electronics

MDPI

Article

Design and Implementation of an Anonymous and Secure Online Evaluation Protocol

Nikolaos Petrakos [1], Stefanos Monachos [1], Emmanouil Magkos [2,*] and Panayiotis Kotzanikolaou [1]

[1] Department of Informatics, University of Piraeus, GR-18534 Piraeus, Greece; npetrakos@unipi.gr (N.P.); stefanosmonachos@gmail.com (S.M.); pkotzani@unipi.gr (P.K.)

[2] Department of Informatics, Ionian University, GR-49100 Corfu, Greece

* Correspondence: emagos@ionio.gr; Tel.: +30-26610-87704

Received: 21 July 2020; Accepted: 24 August 2020; Published: 1 September 2020

Abstract: Course evaluations have become a common practice in most academic environments. To enhance participation, evaluations should be private and ensure a fair result. Related privacy-preserving method and technologies (e.g., anonymous credentials, Privacy Attribute-Based Credentials, and domain signatures) fail to address, at least in an obvious way, the minimal security and practicality requirements. In this paper, we propose, evaluate, and implement an efficient, anonymous evaluation protocol for academic environments. The protocol borrows ideas from well-known and efficient cryptographic approaches for anonymously submitting ballots in Internet elections for issuing one-time credentials and for anonymously broadcasting information. The proposed protocol extends the above approaches in order to provably satisfy properties such as the eligibility, privacy, fairness and verifiability of the evaluation system. Compared to the state of the art, our approach is less complex and more effective, while security properties of the proposed protocol are verified using the ProVerif cryptographic protocol verifier. A web-based implementation of the protocol has been developed and compared to other approaches and systems.

Keywords: anonymity; fairness; cryptography; security; evaluation protocol; ProVerif

1. Introduction

Presently, most academic institutions measure teaching quality throughout their academic programs by systematically performing and supporting student course evaluations. To increase participation, evaluations should guarantee the privacy of the evaluators, while being fair for the teacher (i.e., in that only students that have enrolled the course and attended a minimum number of lectures). During the Covid-19 pandemic, the education sector, across the globe, was forced to adopt remote and online learning technologies, making the process of evaluating the effectiveness of online learning, in all educational stages, quite important [1]. Typical approaches, which involve students filling a web-based questionnaire after being authenticated to the department's or university's MOOC platform fail to protect the privacy of the evaluator or cannot be trusted [2]. Related privacy-preserving methods and technologies (e.g., anonymous credentials [3–5]), Privacy Attribute-Based Credentials [6], domain signatures [7], fail to address, at least in an obvious way, the minimal security and practicality requirements.

Our Contribution. The purpose of this work is to propose and implement an anonymous and secure online course evaluation protocol, which ensures the privacy of the evaluators, provides a fair result and is practical to implement in real education environments. To do that, it uses and extends concepts and ideas from [8,9] for polling over public networks using blind signatures to ensure the privacy of the evaluator and eligibility of submitted evaluations. The proposed system consists of

two pillars: The former is for ensuring that only valid users submit their evaluations as well as for delivering a student ticket to be used during their evaluation. The purpose of the second pillar is for users to anonymously submit their evaluation, and for system authorities to certify the validity of the tickets and collect the ratings. We define security properties for an anonymous evaluation protocol and mathematically analyze the proposed protocol for its validity, explaining all the security issues that may arise, regarding the anonymity of users. The security properties of the system are also proven using ProVerif, a formal protocol security analyzer. A web-based proof of concept implementation of the proposed protocol has been performed and detailed implementation results, concerning the execution time for each phase of the protocol, are presented. Our simulation results show that the proposed protocol can be practically implemented for several hundreds of evaluators.

Main phases of the protocol-simplified. At a high level, the proposed scheme is implemented in three phases. In Phase A (Figure 1), a student, after proving to an authority (Issuer) her eligibility to attend a specific course, gets a blinded credential. The student unblinds and uses this credential as a ticket to attend, in an untraceable manner, the course in Phase B (Figure 2), where she gets a proof for each attendance, issued in the classroom by a second system authority (Lecturer). In Phase C (Figure 3), at the end of the semester, or after reaching the required threshold of attendances, the student uses the attendance proofs to anonymously submit the final evaluation to a third system authority (Questionnaire Server).

Structure. This paper is structured as follows: In Section 2, related work in the area is presented, while in Section 3 the details of the proposed anonymous online evaluation protocol are given. Section 4 analyzes the security properties of the protocol, and Section 5 presents implementation results. Section 6 concludes the paper.

2. Related Work on Anonymous Evaluation Systems

In the privacy ABC setting [6], users are issued cryptographically unlinkable pseudonyms, while using the same secret key (typically generated by some trusted hardware), to authenticate to the same or different online service providers. In settings where a specific form of linkability is allowed, e.g., anonymous opinion polls, a user may be issued a scope-exclusive pseudonym, unique for the user's secret key and a given scope string. The user is also issued one or more credentials, bound to the user's secret key, where each credential encodes several attributes types and values. During authentication, the user presents a pseudonymous credential that reveals all attributes that are explicitly requested by a Verifier. Privacy ABCs also provide options to add accountability and revocation to misbehaving users. Two prominent implementations of the Privacy-ABC concept are IBM's Identity Mixer [10,11] and Microsoft's U-Prove [12]. The use of Privacy ABCs has been used as the main pilot in the ABC4Trust Project [13–15], and particularly for implementing anonymous evaluation in the academic environment [16–18]. For example, in the pilot study of [18], 45 students of the University of Patras used Privacy-ABC technology for remotely participating in privacy-preserving course evaluation. In the online educational evaluation system of [17], Attribute Based Credentials [15] were employed to allow students to evaluate, anonymously, the competence of the subjects they have selected. Privacy ABCs, as most anonymous credentials instantiations, rely on user's hardware and employ cryptographic constructs which are considered to be heavy with respect to computation and storage [2,19]. Having said that, in recent years there have been attempts to improve their practicality, e.g., [20,21], however much work is still needed before anonymous credentials are considered practical.

In the anonymous evaluation system proposed by Kluczniak et al. [2], an evaluator i registers her standard DSA public signature keys (in the form of $PK_i \leftarrow g^{x_i}$) to the Registration Server (controlled by the University), and then uses her corresponding private keys x_i to authenticate their evaluations. To protect the evaluators' anonymity, the Questionnaire Server receives a list of randomized public keys (i.e., a white list), in the form of $R(PK_i) \leftarrow (g^{x_i})^r$, where the randomizing exponent r is the product of powers $r_j, j = 1, ..., n$ shared between the Registration Server and several n Mixers (controlled bu the students' Council), where $n \geq 1$. Evaluators then sign (using the probabilistic signature scheme) and

(anonymously) submit their evaluations to the Questionnaire Server which uses the white list to verify the signatures. To preclude cheating by the mixing server, a cut-an-choose, interactive zero-knowledge based protocol is run between the mixing servers and an inspection system. Overall, the scheme proposed by Kluczniak et al. [2] is a relatively complex system that achieves most of the security and privacy requirements; however it fails to achieve fairness, i.e., ensure that an evaluator has attended a minimum number of lectures for a specific course. In addition, the assumption that several mixing servers will not collude to violate the evaluators' privacy could be seen as hard.

3. Protocol Description

The proposed anonymous online evaluation protocol is based on known privacy primitives such as blind signatures [22], hash chains [23] and broadcast anonymous communication channels [24]. The protocol consists of four entities: the Student (or Evaluator) S, the Issuing Authority Server I, the Lecturer Server L and the Questionnaire Server Q. The protocol fulfills the anonymous evaluation of a given course i, in three phases. At a high level of abstraction, initially, Student S, after proving that she is a university student who is enrolled in a specific course i, receives from the Issuer Server I a blindly signed eligible attendance ticket for course i. She will use that ticket in phase B where, per each attendance, S will get, in an untraceable manner, from the Lecturer Server L a token that proves she has attended the lecture for course i. In phase C, the final phase, S will anonymously send to the Questionnaire Authority Server Q the evaluation of the course, together with all tokens and tickets that prove that an eligible and fair (i.e., submitted after a threshold of attendances has been reached) evaluation has been submitted. The detailed security properties and description of the protocol are given in the remaining section.

3.1. Desired Security Properties

An anonymous evaluation system should fulfill the following properties [2,13,15]:

1. *Eligibility:* A student should be able to evaluate a university course if and only if: (a) The student belongs to the university, and (b) the student is enrolled in the specific course.
2. *Fairness:* An evaluation is considered fair if and only if: only a student that has attended a minimum number of lectures for the specific course is able to evaluate the course.
3. *Privacy:* An evaluation is considered private if and only if it provides the following properties: (a) Untraceability—A student's evaluation of a specific course should not be traceable to the student's identity. (b) Unlinkability—Evaluations of different courses should not be linkable.
4. *Unforgeability*: An evaluation is considered unforgeable if and only if: (a) A student should not be able to use a credential issued for course i to evaluate course j, where $i \neq j$ or to participate more than once in a course evaluation. (b) A student should not be able to use one or more credentials issued for another honest student. (c) It should not be able for any system entity (students and/or system authorities) to modify an evaluation submitted by a student user. (d) It should not be able for a system authority to submit a fake evaluation on a specific course.

The proposed system will provide the above security properties under reasonable and realistic assumptions for the university environment, as described in Section 4.1.

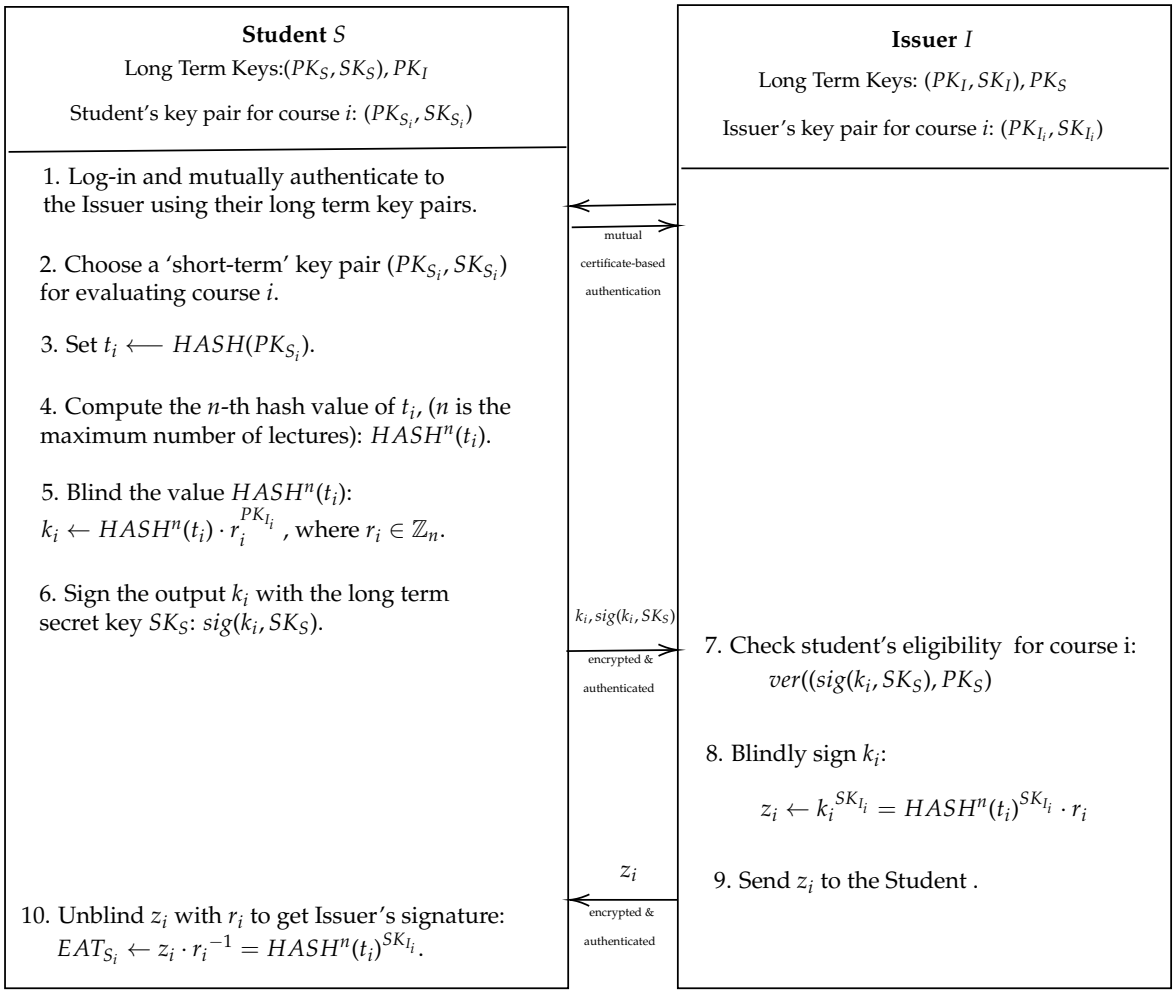


Figure 1. Phase A: Eligible Attendance Ticket (EAT) Issuing Phase.

3.2. Phase A: Issuing Phase-Eligible Acceptance Ticket (EAT)

Phase A is executed by two entities: a student S, acting as the evaluator, and the Issuing Authority server I. Both entities are assumed to possess certified long term key pairs, (PK_S, SK_S) and (PK_I, SK_I) respectively. Long term keys are used by users, during this phase, to authenticate themselves and prove their eligibility to participate in the evaluation process. In addition, for each course i each student creates an independent key pair (PK_{S_i}, SK_{S_i}) for that course. The public key PK_{S_i} will not be revealed until the last phase of the protocol, where it will be disclosed to the Questionnaire Server via an anonymous channel in order to validate, in an untraceable manner, the submitted evaluation.

Phase A consists of the following steps, as depicted in Figure 1. Initially, we assume that all communication in Phase A is done over a mutually authenticated and secure channel. This is trivial to implement by applying a Transport Layer Security (TLS) [25] channel with mutual authentication, by using the Student's and the Issuer's long term keys and relevant certificates. Then, S creates a short term pair key for evaluating a specific course i. Let $HASH$ be a cryptographic one-way hash function. Each student is allowed to run this phase once for each different course. The student S will produce a different value t_i for each course she attends by hashing the short term key PK_{S_i} linked to course i ($t_i \longleftarrow HASH(PK_{S_i})$). Then, S, produces the n_{th} hash of t_i, where n is the maximum number of lectures for that specific course (for example, assume that course i has 10 lectures, then: for $n = 1$, $HASH(t_1) \rightarrow t_1$, for $n = 2$, $HASH(t_1) \rightarrow t_2$, ..., for $n = 10$, $HASH(t_9) \rightarrow t_{10}$). The student also stores all the intermediate values, since they will be later used, in reverse order, in order to attend different lectures of this course. In the next step, S blinds the value $HASH^n(t_i)$ using a random blinding factor r

powered to I's public key for course i: $r_i^{PK_{I_i}}$. Afterwards, S signs the k_{i} with her long term signature key, SK_S and sends it along with k_{i} to I. Then, I checks student's eligibility for course i, by validating S signature on k_{i}. On completion, I signs k_{i} with its secret signature key for course i, SK_{I_i} and sends that value z_{i} to S. S multiplies z_{i} with ${r^{-1}}_{\mathrm{i}}$ and the remainder is the signature of I on $HASH^n(t_i)$, namely the Eligible Acceptance Ticket (EAT) of S for course i. Please note that I does not know the contents of the Eligible Acceptance Ticket because of the blinding process. It is also important to note that I will allow each student to run Phase A only *once for each course*, otherwise students would be able to evaluate many times for each course. This is easy to implement since in Phase A students sign in using their certified long term key pair.

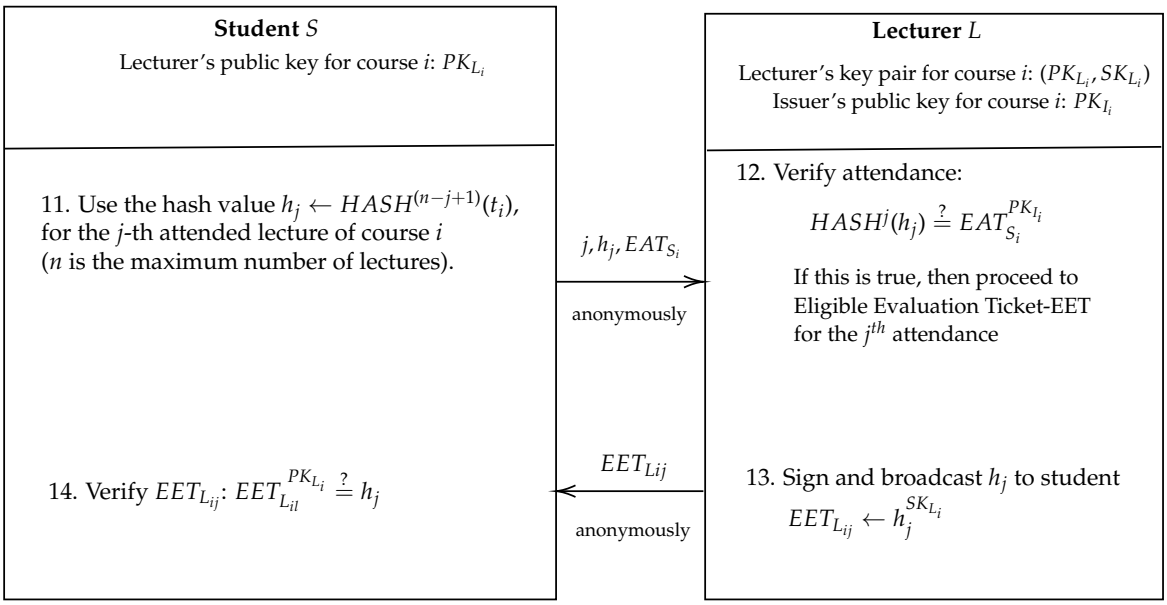


Figure 2. Phase B: Eligible Evaluation Ticket (EET) Issuing Phase.

3.3. Phase B: Eligible Evaluation Ticket (EET)

In Phase B, as shown in Figure 2, two entities participate: S and the Lecturer Server L. L has a long term signature key pair for course i, and is aware of Issuer I's public key for course i. Phase B, whose main goal is to ensure that S has attended the lectures of course i, is completed in the following four steps. Initially, for every lecture that S has attended, lets say, the j-th lecture, she will pick up h_j as the $(n-j+1)$-th hash value of t_i (from Phase A). For example, if $n = 10$ and S attends the 1st lecture of course i, S will use the value $HASH^{10}(t_i)$, if this is his 2nd time, she will use the value $HASH^9(t_i)$, and she will continue this pattern up to the initial value t_i.

In the next step, S sends anonymously to L the values of j, the EAT_{S_i} (from Phase A) and the value h_j. In the verification process, L verifies the attendance data for S. This is done, by hashing h_j up to j times and verifying that the result equals to EAT_{S_i} raised to I's public key for course i, PK_{I_i}. If this verification succeeds, L signs h_j with his secret signature key SK_{L_i} and creates an eligible evaluation ticket-EET for S's j^{th} attendance of course i, $EET_{L_{ij}}$. S verifies $EET_{L_{ij}}$ with the use of L's public key, PK_{L_i}.

3.4. Phase C: Anonymous Evaluation

In Phase C (Figure 3), the participants are the Student S and the Questionnaire Server Q. Q has a long term key pair, and is aware of the public keys of the other system entities of the protocol. Let us assume that S has attended ℓ lectures of course i, and that ℓ is higher than the threshold number required by regulation. Then S adds in a message M_i, the EETs from 1 through ℓ, t_i, EAT_{S_i}, the evaluation form E_i for course i as well as S's public key PK_{S_i} (from Phase A). She signs M_i

with her secret signature key for this course, SK_{S_i}, and sends M_i and its signature to Q via an anonymous channel. The Questionnaire Server Q verifies if the signature is valid, then checks whether $t_i = HASH(PK_{S_i})$ and that $HASH^n(t_i) = EAT_{S_i}$. Q also checks validity of EAT_{S_i} and $EET_{L_{ij}}$ for all $j \leq \ell$, similarly to Phase B. If all checks are true, Q accepts evaluation E_i as valid.

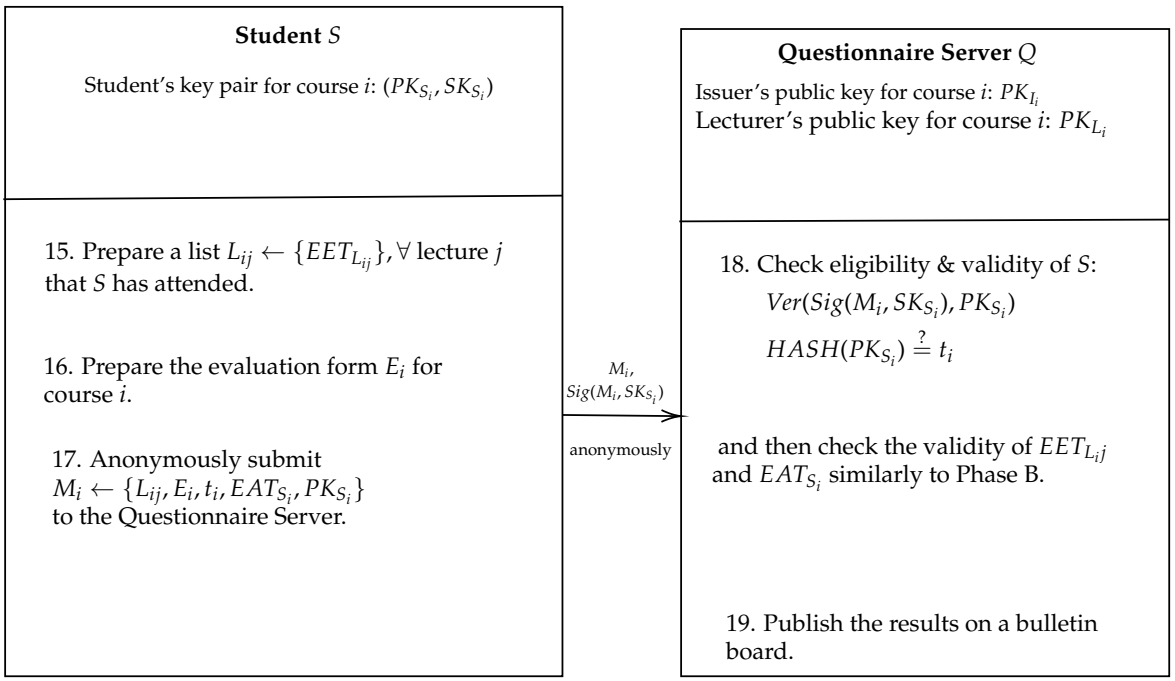


Figure 3. Phase C: Anonymous Evaluation.

4. Security Analysis

To analyze the security properties of the proposed protocol we will define realistic adversaries and security experiments to capture the desired security properties. Then we will analyze the security of the protocol using ProVerif [26] which is a formal protocol security analyzer.

4.1. Assumptions

For the security properties (evaluator eligibility, double evaluation prevention, evaluation unforgeability), we assume that the evaluators (Students) may be colliding malicious users, i.e., they may actively attempt to deviate from the protocol execution to bypass security and may attempt to collide with each other. To support unforgeability against *colluding* malicious evaluators, we assume that all the credentials issued to each evaluator, both long- and short-term credentials, are stored within a secure isolated hardware (e.g., similar to secure wallets [27]—see Note 1 in Section 4.3). For the privacy properties (untraceability, unlinkability, secrecy) we assume a *honest-but-curious* model where entities (Issuer, Lecturer, Questionnaire Server) are expected to comply with the protocol but may attempt to leak private information, such as the students' evaluations. We also assume that the adversary is polynomially bounded and finally that the underlying cryptographic primitives (such as encryption and hash functions) are secure.

4.2. Adversaries

To capture realistic threat scenarios we consider three types of adversaries: $\mathcal{A}^{\mathcal{E}}$ denotes an adversary that captures the capabilities of a malicious evaluator (Student); $\mathcal{A}^{\mathcal{L}}$ denotes an adversary that captures the capabilities of a honest-but-curious Lecturer, and finally $\mathcal{A}^{\mathcal{S}}$ captures the capabilities of a honest-but-curious back-end entity (i.e Issuing server and Questionnaire server). A malicious evaluator $\mathcal{A}^{\mathcal{E}}$ has access to the set $\mathcal{K}_{\mathcal{E}}$ of all the keys available to a legitimate student (such as a long

term key pair PK_S, SK_S, and the short term key pairs PK_{S_i}, SK_{S_i} for course i). In the same way $\mathcal{A}^{\mathcal{L}}$ has access to $\mathcal{K}_{\mathcal{L}}$ denoting all the private keys of a Lecturer, while $\mathcal{A}^{\mathcal{S}}$ has access to $\mathcal{K}_{\mathcal{S}}$ denoting all the private keys of the Issuer and the Questionnaire server. The public keys of all entities are by definition available to all the above key sets of all the adversaries (except from PK_{S_i} as explained in Phase A). Finally, all types of adversaries are able to passively eavesdrop all the communications in all the above phases of the protocol, while $\mathcal{A}^{\mathcal{E}}$ is also allowed to inject messages.

4.3. Evaluator Eligibility

Let Π be the proposed anonymous evaluation protocol. Let PK_S, SK_S be the public/private key pair of a 'target' student S for the course i. Since S is the target, the goal of the adversary is to forge Eligible Attendance Tickets for a student S not controlled by the adversary. We formalize evaluator eligibility by a security experiment $Priv_{\mathcal{A}^{\mathcal{E}},\Pi}(p)$ in which $\mathcal{A}^{\mathcal{E}}$ has access to an oracle $\mathcal{O}^{\mathcal{E}}$ that on input the security parameter p (the length of the private keys used) simulates executions of Phase A (Eligible Attendance Ticket Issuing) of Π. On input $\{\hat{k}, (\hat{k})_{S\hat{K}_S}\}$, produced as described in step 4 of Phase A, the oracle $\mathcal{O}^{\mathcal{E}}$ will output $\{\hat{z}\}$. The adversary will process $\{\hat{z}\}$ to get eligible Attendance Tickets for the target student S. If the output produced by the adversary is valid (i.e., $\hat{EAT}_{S_i} \equiv EAT_{S_i}$) then the experiment will output 1 and the adversary wins. Else, the experiment outputs 0 and the adversary fails.

Definition 1. *Π provides evaluator eligibility if for every polynomially bounded adversary $\mathcal{A}^{\mathcal{E}}$, $\exists$ a negligible function negl such that: $Advantage(\mathcal{A}^{\mathcal{E}}) = Pr[Priv_{\mathcal{A}^{\mathcal{E}},\Pi}(p) = 1] = negl(p)$.*

Theorem 1. *Π provides evaluator eligibility, provided that $\mathcal{A}^{\mathcal{E}}$ has no access to the long-term private key SK_S of the target student S and that the communication in phase A is authenticated and encrypted.*

Proof. In order to execute Phase A and get an eligible attendance ticket, a student S must first log in, using his long term key pair PK_S, SK_S. A student is allowed to run phase A only once for each different course he is registered. Since the communication is authenticated (by using the long-term student key SK_S not controlled by the adversary) the adversary will not be able to initiate an execution of Phase A for the target student S. In addition, since the communication between the Student and the Issuer assumes a secure channel (encrypted and integrity protected) the adversary cannot learn the actual values $\{z_i\}$ corresponding to the actual protocol execution of S and thus learn EAT_{S_i}. □

Note 1: *A collusion attack against the evaluation protocol.* Recall that in our threat model (see Section 4.1) the evaluators (Students) are assumed to be colliding malicious entities. Therefore, Students are allowed to collide with other Students and/or external entities, to break the unforgeability property defined in Section 3.1 as follows: Imagine a malicious eligible Student A who hands over its evaluation credentials, i.e., the EAT ticket and key pairs for a given course i, to another malicious user B (a different Student or an external entity). Then B would be able to attend a specific course and/or submit a evaluation on behalf of A. In fact, this attack could scale, i.e., A could gather evaluation credentials for any number of Bs in order to attack the evaluation system.

A solution based on secure hardware. To prevent such attacks, as stated in Section 4.1, we assume that all the long- and short-term credentials of a Student are stored in an hardware module (e.g., similar to secure wallets [27]) that ensures isolation and secure access of the stored data. This enforces a coarse grained (all-or-nothing) access approach where an imposter B will either have complete access to A's credentials and long-term keys or no access at all. Specifically, short-term evaluation credentials (i.e., EATs and course key pairs) together with long-term keys are stored in a secure hardware module, with the following properties: (a) Secure storage: all keys/are encrypted and integrity protected; (b) Tampering protection: the module is either able to resist invasive attack techniques, or it would erase its stored secrets; (c) Access control: A user authentication protocol is required by the module

to activate any credential (short-term and/or long-term) that is securely stored. The all-or-nothing feature of the proposed approach, would discourage collusion attacks since Students would have to risk disclosure of their long-term secret keys to a colliding malicious evaluator.

4.4. Evaluator Privacy

We formalize evaluator privacy by a security experiment $Priv_{\mathcal{A}^{\mathcal{L},\mathcal{S}},\Pi}(p)$ in which $\mathcal{A}^{\mathcal{L}}, \mathcal{A}^{\mathcal{S}}$ have access to an oracle $\mathcal{O}^{\mathcal{L},\mathcal{S}}$ that on input the security parameter p (the length of the private keys used) simulates executions of Π. $\mathcal{A}^{\mathcal{L},\mathcal{S}}$ gets from $\mathcal{O}^{\mathcal{L},\mathcal{S}}$ a history of simulated protocol executions. For the test, $\mathcal{O}^{\mathcal{L},\mathcal{S}}$ simulates a protocol run for a target student S out of n participating students, using all the public keys of all students. A bit $b = 0|1$ is chosen at random. If $b = 1$ the oracle is given a protocol execution for the actual student S, if $b = 0$ a valid protocol run for another user $\hat{S}$ is given to the oracle. The oracle will output a bit $\hat{b}$. If $b = \hat{b}$ (the oracle is able to distinguish a protocol execution of S from a protocol execution of $\hat{S} \neq S$) then $\mathcal{A}^{\mathcal{L}}$ succeeds to trace an evaluation of a target user and $Priv_{\mathcal{A}^{\mathcal{L},\mathcal{S}},\Pi}(p)$ outputs 1, else it outputs 0.

Definition 2. Π *provides evaluator privacy if for every polynomially bounded adversary* $\mathcal{A}^{\mathcal{L},\mathcal{S}}$, $\exists$ *a negligible function negl such that:* $Advantage(\mathcal{A}^{\mathcal{L},\mathcal{S}}) = Pr[Priv_{\mathcal{A}^{\mathcal{L},\mathcal{S}},\Pi}(p) = 1] = \frac{1}{n} + negl(p)$.

Theorem 2. Π *provides evaluator privacy, provided that the communication channel used in Phases B an C is anonymous and the blind signature scheme used in phase A is secure.*

Proof. In Phase C the Questionnaire server will receive M_i containing $EET_{L_{ij}}$. This has been received anonymously by the student in Phase B. Assume that n students participate in a protocol run of Phase B. Since the communication in Phase B is over an anonymous channel each student participating in Phase B will be n-sender-anonymous. In addition, when a student sends her evaluation E_i in Phase C, the short-term key pair PK_{S_i}, SK_{S_i} chosen by the Student in Phase A will be revealed, so that the Questionnaire server can verify that $t_i = HASH(PK_{S_i})$ and that the evaluation comes from a valid student who has already issued in the past a valid EAT. However, since in Phase A a blind signature scheme is used, the Issuer cannot link each blinded signed credential with a particular student. In addition, since the communication in Phase C is also n-anonymous, it is not possible for $\mathcal{A}^{\mathcal{L},\mathcal{S}}$ to link a particular evaluation with a particular evaluator with probability that is non-negligibly higher than $\frac{1}{n}$. □

4.5. Analysis with Proverif

In order to prove the security properties of the protocol, we use ProVerif, an automated model checker. ProVerif is a powerful cryptographic protocol analyzer, proposed by [26]. It uses the Dolev-Yao adversary model [28]. In that model, the adversary has control over the network. He may be a legitimate participant, he can assign roles and intended partners, he has access to almost all keys, even those that may be discovered during execution. The adversary can pair separate messages and encrypt/decrypt with the right key. The assumption that cryptography is perfect has been used, meaning that the attacker cannot execute a polynomial-time algorithm, and he is limited to apply the cryptographic primitives assigned by the user. The adversary cannot corrupt participants and cannot perform cryptanalysis.

ProVerif uses pi-calculus as the input language, designed for representing processes that interact using communication channels such as the Internet. ProVerif is able to prove reachability properties, meaning terms that are available to the attacker and secrecy of terms can be evaluated with respect to the model. Moreover, the tool can provide correspondence assertions used to capture authentication through relationships between events that can be expressed in the form "if an event ρ has been executed, then event ρ' has been previously executed". For detailed analysis of the architecture and grammar of ProVerif, see Appendixes A and B.

4.5.1. Formal Modelling of the Protocol

The proposed protocol model consists of asymmetric encryption, one-way hash functions, a main process, a Student process and an Issuer process for Phase A. Additionally, it includes a Lecturer and Questioner process for phases B and C. Our main concern is Phase A, where a very essential element of the security properties of the protocol is calculated, the Eligible Acceptance Ticket (EAT).

4.5.2. Student Process

In the student S process the terms t, r_{blind} and m are calculated and transmitted by the student S to the Issuer I via Message 1 $(m, r_{blind})_{SK_{S_i}}$. The ProVerif syntax for student S process is depicted in Figure 4:

```
(*In this process compute *)
let studentS(pkS:pkey, skS:skey, stpkS:pkey, stskS:skey, pkI:pkey, stpkI:pkey) =
                                                    (*Student process*) (*Known keys to S*)

 in (c, pkX: pkey);                                    (*Receiving Get Challenge Message from the Issuer*)
 let t= h(key_to_bitstring(stpkS)) in                  (*Generates t*)
 new r : bitstring;                                  (*Generates r*)

 let r_blind = aenc(r, stpkI) in                       (*Calculate blind value of r*)

 let m = h(t) in                                     (*Generate m, the hash of t *)
 (*Transmit Message 1 to channel c*)
out (c , (aenc (concat (m ,r_blind , (adec((m,r_blind), skS))), pkI)));
 event acceptsStudent(concat1 (m ,r_blind));          (*Student accepts, typically before the student sends its last
message*)

 in (c, L:bitstring );
  if (concat1 (m, r) = adec(aenc (L,stpkI ), skS)) then event termStudent(concat1 (m ,r_blind), stpkS).
(*when student terminates (typically at the end of the protocol*)
```

Figure 4. Student Process.

4.5.3. Issuer Process

In this process the EAT calculated and transmitted by the Issuer via channel c, through Message 2. The ProVerif syntax for the issuer I process is shown in Figure 5:

```
let issuerI(pkI:pkey, skI:skey, stpkI:pkey, stskI:skey, pkS:pkey, stpkS:pkey)=   (*Issuer process*)(*Known keys to I*)
(*Receive Message1*)
 out(c, pkI);                                  (*Get Challenge message is transmitted to channel*)

 in(c, (n : bitstring, p : bitstring, q : bitstring));  (* Received from S the signed value of the blinded ticket *)
 event acceptsIssuer(p, stpkS);                (*Issuer accepts, typically before the issuer sends its last message at
Phase A*)
 (* Decryptions tasks*)
 let z = adec ((n,p,q),skI) in                 (*Decrypt the information*)
  let v = adec (p, stskI) in                   (*Decrypt the information*)
  let w = aenc (q, pkS) in                     (*Check the signature*)

 let (=n, =p ) = aenc ( w, pkS ) in
  (*Transmit Message 2  phase A*)
 out (c, (aenc(adec(concat1(n,p), stskI), pkS)));
  (*Transmit the Eligible Acceptance Ticket-EAT*)
   event termIssuer(p).                                  (*when issuer  terminates (typically at the end of
the protocol*)
```

Figure 5. Issuer Process.

4.5.4. Main Process

In the main process the public keys of the Student and the Issuer are created and transmitted to the network via the public channel. The coding for the main process is depicted in Figure 6.

```
process
(*construct the private short term keys skS,skI long term keys for principals S,I *)
 new skS:skey; let pkS = pk(skS) in out (c,pkS); (*construct the long term keys of S*)
 new skI: skey; let pkI = pk(skI) in out (c,pkI); (*construct the long term keys of I*)
 new stskS:skey; let stpkS = pk(stskS) in out (c,stpkS); (*construct the long term keys of S*)
 new stskI: skey; let stpkI = pk(stskI) in out (c,stpkI); (*construct the long term keys of I*)


 ( (!studentS(pkS, skS, stpkS, stskS, pkI, stpkI)) | (!issuerI(pkI, skI, stpkI, stskI, pkS)) )
```

Figure 6. Main Process.

4.5.5. Results Analysis

The security goals of the protocol like secrecy, authentication and data integrity have been achieved. in order to prove that we used two attacker queries to verify the secrecy of message *s* and *m*, query attacker(s) and query attacker(m) respectively. To verify the authentication between the parties we have used the following queries:

```
(* Injective Correspondance*)
query x:bitstring, y:pkey; event(termStudent(x,y))==>event(acceptsIssuer(x,y)).
query x:bitstring; inj-event(termIssuer(x))==>inj-event(acceptsStudent(x)).
```

The output of ProVerif is the following:

```
Process 0 (that is, the initial process):
{1}new skS: skey;
{2}let pkS: pkey = pk(skS) in
{3}out(c, pkS);
{4}new skI: skey;
{5}let pkI: pkey = pk(skI) in
{6}out(c, pkI);
{7}new stskS: skey;
{8}let stpkS: pkey = pk(stskS) in
{9}out(c, stpkS);
{10}new stskI: skey;
{11}let stpkI: pkey = pk(stskI) in
{12}out(c, stpkI);
(
  {13}!
  {14}in(c, pkX: pkey);
  {15}let t: bitstring = h(stpkS) in
  {16}new r: bitstring;
  {17}let r_blind: bitstring = aenc(r,stpkI) in
  {18}let m_1: bitstring = h(t) in
  {19}out(c, aenc(concat(m_1,r_blind,adec((m_1,r_blind),skS)),pkI));
  {20}event acceptsStudent(concat1(m_1,r_blind));
  {21}in(c, L: bitstring);
  {22}if (concat1(m_1,r) = adec(aenc(L,stpkI),skS)) then
  {23}event termStudent(concat1(m_1,r_blind),stpkS)
) | (
  {24}!
  {25}out(c, pkI);
  {26}in(c, (n: bitstring,p: bitstring,q: bitstring));
```

```
  {27}event acceptsIssuer(p,pkS);
  {28}let z: bitstring = adec((n,p,q),skI) in
  {29}let v: bitstring = adec(p,stskI) in
  {30}let w: bitstring = aenc(q,pkS) in
  {31}let (=n,=p) = aenc(w,pkS) in
  {32}out(c, aenc(adec(concat1(n,p),stskI),pkS));
  {33}event termIssuer(p)
)
– Query not attacker(s[]) in process 0.
Completing...
Starting query not attacker(s[])
RESULT not attacker(s[]) is true.
– Query not attacker(m[]) in process 0.
Completing...
Starting query not attacker(m[])
RESULT not attacker(m[]) is true.
– Query event(termStudent(x,y)) ==> event(acceptsIssuer(x,y)) in process 0.
Completing...
Starting query event(termStudent(x,y)) ==> event(acceptsIssuer(x,y))
RESULT event(termStudent(x,y)) ==> event(acceptsIssuer(x,y)) is true.
– Query inj-event(termIssuer(x)) ==> inj-event(acceptsStudent(x)) in process 0.
Completing...
Starting query inj-event(termIssuer(x)) ==> inj-event(acceptsStudent(x))
RESULT inj-event(termIssuer(x)) ==> inj-event(acceptsStudent(x)) is true.
——————————————————————————
Verification summary:
Query not attacker(s[]) is true.
Query not attacker(m[]) is true.
Query event(termStudent(x,y)) ==> event(acceptsIssuer(x,y)) is true.
Query inj-event(termIssuer(x)) ==> inj-event(acceptsStudent(x)) is true.
——————————————————————————
```

ProVerif attempts to prove that a state in which a property is violated is unreachable. It provides three kind of results: RESULT[Query] is true: the query is proved, there is no attack, RESULT[Query] is false: the query is false, an attack has been discovered towards the desired security property, and the RESULT[Query] cannot be proved: It cannot prove that the query is true but cannot find an attack to prove that the query is false either.

Based on the output of the verifier results, our protocol satisfies the following security goals: Secrecy, because values s and m are only known to the student S and the issuer I, since the specific queries are proved and there are no attacks. Authentication of student to the issuer and vice versa, based on the validation that the queries event and inj-event are true. The student S is willing to share her secret only with the issuer I, and as a result, if she has completed the protocol, then she believes she has completed it with I and then authentication holds. On the other hand, the issuer I is willing to run the protocol with any eligible student S, and therefore at the end of the protocol she only expects authentication of S to I to hold, if she believes S was her interlocutor. At the end of the protocol, the student thinks she executed the protocol with the issuer and also the issuer thinks he executed the protocol with the student.

5. System Evaluation

To evaluate the proposed system, a proof of concept web application was implemented in Java using the Spring Model-View-Controller (MVC) framework. The security functionality was based on the Spring Security and the WebCrypto API. Other technologies that were used include Hibernate, JQuery, Log4j2 and SonarQube. All actions that refer to the entity "Student" will be performed in the front end of each respective web application. Therefore, the students only need a web browser to participate in the evaluation process. To simulate the anonymous communication channel in phases *B* and *C* we utilize the Broadcast Anonymous Routing (BAR) protocol [24] using a single Broadcast Anonymous Server [29] offering *n*-anonymity for the sender, since this is sufficient for our environment. Obviously students who are attending/evaluating a course disclose their identity to the lecturer/questionnaire server and their goal is to achieve sender *n*-anonymity when broadcasting their evaluations.

5.1. Implementation

The basic architecture of the system is depicted in Figure 7. The requests will be handled using the MVC architecture implemented in Java SE 12 and the Spring Framework. All connections to the common database are using Hibernate and Java Persistence API (JPA), to achieve an Object Relational Mapping. Implementing all database connectivity with JPA also prevents data corruption from incomplete transactions. The connections between a student's browser and the back-end modules, need to be encrypted and trusted. This is achieved by using the TLS protocol for all the back end system connections. Furthermore, Spring Security will ensure the correct usage of sessions, preventing most of the session-based attacks.

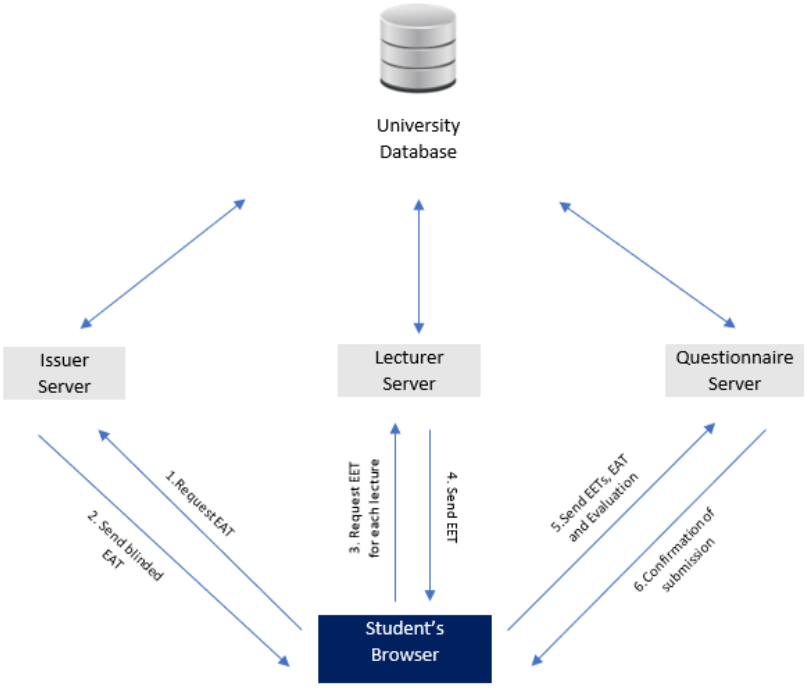


Figure 7. A high level description of the implementation architecture.

On the front end, all cryptographic operations are computed using the Web Crypto API. The scripts that run on the student's browser use mostly JQuery. The back end applications use Bootstrap 4 to be responsive, which means that the user interface is mobile and tablet friendly. All the frameworks above, were downloaded directly from the Maven repositories, to ensure that no jar

file is corrupted or injected with malicious code that could compromise the privacy of the *EAT*s, *EET*s or the evaluations themselves.

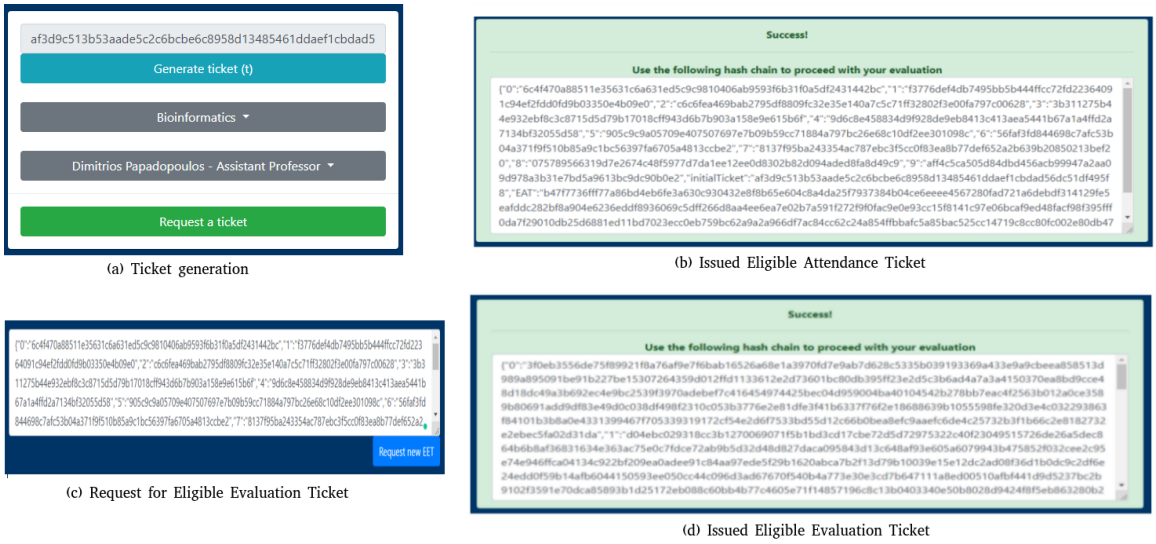
All the generated key pairs are based on RSA and are stored in Java keystores using naming conventions as the alias of the pair. Each of the back end modules use their own keystores and their own naming conventions.

The applications can be deployed to any Apache Tomcat Web Server. All connections between a student's browser and the back end modules are also TLS encrypted to prevent eavesdropping and man-in-the-middle attacks. Also, a token is used to prevent Cross-Site Request Forgery. Finally all session-based attacks are prevented using Spring Security. By using the default filters, the sessions will have unpreventable identifications and be invalidated when needed.

5.2. System Evaluation Scenario

The student, after authentication, can use the GUI interface to generate the short-term key per course. The hash value of this key will be used as the starting point of the hash chain. Then, the student can issue the EAT by simply choosing the corresponding instructor and course, as shown in Figure 8. The result is a JSON object that contains a key-value pair. The key is the number of the lecture that the hash is representing (e.g., key 0 is the first lecture). This object also contains the 'initialTicket' which is the first hash of the short-term key, and the EAT, which is the last hash of the chain.

After each lecture, the student has to use this hash chain in order to get her EET of the respective lecture. This hash chain needs to be inserted in a simple text box, in order to be transmitted through the secure channel and get the result of the new EET by the lecturer. Figure 8 also demonstrates the issuing of the EAT. The interface of the 'Lecturer' module is similar to that of the 'Issuer' module, so that it can be easily understood by the students.



(a) Ticket generation

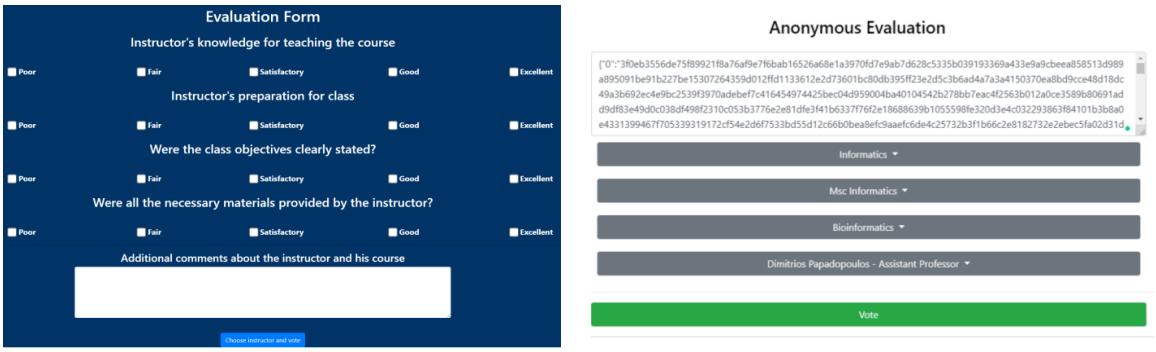
(b) Issued Eligible Attendance Ticket

(c) Request for Eligible Evaluation Ticket

(d) Issued Eligible Evaluation Ticket

Figure 8. Request and Issuing of EAT and EET.

In the final phase the student, after filling the evaluation form, needs to present this JSON Object to the Questionnaire Server, in order to submit the evaluation. Figure 9 depicts the evaluation form used, as well as the form that is used to select the course and lecturer under evaluation.



(a) Evaluation form (b) Anonymous evaluation

Figure 9. Anonymous evaluation.

5.3. Performance

We used JMeter v.5.3 to calculate the execution time for each phase of the proposed protocol. The number of concurrent threads allowed by the Tomcat server was set to the default (250 concurrent threads). The results could change based on this configuration's value since when the tomcat container is handling more than 250 requests, the rest of the requests are getting queued and are not being processed before another thread is finished by serving the response.

For the Issuer, we measured the time needed for checking the student's eligibility, the time needed to extract the information from the keystores and the calculation of the EAT. For the Lecturer, we measured the time needed for the anonymous (BAR) channel to transmit the response, as well as the calculation of the EET for every participation in the lecture by the students. Finally for the Questionnaire server the time includes the average time needed for checking the validity of both the EAT and the EETs. As shown in Figure 10 the total execution time grows from around 2.2 s for 100 students to 5.6 s for 400 students. The execution time is reasonable enough to be practically implemented in realistic university environments, without requiring students to possess special hardware. The detailed computation times for each phase are shown in Table 1.

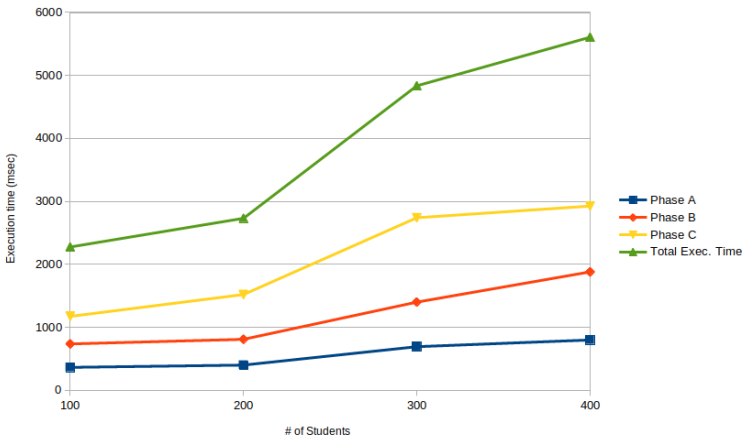


Figure 10. Execution time of the proposed protocol (chart view).

Table 1. Execution time of the proposed protocol for 100 to 400 participants (all times in ms).

No. of Students	Phase A	Phase B	Phase C	Total Exec. Time
100	365	737	1174	2276
200	401	810	1521	2732
300	693	1401	2741	4835
400	800	1881	2926	5607

6. Conclusions

Although various anonymous online evaluation protocols exist in the literature, they are not always easy to implement in real university environments due to practical limitations that may exist. The proposed anonymous evaluation protocol aims to provide a practical solution that can be implemented in most university environments, without sacrificing important anonymity and security properties and under realistic assumptions such as a local broadcast anonymous server. The proposed protocol provides strong assurances for evaluator eligibility and unforgeability, while at the same time protects evaluator anonymity against honest-but-curious adversaries. We also used ProVerif, a cryptographic protocol analyzer, to show core security aspects that hold in our protocol such as secrecy and authentication. Our web-based proof of concept implementation and our simulation results show that the proposed protocol can be practically implemented for several hundreds of evaluators.

In future work, we intend to extend the cryptographic verification to all aspects of security assurances of the protocol, not only with Proverif, but also using additional formal methods analyzers such as scyther [30] and cpsa [31].In our future work we also intend to extend our protocol validation by integrating in our case study secure hardware storage for the students' credentials, in order to analyse and elaborate on trust and cost management issues.

Author Contributions: Conceptualization, N.P. and S.M.; methodology, P.K. and E.M.; software, S.M.; validation, P.K. and E.M.; formal analysis, N.P. and P.K.; investigation, E.M. and N.P.; resources, N.P., S.M. and E.M.; data curation, E.M. and P.K.; writing–original draft preparation, all; writing–review and editing, N.P., E.M. and P.K.; supervision, P.K. and E.M.; project administration, P.K.; funding acquisition, P.K. All authors have read and agreed to the published version of the manuscript.

Acknowledgments: This work has been partially supported by the University of Piraeus Research Center (UPRC).

Conflicts of Interest: The authors declare no conflict of interest.

Appendix A. Proverif Architecture

ProVerif supports cryptographic primitives defined by rewrite rules and by equations that satisfy the finite variant property by [32]. The architecture of the tool is depicted in Figure A1. Proverif takes as input a model of the security protocol that needs to be checked, written in a pi-calculus extension. The user has encompassed the security properties she wants to prove in the model. Proverif uses a resolution algorithm with free selection to determine whether a fact is derivable from the clauses [26]. If the fact fails (not derivable) then the desired security property is proved. If the fact is derivable then an attack has been found. It also may reconstruct attacks when the security property do not hold.

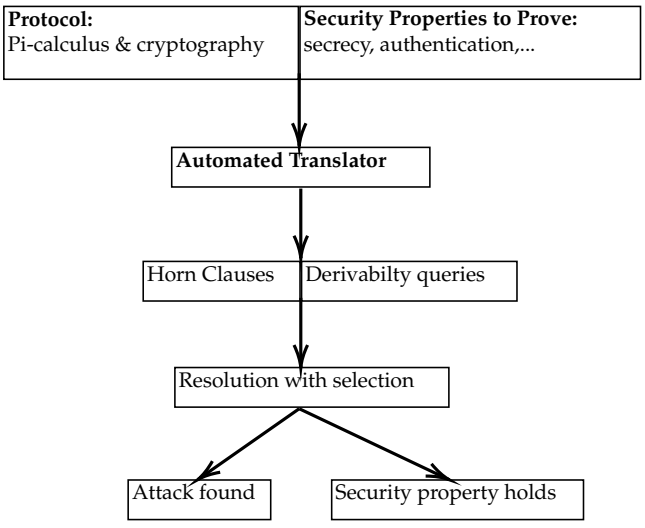


Figure A1. ProVerif Architecture.

Appendix B. Basic Grammar of Proverif

The basic grammar of the language is depicted in Table A1. Terms of the form M,N consist of names a, b, c, ..., variables x, y, z, tuples (M_1, ..., M_j), where j is the arity of the tuple, and constructor/destructor application defined h(M_1, ..., M_k), where k is the arity of h and arguments M_1, ..., M_k have appropriate types. P and Q represent processes. The null process does nothing, P | Q is the parallel composition of P and Q meaning that participants of the protocol running in parallel.

Table A1. Term and process grammar.

Term and Process Grammar	
M, N ::=	terms
a, b, c, k, m, n, ...	names
x, y, z	variables
(M_1, . . . , M_k)	tuple
h(M_1, . . . , M_k)	constructor/destructor application
M = N	term equality
M <> N	term inequality
M && M	conjunction
M | | M	disjunction
not(M)	negation
P, Q ::=	processes
0	null process
P | Q	parallel composition
!P	replication
new n : t; P	name restriction
in(M, x : t); P	message input
out(M, N); P	message output
if M then P else Q	conditional
let x = M in P else Q	term evaluation
R(M1, . . . , Mk)	macro usage

References

1. Li, C.; Lalani, F. The COVID-19 pandemic has changed education forever. Available online: https://www.weforum.org/agenda/2020/04/coronavirus-education-global-covid19-online-digital-learning/ (accessed on 16 June 2020)
2. Kluczniak, K.; Hanzlik, L.; Kubiak, P.; Kutyłowski, M. Anonymous evaluation system. In *International Conference on Network and System Security*; Springer: Berlin/Heidelberg, Germany, 2015; pp. 283–299.
3. Baldimtsi, F.; Lysyanskaya, A. Anonymous credentials light. In Proceedings of the 2013 ACM SIGSAC Conference on Computer & Communications Security, Berlin, Germany, 4–8 November 2013; pp. 1087–1098.
4. Camenisch, J.; Lysyanskaya, A. A signature scheme with efficient protocols. In *International Conference on Security in Communication Networks*; Springer: Berlin/Heidelberg, Germany, 2002; pp. 268–289.
5. Camenisch, J.; Lysyanskaya, A.; Meyerovich, M. Endorsed e-cash. In Proceedings of the 2007 IEEE Symposium on Security and Privacy (SP'07), Berkeley, CA, USA, 20–23 May 2007.
6. Camenisch, J.; Dubovitskaya, M.; Lehmann, A.; Neven, G.; Paquin, C.; Preiss, F.S. Concepts and languages for privacy-preserving attribute-based authentication. In *IFIP Working Conference on Policies and Research in Identity Management*; Springer: Berlin/Heidelberg, Germany, 2013; pp. 34–52.

7. Bender, J.; Dagdelen, Ö.; Fischlin, M.; Kügler, D. Domain-specific pseudonymous signatures for the german identity card. In *International Conference on Information Security*; Springer: Berlin/Heidelberg, Germany, 2012; pp. 104–119.
8. Cranor, L.F.; Cytron, R.K. Sensus: A security-conscious electronic polling system for the internet. In Proceedings of the Thirtieth Hawaii International Conference on System Sciences, Wailea, HI, USA, 7–10 January 1997.
9. Fujioka, A.; Okamoto, T.; Ohta, K. A practical secret voting scheme for large scale elections. In *International Workshop on the Theory and Application of Cryptographic Techniques*; Springer: Berlin/Heidelberg, Germany, 1992; pp. 244–251.
10. Camenisch, J.; Van Herreweghen, E. Design and implementation of the idemix anonymous credential system. In *Proceedings of the 9th ACM Conference on Computer and Communications Security*; ACM: New York, NY, USA, 2002; pp. 21–30.
11. Camenisch, J. Specification of the identity mixer cryptographic library. *Tech. Rep.* **2010**, *76*, 5–6.
12. Paquin, C.; Zaverucha, G. *U-prove Cryptographic Specification v1.1*; Technical Report; Microsoft Corporation: Redmond, WA, USA, 2011.
13. Benenson, Z.; Girard, A.; Krontiris, I.; Liagkou, V.; Rannenberg, K.; Stamatiou, Y. User acceptance of privacy-abcs: An exploratory study. In *International Conference on Human Aspects of Information Security, Privacy, and Trust*; Springer: Berlin/Heidelberg, Germany, 2014; pp. 375–386.
14. Sabouri, A.; Rannenberg, K. Abc4trust: Protecting privacy in identity management by bringing privacy-abcs into real-life. In *IFIP International Summer School on Privacy and Identity Management*; Springer: Berlin/Heidelberg, Germany, 2014; pp. 3–16.
15. Rannenberg, K.; Camenisch, J.; Sabouri, A. Attribute-based credentials for trust. In *Identity in the Information Society*; Springer: Berlin/Heidelberg, Germany, 2015.
16. Krontiris, I.; Benenson, Z.; Girard, A.; Sabouri, A.; Rannenberg, K.; Schoo, P. Privacy-ABCs as a Case for Studying the Adoption of PETs by Users and Service Providers. In *Annual Privacy Forum*; Springer: Berlin/Heidelberg, Germany, 2015; pp. 104–123.
17. Liagkou, V.; Stylios, C.; Petunin, A. Handling Privacy and Concurrency in an Online Educational Evaluation System. *Balt. J. Mod. Comput.* **2019**, *7*, 86–98. [CrossRef]
18. Stamatiou, Y.; Benenson, Z.; Girard, A.; Krontiris, I.; Liagkou, V.; Pyrgelis, A.; Tesfay, W. Course evaluation in higher education: The patras pilot of ABC4Trust. In *Attribute-Based Credentials for Trust*; Springer: Cham, Swizerland, 2015; pp. 197–239.
19. Schanzenbach, M.; Kilian, T.; Schütte, J.; Banse, C. Zklaims: Privacy-preserving attribute-based credentials using non-interactive zero-knowledge techniques. *arXiv* **2019**, arXiv:1907.09579.
20. Mostowski, W.; Vullers, P. Efficient u-prove implementation for anonymous credentials on smart cards. In *International Conference on Security and Privacy in Communication Systems*; Springer: Berlin/Heidelberg, Germany, 2011; pp. 243–260.
21. Sene, I.; Ciss, A.A.; Niang, O. I2pa: An efficient abc for iot. *Cryptography* **2019**, *3*, 16. [CrossRef]
22. Chaum, D.; Fiat, A.; Naor, M. Untraceable electronic cash. In *Conference on the Theory and Application of Cryptography*; Springer: Berlin/Heidelberg, Germany, 1988; pp. 319–327.
23. Lamport, L. Password authentication with insecure communication. *Commun. ACM* **1981**, *24*, 770–772. [CrossRef]
24. Kotzanikolaou, P.; Chatzisofroniou, G.; Burmester, M. Broadcast anonymous routing (BAR): Scalable real-time anonymous communication. *Int. J. Inf. Secur.* **2017**, *16*, 313–326. [CrossRef]
25. Dierks, T.; Rescorla, E. The Transport Layer Security (TLS) Protocol Version 1.2. 2008. Available online: www.hjp.at/doc/rfc/rfc5246.html (accessed on 20 July 2020)
26. Blanchet, B. Modeling and Verifying Security Protocols with the Applied pi Calculus and Proverif. 2016. Available online: https://hal.inria.fr/hal-01423760/ (accessed on 22 April 2020)
27. Das, P.; Faust, S.; Loss, J. A formal treatment of deterministic wallets. In *Proceedings of the 2019 ACM SIGSAC Conference on Computer and Communications Security*; ACM: New York, NY, USA, 2019; pp. 651–668.
28. Dolev, D.; Yao, A. On the security of public key protocols. *IEEE Trans. Inf. Theory* **1983**, *29*, 198–208. [CrossRef]
29. GitHub Development Platform. Available online: https://github.com/sophron/BAR (accessed on 27 February 2020)

30. Cremers, C.J. The Scyther Tool: Verification, falsification, and analysis of security protocols. In *International Conference on Computer Aided Verification*; Springer: Berlin/Heidelberg, Germany, 2008; pp. 414–418.
31. Doghmi, S.F.; Guttman, J.D.; Thayer, F.J. Searching for shapes in cryptographic protocols. In *International Conference on Tools and Algorithms for the Construction and Analysis of Systems*; Springer: Berlin/Heidelberg, Germany, 2007; pp. 523–537.
32. Comon-Lundh, H.; Delaune, S. The finite variant property: How to get rid of some algebraic properties. In *International Conference on Rewriting Techniques and Applications*; Springer: Berlin/Heidelberg, Germany, 2005; pp. 294–307.