertise, which may potentially play the role of quasi-identifier if the combination of these skills is unique (or almost unique) at the reviewer population level. In particular, this may lead to the following inference attacks that can be carried out by the program chair:

1. The program chair can try to match each skill form with a particular reviewer and use it as a quasi-identifier to de-anonymize him.
2. Our decision to associate a single skill form to each reviewer defeats the desirable property of unlinkability for the reviewers. Indeed, even if the program chair cannot de-anonymize the reviewers, he is responsible for assigning the papers to a particular reviewer with skill form identifier $id_r$, and therefore he knows that all these papers have been assigned to the same entity. Note that it would be difficult for the program chair to balance the work equally between reviewers otherwise.

Initially, we had designed an architecture in which each reviewer had multiple skill forms in order to separate the skills in different fields, thus protecting the unlinkability and preventing an attack by intersection on the areas of expertise. However, this approach was abandoned because it could lead to the following important biases and security problems.

- It could lead to a bias in the assignment of papers to reviewers. The system preserves the anonymity of the reviewers in order to ensure the objectivity of the process and it tries to assign the evaluation of a paper anonymously to the most qualified person. However, if the paper covers several topics, it is desirable that it be evaluated by a reviewer having simultaneously those areas of expertise. This would be difficult to achieve if the program chair cannot link two complementary forms to one and the same reviewer.
- A more serious problem that could occur if each reviewer had distinct and unlinkable skill forms is that the program chair could assign the same paper multiple times to the same reviewer! This vulnerability could potentially be used by a malicious reviewer to manipulate the decision of the program chair concerning a paper by sinking it with multiple negative reviews. This would not be detected because the program chair cannot determine if two reviews have been written by the same person in the case of rejected papers since neither their authors nor reviewers are ever de-anonymized. (This ploy could not be used to *increase* the chance for a paper to be accepted, however, since it would be detected upon de-anonymization following acceptance.)

Note that the program chair cannot link papers that have been submitted by the same author in this way because each submission results in different sets of encryption keys $\{pk_p, sk_p\}$ (one for each paper). Hence, the unlinkability of authors is preserved not only against the reviewers but also against the program chair.

# 8 Conclusion

The peer review paradigm remains the most widely used process for the evaluation of scientific work in the academic community. However, current management systems can

induce a bias into the review process as they do not fully protect the privacy of the participants (authors and reviewers), thus raising the question of objectivity of the review process, and also possibly causing damage to the reputation of some participants whose papers have been rejected. Consequently, we have developed P3ERS, a privacy-preserving peer review system ensuring the anonymity of the authors and reviewers during the review process mainly through the use of a group signature scheme. All the basic functionalities of a conference review system are considered (author and reviewer registration, paper submission, bidding phase, conflict of interest management, assignment of papers, review submission and publication of the final decision), while protecting the privacy of the different actors, thus increasing the fairness of the review process. Even though we have analysed the theoretical properties of the P3ERS architecture, we still need to evaluate the applicability of our system through its use during the management of a real conference.

We leave for future work the improvement of our approach on the management of conflicts of interest between authors and reviewers as well as on the anonymity and unlinkability of reviewers as seen by the program chair. An important issue that we have not addressed is the need to handle external reviewers (also known as subreviewers), who are not members of the program committee but could nevertheless be asked to help the committee with occasional reviewing. We also leave for future work the design of other application domains using our generic library of group signatures. For instance, we could use similar techniques in the context of e-learning to better protect the privacy of the various actors. In particular, one could envision to use P3ERS to assign copies to graders while maintaining their anonymity as well as the anonymity of students.

## Acknowledgements

E. A. is supported in part by Canada's Natural Sciences and Engineering Research Council (NSERC). G. B. is supported in part by NSERC, the Canada Research Chair program and the Canadian Institute for Advanced Research. S. G. is supported in part by the INRIA Project Lab CAPPRIS (Collaborative Action on the Protection of Privacy Rights). We have benefited from scientific discussions with Mark Ryan. Ala Eddine Gandouz helped us finalize some of the figures.

## References

[1] E. Aïmeur, G. Brassard, J. M. Fernandez and F. S. Mani Onana. ALAMBIC: A privacy-preserving recommender system for electronic commerce. *International Journal of Information Security* **7**(5):307–334, 2008.

[2] M. Arapinis, S. Bursuc and M. Ryan. Privacy supporting cloud computing: ConfiChair, a case study. *Proceedings of First International Conference on Principles of Security and Trust, POST'12*, pp. 89–108. Springer, 2012.

[3] I. Blake, G. Seroussi, N. Smart and J. W. S. Cassels. *Advances in Elliptic Curve Cryptography*. Cambridge University Press, 2005.

[4] X. Boyen and B. Waters. Full-domain subgroup hiding and constant-size group signatures. *Proceedings of Public Key Cryptography, PKC'07*, pp. 1–15. Springer, 2007.

[5] J. Camenisch and J. Groth. Group signatures: Better efficiency and new theoretical aspects. *Proceedings of Security in Communication Networks, SNC'05*, pp. 120–133. Springer, 2005.

[6] J. Camenisch and M. Stadler. Efficient group signature schemes for large groups. *Advances in Cryptology – Proceedings of CRYPTO '97*, pp. 410–424. Springer, 1997.

[7] D. Chaum and E. Van Heyst. Group signatures. *Advances in Cryptology – Proceedings of EUROCRYPT '91*, pp. 257–265. Springer, 1991.

[8] N. Desmoulins, S. Canard and J. Traoré. Java implementation of group and blind signatures. *Proceedings of the Workshop on Anonymous Digital Signatures, 3rd International Conference on Trust and Trustworthy Computing*, 2010.

[9] R. Dingledine, N. Mathewson and P. Syverson. Tor: The second-generation onion router. *Proceedings of the 13th USENIX Security Symposium*, pp. 303–320, 2004.

[10] J. Groth. Fully anonymous group signatures without random oracles. *Advances in Cryptology – Proceedings of ASIACRYPT 2007*, pp. 164–180. Springer, 2007.

[11] M. Howard and D. E. Leblanc. *Writing Secure Code*. Microsoft Press, Redmond, WA, USA, 2nd edition, 2002.

[12] T. Isshik, K. Mori, K. Sako, I. Teranishi and S. Yonezawa. Using group signature for identity management and its implementation. *Proceedings of the Second ACM Workshop on Digital Identity Management, DIM '06*, pp. 73–78. ACM press, 2006.

[13] C. Lee, P. Ho and M. Hwang. A secure e-auction scheme based on group signatures. *Information Systems Frontiers* **11**:335–343, July 2009.

[14] Y. Lindell and B. Pinkas. An efficient protocol for secure two-party computation in the presence of malicious adversaries. *Advances in Cryptology – Proceedings of EUROCRYPT 2007*, pp. 52–78. Springer, 2007.

[15] National Institute of Standards and Technology. Recommended elliptic curves for federal government use. NIST, 1999.

[16] K. Sakurai and S. Miyazaki. An anonymous electronic bidding protocol based on a new convertible group signature scheme. *Proceedings of the 5th Australasian Conference on Information Security and Privacy*, pp. 385–399. Springer, 2000.

[17] N. P. Smart and B. Warinschi. Identity based group signatures from hierarchical identity-based encryption. *Proceedings of the 3rd International Conference on Pairing-Based Cryptography*, pp. 150–170. Springer, 2009.

[18] R. Snodgrass. Single- versus double-blind reviewing: An analysis of the literature. *ACM Sigmod Record* **35**(3):8–21, September 2006.

[19] D. Song. Practical forward secure group signature schemes. *Proceedings of ACM Symposium on Computer and Communication Security*, pp. 225–234. ACM press, 2001.

[20] X. Wen, Y. Tian, L. Ji and X. Niu. A group signature scheme based on quantum teleportation. *Physica Scripta* **81**(5):055001, 2010.

[21] S. Xia and J. You. A group signature scheme with strong separability. *Journal of Systems and Software* **60**(3):177–182, 2002.