Trustworthy Exams Without Trusted Parties

Giampaolo Bella[a], Rosario Giustolisi[b], Gabriele Lenzini[c], Peter Y. A. Ryan[c]

[a]*Dipartimento di Matematica e Informatica, Università di Catania, Italy*
[b]*Swedish Institute of Computer Science, Stockholm, Sweden*
[c]*SnT, University of Luxembourg*

Abstract

Historically, exam security has mainly focused on threats ascribed to candidate cheating. Such threats have been normally mitigated by invigilation and anti-plagiarism methods. However, as recent exam scandals confirm, also invigilators and authorities may pose security threats. The introduction of computers into the different phases of an exam, such as candidate registration, brings new security issues that should be addressed with the care normally devoted to security protocols.

This paper proposes a protocol that meets a wide set of security requirements and resists threats that may originate from candidates as well as from exam administrators. By relying on a combination of oblivious transfer and visual cryptography schemes, the protocol does not need to rely on any trusted third party. We analyse the protocol formally in ProVerif and prove that it verifies all the stated security requirements.

Contents

1. Introduction

Exams play a key role in assessing skills and knowledge of people, hence they are indispensable to bring meritocracy in modern societies. The idea of using exams as a procedure to eliminate favouritism and corruption for personnel selection came to West in the 17$^{\text{th}}$ century [1]. Nowadays, exams are extensively used in various situations, for example in universities for admission and progression of students to a degree, in non-profit organizations to certify professional qualifications, and in the work sector for recruitment and promotion.

An exam is a complex procedure that begins with the appointment of examiners and concludes with the notification of the results. Ideally, every step of such a procedure should resist any misbehaviour so that people could trust the exam system, and society could truly achieve meritocracy. However, the participating principals may be corrupted, and so they may commit misconducts that are hard to eradicate. Recent scandals confirm that administrators, as well as candidates, may misbehave [2, 3, 4], and the introduction of computers in the exam procedures has not reduced the trend [5, 6], making the exam not necessarily more secure than it was.

Such scandals make people doubtful about exam procedures. We observe that current exams generally fail to recognise that every principal may profitably misbehave, as they are normally designed to only resist cheating candidates. In this paper, we advance a novel exam protocol that also considers misbehaving administrators and, therefore, we study the security of the protocol without assuming any trusted parties. The underlying idea of our protocol is to combine oblivious transfer and visual cryptography to allow candidate and administrator to jointly generate a pseudonym that anonymises the candidate's test. The pseudonym is revealed only to the candidate at the beginning of testing. We analyse the protocol formally in ProVerif [7] and prove that it meets a wide set of security requirements, even with the presence of corrupted administrators. We believe that by removing this additional layer of trust in the design and analysis of exams, we foster their trustworthiness and contribute to people's confidence in the exam practice in general.

We extend a conference paper of ours [8] by extending and updating the security requirements, by updating the formal definitions, and by redesigning our protocol to meet the novel set of requirements. More in detail, this article provides the following novel contributions that exceed the conference paper:

- It extends the list of requirements with *Mark Authentication*, which is unpublished.
- It further extends the list of requirements with *Test Authenticity*, *Anonymous Examiner* and *Test Integrity Verifiability* using existing definitions [9].
- It updates the formal definition of requirement *Mark Authenticity* so that appropriate authentication requirements now cover all the phases of an exam.

- It provides a novel graphical representation of the authentication requirements with their coverage of all protocol phases.
- It updates the formal definition of requirement *Dispute Resolution* so that the scenario with both administrator and candidate being corrupted is admitted.
- It presents a novel set of 15 requirements derived from the 11 published ones [8] by the extensions and updates outlined above and by rewriting otherwise towards consistency of style and readability.
- It updates several parts of the exam protocol to meet the novel set of requirements.
- It updates the examiner role by partitioning it as administrator, examiner and invigilator roles so that both protocol design and analysis gain detail.
- It redesigns the Testing Dispute Resolution algorithm to meet the new definition of Dispute Resolution.
- It proves by ProVerif that the updated exam protocol meets the novel set of requirements.

Outline. The paper is organized as follows. Section 2 analyses some related work. Section 3 details the list of security requirements and their formalisation. Section 4 describes the protocol. Section 5 details the formal analysis of the protocol in ProVerif. Finally, Section 6 draws future work and conclusions.

2. Related work

Few works [10, 11] list a number of security requirements for exams, still only informally. Only Dreier *et al.* [12] define a formal framework in the applied π-calculus to define and analyse authentication and privacy requirements for exams, which this paper extends with new requirements. They analyse two existing electronic exam protocols as case studies. In the domain of Computer Supported Collaborative Working, Foley and Jacob [13] formalised confidentiality requirements and proposed an exam as a case study.

Recently, a number of secure exam protocol have been introduced. Castella-Roca *et al.* [14] develop an exam protocol that meets authentication and privacy properties in the presence of a fully trusted exam manager. Bella *et al.* [15] propose WATA IV, a protocol that considers a corrupted examiner but assumes an honest-but-curious anonymiser. Huszti-Pethő [16] advance an Internet-based exam with few trust requirements on principals, but with some security flaws in the design [12]. Giustolisi *et al.* [17] describe *Remark!*, another Internet-based exam protocol that ensures authentication and conditional anonymity requirements with minimal trust assumption. The protocol generates pseudonyms via an exponentiation mixnet, which assumes at least one honest mix server.

Many universities and companies for language proficiency tests or personnel selection replaced traditional exams with Internet-based exams [18, 19] or Computer-assisted exams [20, 21]. For example, the European Union adopts computer-based exams for the selection of EU personnel [22]. However, their designs include trusted exam authorities that are in charge of the critical tasks of the exam. Conversely, our protocol is designed to minimise the reliance on trusted parties.

Maffei *et al.* [23] implemented a course evaluation system that guarantees privacy using anonymous credential schemes without a trusted third party. Similarly, Hohenberger *et al.* [24] advanced *ANONIZE*, a protocol for surveys that ensures authentication and privacy in presence of corrupted authorities. However, surveys have different security requirements than exams, for instance, surveys do not consider test authorship and fixed-term anonymity definitions.

Some related protocols have been proposed in the area of conference management systems. Kanav *et al.* [25] introduced *CoCon*, a formally verified implementation of conference management system that guarantees confidentiality. Arapinis *et al.* [26] introduced and formally analysed *ConfiChair*, a cryptographic protocol that addresses secrecy and privacy risks coming from a *malicious-but-cautious* cloud. Their work has been recently extended to support any cloud-based system that assumes honest managers, such as public tender management and recruitment process [27].

Few protocols [28, 29] have been proposed to ensure *anonymous marking* still assuming trusted authorities. The goal of our protocol is to meet anonymous marking and other properties without the need of a TTP.

3. Security requirements

Our exam protocol involves four roles. These are obtained by partitioning the examiner role [8] as administrator, examiner and invigilator to increase the detail.

- The *candidate (C)* role, who takes the exam.
- The *administrator (A)* role , who checks the eligibility of candidates who wish to take an exam, populates a list of registered candidates accordingly, informs the candidates of the marks that their respective tests received, and stores this information with some recorders.
- The *invigilator (K)* role, who distributes tests to candidates, checks candidates' identities, follows candidates while they take their test preventing them from misbehaving, collects the tests from the candidates and then distributes the test answers to the examiners.
- The *examiner (E)* role, who reads the test answers and produces adequate marks for them.

A typical exam runs in phases: at *preparation*, the administrator files a new exam, registers eligible candidates, generates the tests and all the relevant material for the subsequent phase. For example, creation of questions, printing of tests, and generation of pseudonyms to anonymise tests are tasks accomplished in this phase; at *testing* each registered candidate gets a test. Invigilators watch candidates through this phase. Each candidate answers her test and may have to complete it with her personal details. The candidate then submits her test answer to the invigilator; at *marking* the test answers of all candidates reach the examiner for evaluation. The examiner reads the test answers and evaluates their adherence to the required knowledge, then forming a mark, chosen on a given scale, for each test; at *notification*, the administrator gives each candidate the mark for the test answer the candidate submitted and stores the mark.

We consider a list of seven authentication, five privacy, two verifiability, and one accountability requirements. This list updates (and extends) a previous one as described above (§1). Although we find them highly desirable out of personal experience and discussions with colleagues, the list of requirements is not meant to be universal or exhaustive. It means that certain exam protocols might demand additional requirements. However, we consider our set of requirements to be fundamental as similar requirements can be found in other independent works [10, 11], still only informally. As we shall see later, the list of our requirements includes the classic confidentiality, integrity, and authentication requirements and many others.

To express our requirements unambiguously, we use the applied π-calculus [30], a formal language for the description and analysis of security protocols, in which principals are represented as processes. The processes communicate via a public channel that is controlled by an attacker. We chose this approach rather than others (e.g., ISO security standard, informal argumentation) because the applied π-calculus is the input language of ProVerif, an automatic protocol verifier tool. Authentication can be defined using *correspondence assertions* [31]. An event $\mathtt{e}$ is a message emitted into a special channel that is not under the control of the attacker. To model correspondence assertions, we annotate processes with events such as $\mathtt{e}\langle M_1, ...M_n\rangle$ and reason about the relationships ($\rightsquigarrow$) between events and their arguments in the form *"if an event* $\mathtt{e}\langle M_1, ...M_n\rangle$ *has been executed, then an event* $\mathtt{e}'\langle N_1, ...N_n\rangle$ *has been previously executed"*. Verifiability requirements demand the existence of sound and complete *verifiability-tests* [32]. Soundness and completeness can be checked in the applied π-calculus as reachability properties. The same approach can be used to express accountability. The applied π-calculus supports the notion of *observation equivalence*. Informally, two processes are observational equivalent if an observer cannot distinguish the processes even if they handle different data or perform different computations. The indistinguishability characterization of the definition of observation equivalence allows us to capture privacy requirements.

3.1. Authentication

In the applied π-calculus, an event is a message output $e(\vec{a})$: e is the event channel and $\vec{a}$ is a, possibly empty, list of arguments. The message appears in the trace as soon as the execution of the process reaches the event.

To model authentication requirements as correspondence assertions, it is necessary to define a number of relevant events. Events normally need to agree on some arguments to capture authentication. Thus, we introduce the terms that serve as arguments in our events.

- id_c refers to the identity of the candidate;
- $ques$ denotes the question(s) of the test;
- ans denotes the answer of a test;
- $mark$ denotes the mark assigned to the test;
- id_e refers to the identity of the examiner;
- id_test refers to the identifier of the test.

We then define a list of eight events that allow us to specify seven fundamental authentication requirements for exams:

- `registered`$\langle id\_c \rangle$ means that the administrator considers the candidate id_c registered for the exam. The event is inserted into the process of the administrator at the location where the registration of the candidate id_c concludes.
- `submitted`$\langle id\_c, ques, ans, id\_test \rangle$ means that the candidate id_c considers the test id_test, which consists of question $ques$ and answer ans, submitted for the exam. The event is inserted into the process of the candidate at the location where the test is sent to the invigilator.
- `collected`$\langle id\_c, ques, ans, id\_test \rangle$ means that the invigilator accepts the test id_test, which originates from the candidate id_c. The event is inserted into the process of the invigilator at the location where the test is considered as accepted.
- `distributed`$\langle id\_c, ques, ans, id\_test, id\_e \rangle$ means that the invigilator considers the test id_test, which originates from the candidate id_c, associated with the examiner id_e for marking. The event is inserted into the process of the invigilator at the location where the test is distributed to the examiner.
- `marked`$\langle ques, ans, mark, id\_test, id\_e \rangle$ means that the examiner id_e considers the test id_test, which consists of question $ques$ and answer ans, evaluated with $mark$. The event is inserted into the process of the examiner at the location where the test is marked.
- `requested`$\langle id\_c, id\_test \rangle$ means that the candidate id_c accepts to learn the mark associated to the test id_test. The event is inserted into the process of the candidate at the location where the request is sent to the administrator.

- $\texttt{stored}\langle id\_c, id\_test, mark\rangle$ means that the administrator considers the candidate id_c associated with $mark$. The event is inserted into the process of the administrator at the location where it officially registers the mark assigned to the candidate.

- $\texttt{notified}\langle id\_c, id\_test, mark\rangle$: means that the candidate id_c officially accepts the mark $mark$. The event is inserted into the process of the candidate at the location where the mark is considered accepted.

These events mark important steps of an exam protocol, and some can be associated with the phases of an exam. The event `registered` normally concludes the preparation phase, while `collected` concludes the testing phase. The event `distributed` begins the marking phase, which the event `marked` concludes. Finally, the events `requested` and `notified` respectively opens and concludes the notification phase. Note that these events implicitly refer to the same exam session. However, one might want to parameterise all the events with a common term in order to distinguish among exam sessions.

The first authentication requirement we consider is *Candidate Authorisation*, which concerns preparation and testing. Informally, we want to capture the requirement that only registered candidates can take the exam. More specifically, the requirement says that if a candidate submits her test, then the candidate was correctly registered for the exam.

Requirement 1 (**Candidate Authorisation**). *An exam protocol ensures* Candidate Authorisation *if for every exam process EP*

$$\texttt{submitted}\langle id\_c, ques, ans, id\_test\rangle \rightsquigarrow inj\ \texttt{registered}\langle id\_c\rangle$$

on every execution trace.

This requirement is modelled as injective correspondence assertion because the exam should consider only one submission per registered candidate.

The second authentication requirement that we advance is *Answer Authenticity*, which concerns the testing phase. It states that the collector should consider only answers that candidates actually submitted, and that the contents of each collected test are not modified after submission. It says that a test must be bound to a candidate identity. A practical implication of this requirement is that two candidates will be unable to get tested on each other's questions, something they could attempt due to a variety of colluding aims. Moreover, only one test from each candidate should be considered, namely every time the collector process emits `collected`, there is a distinct earlier occurrence of the event `submitted` that satisfies the relationship between their arguments.

Requirement 2 (**Answer Authenticity**). *An exam protocol ensures* Answer Authenticity *if for every exam process EP*

$$\texttt{collected}\langle id\_c, ques, ans, id\_test\rangle \rightsquigarrow inj\ \texttt{submitted}\langle id\_c, ques, ans, id\_test\rangle$$

on every execution trace.

The third requirement is *Test Origin Authentication* and concerns preparation and testing. Informally, it says that the collector should accept only tests that originate from registered candidates. This requirement should be modelled as an injective agreement to enforce that only one test from each registered candidate is actually collected.

Requirement 3 (Test Origin Authentication). *An exam protocol ensures* Test Origin Authentication *if for every exam process EP*

$$\texttt{collected}\langle id\_c, ques, ans, id\_test\rangle \rightsquigarrow inj\ \texttt{registered}\langle id\_c\rangle$$

on every execution trace.

Another authentication requirement is *Test Authenticity* and concerns testing and marking. It is stated below using a recent definition [9] that was not available in the conference version [8] of the present article. Because the tests are possibly assigned to a number of examiners to even the marking load, Test Authenticity insists that an examiner only marks the tests intended for him. Moreover, the contents of each test should not be modified until the tests are marked.

Requirement 4 (Test Authenticity). *An exam protocol ensures* Test Authenticity *if for every exam process EP*

$$\begin{aligned}&\texttt{marked}\langle ques, ans, mark, id\_test, id\_e\rangle \rightsquigarrow\\ &inj\ \texttt{collected}\langle id\_c, ques, ans, id\_test\rangle \cup\\ &inj\ \texttt{distributed}\langle id\_c, ques, ans, id\_test, id\_e\rangle\end{aligned}$$

on every execution trace.

Some universities allow candidates to take an exam up to a fixed number of times until they succeed. However, if the candidate withdraws from an exam before or during testing, this is not counted towards the number of attempts. Other universities have a policy that prevents the candidate to resit a failed exam at the very next session unless the fail derived from the candidate's decision to withdraw before being notified a mark. Thus, we consider the requirement of *Notification Request Authentication.* It says that a mark should be associated with the candidate only if she requests to learn her mark.

Requirement 5 (Notification Request Authentication). *An exam protocol ensures* Notification Request Authentication *if for every exam process EP*

$$\texttt{stored}\langle id\_c, id\_test, mark\rangle \rightsquigarrow inj\ \texttt{requested}\langle id\_c, id\_test\rangle$$

on every execution trace.

The requirement *Mark Authenticity* concerns marking and notification. It prescribes that a mark should be correctly recorded for the corresponding test and candidate, namely that the administrator should store the mark assigned to a test during marking by the examiner. This definition updates our previous one

[8] by involving a pair of events that are now closer together. This contributes to a finer and fuller coverage of all the phases of an exam in terms of authentication, as Figure 1 will confirm graphically.

Requirement 6 (Mark Authenticity). *An exam protocol ensures* Mark Authenticity *if for every exam process EP*

$$\texttt{stored}\langle id\_c, id\_test, mark\rangle \rightsquigarrow \texttt{marked}\langle ques, ans, mark, id\_test, id\_e\rangle$$

on every execution trace.

A final, important requirement is *Mark Authentication*, which has never been published before. It says that the candidate is notified with the same mark that has been stored by the administrator.

Requirement 7 (Mark Authentication). *An exam protocol ensures* Mark Authentication *if for every exam process EP*

$$\texttt{notified}\langle id\_c, id\_test, mark\rangle \rightsquigarrow \texttt{stored}\langle id\_c, id\_test, mark\rangle$$

on every execution trace.

In summary, an exam protocol that ensures all requirements outlined above preserves the association between candidate identity, mark, and test, including question and answer, through all the phases of the exam. The relationships between authentication requirements with respect to exam run and principals can be graphically represented as in Figure 1. This novel representation shows that the stated requirements rest on an ordered sequence of events. It can be noted that there is no requirement relating directly the events `collected`, `distributed`, `marked` and `requested` (i.e., the densely dotted arrows). We chose not to specify the requirement *"the collector distributes the accepted tests"* since such requirement is usually enforced by the sequential execution of the examiner process as both events belong to the same process. Moreover, it always holds if the common arguments of the two events are derived from the same source, for example, if the common arguments are built from the same message input. Likewise, we did not specify the requirement *"the candidate requests the mark of their test"* because, in general, the events `marked` and `requested` may be emitted in any order. In fact, a candidate may request the mark to the administrator even before the examiner marks the test, but this is not a security issue.

Figure 1 also emphasises that the combination of these requirements produce novel requirements. If an exam protocol guarantees Candidate Authorisation and Answer Authenticity, then the protocol also guarantees Test Origin Authentication, namely the tests submitted by registered candidates are actually collected. Conversely, a protocol that guarantees Test Origin Authentication may guarantee neither Candidate Authorisation nor Answer Authenticity. Test Origin Authentication only states that collected tests originate from registered candidates without considering the actually submitted tests. It follows that a test modified after submission may meet Test Origin Authentication but not Answer Authenticity.

In general, if we consider a requirement in a certain phase of the exam, we cannot infer anything about other phases. For example, Mark Authenticity signifies that the administrator stores the same mark that the examiner assigned to the candidate's test. However, the test provided by the invigilator to the examiner may contain a different answer with respect to the answer the candidate submitted at testing. Only if the exam protocol also guarantees Answer Authenticity and Test Authenticity, are the contents of the tests identical through the whole exam.

3.2. Privacy

To model privacy requirements as equivalence properties, we use the definition of labelled bisimilarity ($\approx_l$) as defined by Abadi and Fournet in [30]. We also use the definition of exam process and the corresponding notations introduced by Dreier et al. [12] to specify the requirements.

We denote with EP a closed process such that:

$$EP = (C\sigma_{id_1}\sigma_{a_1}| \dots | \; C\sigma_{id_j}\sigma_{a_j}| \; E\sigma_{id'_1}\sigma_{m_1}| \dots \; | \; E\sigma_{id'_k}\sigma_{m_k}| \; A\sigma_q| \; K\sigma_{test})$$

where:

- $C\sigma_{id_i}\sigma_{a_i}$'s are the processes run by the candidates. The substitutions σ_{id_i} and σ_{a_i} specify the identity and the answers associated with the i^{th} candidate;
- $E\sigma_{id'_i}\sigma_{m_i}$'s are the processes run by examiners. The substitution $\sigma_{id'_i}$ and σ_{m_i} specify the identity and the mark associated with the i^{th} examiner;
- $K\sigma_{test}$ is the process run by the collector. The substitution σ_{test} associates a test with an examiner for marking;
- $A\sigma_q$'s is the process run by the administrator. The substitution σ_q specifies the exam questions;

We denote with "$EP_I[\_]$" the context of the exam process EP pruned of identities that appear in the set I. For example, the process

$$\nu\tilde{n}.(\_|\_|C\sigma_{id_3}\sigma_{a_3}| \dots |C\sigma_{id_j}\sigma_{a_j}|E\sigma_{id'_1}\sigma_{m_1}| \dots |E\sigma_{id'_k}\sigma_{m_k}|K\sigma_{dist}|A\sigma_q)$$

can be concisely written as $EP_{\{id_1,id_2\}}[\_]$. Such compact notation is useful to specify and focus exactly on the processes concerned by the requirement. For example, we can write $EP_{\{id_1,id_2\}}[C\sigma_{id_1}\sigma_{a_1}|C\sigma_{id_2}\sigma_{a_2}]$ to reason about candidates id_1 and id_2 without repeating the entire exam instance.

Second, we denote with "$EP|_{\mathtt{e}}$" the process EP pruned of the code that follows the event e. For example, the process $EP|_{\mathtt{marked}}$ considers an exam instance that terminates at marking, precisely after the event `marked` is emitted. This notation is useful to capture fixed-term requirements such as anonymous marking, which is intended to hold until after the marking, but is eventually falsified at notification when the mark is assigned to the candidate.

The first privacy requirement we consider is *Question Indistinguishability*, which says that the questions are not revealed until the testing phase begins. Thus, it is a fixed-term requirement ending with the preparation phase.

Requirement 8 (**Question Indistinguishability**). *An exam protocol ensures* Question Indistinguishability *if for every exam process EP and every questions q_1 and q_2*

$$EP[A\sigma_{q_1}]|_{\texttt{registered}} \approx_l EP[A\sigma_{q_2}]|_{\texttt{registered}}$$

Question Indistinguishability states that two processes with different questions have to be observationally equivalent until after the preparation phase. Note that this requirement is more stringent than reachability-based secrecy because the attacker should not be able to distinguish whether the exam will use q_1 or q_2 although he knows both the questions in advance. For example, the attacker cannot say whether the questions of the current exam are similar to the questions of the previous exam, which are already in the attacker's knowledge. The analysis of Question Indistinguishability is particularly interesting when considering corrupted candidates who may want to know the questions in advance. For example, in the particular case of a corrupted candidate id_1, the requirement gets rewritten as

$$\begin{aligned} &EP_{\{id_1\}}[(C\sigma_{id_1}\sigma_{a_1})^{c_1,c_2}|A\sigma_{q_1}]|_{\texttt{registered}} \approx_l \\ &EP_{\{id_1\}}[(C\sigma_{id_1}\sigma_{a_1})^{c_1,c_2}|A\sigma_{q_2}]|_{\texttt{registered}} \end{aligned}$$

The next requirement is *Anonymous Marking*, which covers preparation, testing, and marking. This requirement signifies that the examiner marks a test while ignoring its author, namely an anonymous test. It is a clear contribution to the fairness of the marking. As it stands below, the requirement insists on the anonymity of a test only until the point that the examiner affixes a mark on the test. Anonymous Marking can be specified as two exam instances in which the processes of two candidates who swap their answers cannot be distinguished until after the end of the marking phase.

Requirement 9 (**Anonymous Marking**). *An exam protocol ensures* Anonymous Marking *if for every exam process EP, every two candidates id_1 and id_2, and every two answers a_1 and a_2*

$$EP_{\{id_1,id_2\}}[C\sigma_{id_1}\sigma_{a_1}|C\sigma_{id_2}\sigma_{a_2}]|_{\texttt{marked}} \approx_l EP_{\{id_1,id_2\}}[C\sigma_{id_1}\sigma_{a_2}|C\sigma_{id_2}\sigma_{a_1}]|_{\texttt{marked}}$$

In other words, Anonymous Marking says that the process where candidate id_1 submits a_1 and candidate id_2 submits a_2 is indistinguishable to the process where candidate id_1 submits a_2 and candidate id_2 submits a_1. It prevents the attacker to obtain the identity of the candidate who submits a certain answer before the marking ends.

Anonymous Marking means that nobody knows who submitted a test while this is being marked, except the official author of the test. Similarly to Question

Indistinguishability, it is interesting to consider corrupted principals also in the analysis of this requirement, so that test anonymity during marking will even resist collusion of the examiner with other authorities and candidates. Also with this requirement, the definition of *corrupted process* [33] can model corrupted examiners and authorities.

Of course, the definition could also be extended to corrupted candidates similarly. However, meeting the requirement would then impose stating a limitation: the candidates id_1 and id_2 who submit two different answers, to be honest. This limitation would avoid the corner case in which all candidates but one reveal their answers to the attacker, who could then easily associate the remaining answer with the honest candidate and thus trivially violate the requirement.

We now consider the requirement *Anonymous Examiner*. It is stated below using a recent definition [9] that was not available in the conference version [8] of the present article. It concerns all the phases of an exam because examiner anonymity could be required to hold forever to prevent bribing or coercion. Thus, the requirement of Anonymous Examiner says that no candidate knows which examiner marked her test.

Requirement 10 (Anonymous Examiner). *An exam protocol ensures* Anonymous Examiner *if for every exam process EP, every two candidates id_1 and id_2, every two examiners id'_1 and id'_2, every two marks m_1 and m_2, and two associations $test_1$ and $test_2$*

$$EP_{\{id_1,id_2,id'_1,id'_2\}}\ [C\sigma_{id_1}\sigma_{a_1}|C\sigma_{id_2}\sigma_{a_2}|E\sigma_{id'_1}\sigma_{m_1}|E\sigma_{id'_2}\sigma_{m_2}|K\sigma_{test_1}]\ \approx_l$$
$$EP_{\{id_1,id_2,id'_1,id'_2\}}[C\sigma_{id_1}\sigma_{a_1}|C\sigma_{id_2}\sigma_{a_2}|E\sigma_{id'_1}\sigma_{m_2}|E\sigma_{id'_2}\sigma_{m_1}|K\sigma_{test_2}]$$

where

- σ_{test_1} *associates the test of candidate id_1 to examiner id'_1 and the test of candidate id_2 to examiner id'_2;*
- σ_{test_2} *associates the test of candidate id_1 to examiner id'_2 and the test of candidate id_2 to examiner id'_1.*

Thus, Anonymous Examiner states that a process in which the examiner id'_1 evaluates the test of candidate id_1 while the examiner id'_2 evaluates the test of candidate id_2 is indistinguishable to the process in which the examiner id'_1 evaluates the test of candidate id_2 while the examiner id'_2 evaluates the test of candidate id_1. Note that the two marks σ_{m_1} and σ_{m_2} are swapped on the examiner processes to ensure that each test is evaluated with the same mark in both cases. In the field of peer review systems, this requirement is known as *blind review.* The requirement of *double-blind review* instead refers to a peer review system that ensures both Anonymous Examiner and Anonymous Marking, namely anonymity is provided to authors and examiners. To ensure a stronger version of Anonymous Marking it is possible to model corrupted administrator and candidates, provided that examiners id'_1 and id'_2 are honest.

This would avoid the corner case in which an examiner reveals the mark to the attacker, a case that would trivially violate the requirement.

The requirement of *Mark Privacy* concerns all phases of an exam. It states that the mark ultimately attributed to a candidate is treated as valuable personal information of the candidate's. More specifically, no one learns the mark, besides the examiner, the concerned candidate, and the authority responsible for the notification. This means that the marks cannot be public.

Requirement 11 (**Mark Privacy**). *An exam protocol ensures* Mark Privacy *if for every exam process EP and every two marks m_1 and m_2*

$$EP_{\{id'\}}[E\sigma_{id'}\sigma_{m_1}] \approx_l EP_{\{id'\}}[E\ \sigma_{id'}\sigma_{m_2}].$$

The definition of Mark Privacy means that a process in which the examiner id' assigns the mark m_1 to an answer cannot be distinguished from a process in which the same examiner assigns a different mark m_2 to the same answer. If an exam protocol guarantees Mark Privacy, then the administrator cannot publicly disclose the marks even if these cannot be associated with the corresponding candidates. In fact, the publication of the marks allows the attacker to distinguish the processes.

Also with this requirement some candidates and examiners can be assumed to be corrupted, namely to collaborate with the attacker to find out the marks of other candidates. However, the examiner who assigns the different marks, the two candidates who submit the tests, and the administrator should be honest. Otherwise, any of these could violate the requirement by revealing the mark to the attacker.

Since Mark Privacy may be a too strong definition for certain exams, we introduce a variant called *Mark Anonymity.* This requirement states that no one learns the association between a mark and the corresponding candidate. Intuitively, an exam protocol that publishes the list of all marks might still ensure Mark Anonymity, but not Mark Privacy. This is a common privacy requirement for real-world applications such as public competitions, in which marks are published and associated with a list of pseudonyms for transparency.

Requirement 12 (**Mark Anonymity**). *An exam protocol ensures* Mark Anonymity *if for every exam process EP, every two candidates id_1, id_2, every examiner id', every two answers a_1, a_2, two substitutions σ_{m_a} and σ_{m_b} and an association test*

$$EP_{\{id_1,id_2,id'\}}[C\sigma_{id_1}\sigma_{a_1}|C\sigma_{id_2}\sigma_{a_2}|E\sigma_{id'}\sigma_{m_a}|K\sigma_{test}] \approx_l$$
$$EP_{\{id_1,id_2,id'\}}[C\sigma_{id_1}\sigma_{a_1}|C\sigma_{id_2}\sigma_{a_2}|E\sigma_{id'}\sigma_{m_b}|K\sigma_{test}]$$

where

– *σ_{test} associates the tests of both candidates id_1 and id_2 to the examiner id';*

– *σ_{m_a} attributes the mark m_1 to the answer a_1 and the mark m_2 to the answer a_2;*

– *σ_{m_b} attributes the mark m_2 to the answer a_1 and the mark m_1 to the answer a_2.*

In other words, a process in which an examiner evaluates two answers a_1 and a_2 respectively with m_1 and m_2 is indistinguishable for the attacker to a process in which the examiner evaluates the same answers but with swapped marks, namely the examiner marks a_1 and a_2 respectively with m_2 and m_1. In doing so, the authority can make the list of marks public assuming the attacker cannot associate the marks to the candidates. The analysis of Mark Anonymity requires the two concerned candidates, the examiner, and the administrator to be honest. Otherwise, they can simply reveal the answer and the associated mark to allow the attacker to distinguish the two case processes. Other principals can be considered corrupted. It can be noted that an exam protocol that guarantees *Mark Privacy* also guarantees *Mark Anonymity.* The σ_{m_a} and σ_{m_b} defined in Mark Anonymity are in fact special instances of the σ_{m_1} and σ_{m_2} defined in Mark Privacy.

3.3. Verifiability

We consider the verifiability requirements *Mark Integrity Verifiability* and *Test Integrity Verifiability*, as advanced by Dreier et al. in [9]. In general, a protocol is verifiable with respect to a specific property if a *verifiability-test* exists, namely an algorithm that decides the property, and the algorithm is sound and complete. This particular requirement means that the candidate can verify that she was notified with the very mark that had been assigned to her test.

In the following, we will advance a verifiability-test, conveniently named `testMIV`, that outputs *true* if the candidate was notified with the mark of her test, or *false* otherwise. In the applied π-calculus, `testMIV` can be specified as a process that emits the event $\texttt{OK}(id\_c, pid, mark)$ when it is supposed to output *true* and `KO` otherwise. The event $published(pid)$ is emitted by the bulletin board when a test identified with pid is available; the event $assigned(id\_c, pid, mark)$ is emitted by the candidate at the end of notification. We say that `testMIV` is sound if the event $\texttt{OK}(id\_c, pid, mark)$ is preceded by the two specific events $assigned(id\_c, pid, mark)$ and $published(pid)$ in every execution trace of the protocol; we say that `testMIV` is complete if the event `KO` is emitted in no execution trace of the protocol when the test is fed with correct data.

Requirement 13 (**Mark Integrity Verifiability**). *An exam protocol ensures Mark Integrity Verifiability if there exists a verifiability-test for Mark Integrity, namely a sound and complete algorithm that decides Mark Integrity.*

Another requirement in this group is Test Integrity Verifiability, which states that the candidate can check that her test is accepted and marked precisely in the same form as she submitted it. It is stated below using a recent definition [9] that was not available in the conference version [8] of the present article. Similarly to the formalisation of the previous requirement, Test Integrity Verifiability is met through the existence of an appropriate verifiability-test, namely an algorithm `testTIV` that outputs *true* if the candidate's test was marked as it had been submitted, or *false* otherwise.

In the applied π-calculus, `testTIV` can be specified as a process that emits the event $\texttt{OK}(\mathit{id\_c}, \mathit{pid}, \mathit{answer})$ when it is supposed to output *true* and `KO` when it supposed to output *false*. The event $\mathit{published}(\mathit{pid})$ is emitted by the administrator when a test identified with pid is available; the event $\mathit{accepted}(\mathit{id\_c}, \mathit{pid}, \mathit{mark})$ is emitted by the candidate at the end of marking. We say that `testTIV` is sound if the event $\texttt{OK}(\mathit{id\_c}, \mathit{pid}, \mathit{answer})$ is preceded by the two specific events $\mathit{accepted}(\mathit{id\_c}, \mathit{pid}, \mathit{answer})$ and $\mathit{published}(\mathit{pid})$ in every execution trace of the protocol; we say that `testTIV` is complete if the event `KO` is emitted in no execution trace of the protocol when the test is fed with correct data.

Requirement 14 (**Test Integrity Verifiability**)**.** *An exam protocol ensures Test Integrity Verifiability if there exists a verifiability-test for Test Integrity, namely a sound and complete algorithm that decides Test Integrity.*

3.4. Accountability

Finally, an accountability requirement must be introduced: *Testing Dispute Resolution*. In general, accountability allows us to identify which principal is responsible for a protocol failure. With exams, a candidate should be able to submit a test and receive the corresponding mark. Should any of these fail, Dispute Resolution prescribes that the participant who caused such a failure can be identified.

Below, we formally model Testing Dispute Resolution in the applied π-calculus by appealing to a process `dispute` that performs dispute resolution. Its specification is much more detailed than what we published [8]. The definition relies on two caveats: first resort to unreachability of an event to prove soundness; second, as it may happen that *both* candidate and administrator are corrupted, consider also this case towards soundness. The `dispute` process emits the event `Cguilty` when the candidate is the culprit, the event `Aguilty` if the administrator is the culprit, or the event `CAguilty` if both are culprits. If the protocol executes the process `dispute`, then at least either the administrator or the candidate is corrupted. Thus, assuming that `dispute` returns a principal, the idea is to check that `dispute` cannot return an honest principal, if any, instead of the corrupted one. For soundness, this is captured by the following definitions.

Definition 1 (**Soundness of dispute resolution process**)**.** *The dispute resolution process* `dispute` *is sound*

- *in case of corrupted administrator and honest candidate if it emits* `Cguilty` *and* `CAguilty` *in no execution trace of the exam protocol;*
- *in case of corrupted candidate and honest administrator if it emits* `Aguilty` *and* `CAguilty` *in no execution trace of the exam protocol;*
- *in case of corrupted administrator and candidate if it emits* `Cguilty` *and* `Aguilty` *in no execution trace of the exam protocol.*

To prove completeness, we check that the exam protocol never runs the process `dispute`, hence events `Cguilty`, `Aguilty`, and `CAguilty` are not emitted. This is captured by the following definition.

Definition 2 (Completeness of dispute resolution process). *The dispute resolution process* `dispute` *is complete if it emits the events* `Aguilty`*,* `Cguilty`*, and* `CAguilty` *in no execution trace of the protocol with honest roles.*

Requirement 15 (Testing Dispute Resolution). *An exam protocol ensures Testing Dispute Resolution if the dispute resolution process* `dispute` *is sound and complete.*

4. The protocol

In a nutshell, the protocol works as follows. At preparation, candidate and administrator jointly generate the candidate's pseudonym (an alphanumeric *pid*) as a pair of visual cryptography shares, by means of an oblivious transfer scheme. One share is held by the candidate, who prints it on a paper sheet together with the candidate ID and signatures meant for integrity and accountability purposes. The other share is printed by the administrator as a transparency printout and handed to the invigilator before testing. Each share alone does not reveal the pseudonym, which is revealed only when the two shares are overlapped. This is possible only at testing, when the candidate and the invigilator physically meet, and the latter hands the transparency to the candidate. Any dispute that may happen at testing can be resolved thanks to the signatures printed with the printouts. The candidate can write the pseudonym on the answer sheet, and testing concludes when all answer sheets are returned to the examiner. At marking, the examiner evaluates the answers and assigns a mark to each pseudonym, which she commits to and publishes on a bulletin board. At notification, a candidate can retrieve her mark by proving to the administrator that she owns the share that (re)-reveals the pseudonym. The administrator's share is required in this phase, but there is no need for the candidate and the administrator to meet. The candidate sends her share and the signatures to the administrator, and any dispute happening at notification can be again solved using the signatures associated with the shares.

The protocol combines a few cryptographic primitives, namely visual cryptography, commitment, and oblivious transfer schemes. These are briefly expanded upon here before the protocol is detailed.

4.1. Visual Cryptography

It is a secret sharing scheme devised by Naor and Shamir [34] for a visual decryption of a ciphertext. A secret image is "encrypted" by splitting it into a number of image *shares*. The basic version of the scheme is the 2-out-of-2 secret sharing system, in which a secret image is split into two shares $share_A$ and $share_B$. The shares are printed on transparency sheets, which reveal the secret image when the shares are overlapped. This scheme is information-theoretic secure, namely each share leaks no information about the secret image. While a visual decryption by means of an overlay emulates the OR operator, visual cryptography cleverly emulates the XOR. The scheme is information-theoretic secure because of either a black or a white pixel, mapped respectively to 0 and 1, can originate from any of the sub-pixels shown in Figure 2.

4.2. Commitment Scheme

A commitment scheme is used to bind a committer to a secret value. The committer publishes a commitment that hides the value, which remains secret until he reveals it. Should the committer reveal a different value, this would be noticed because it could not be mapped to the published commitment. The Pedersen commitment scheme [35] guarantees unconditional hiding, namely the value remains secret despite a computationally unbounded attacker. The scheme consists of the algorithms of *commitment*, in which the value is chosen, hidden, and bound to the committer, and of *disclosure*, in which the value is publicly revealed. The commitment algorithm takes in two given public generators $g, h \in \mathbb{G}_q$, the secret value v, and a random value $r \in_{\mathcal{R}} \mathbb{Z}_q^*$. The algorithm outputs the commitment $g^v h^r$ denoted with $C_r(v)$. The disclosure algorithm takes in the commitment $C_r(v)$, the values v and r, and outputs `true` if the commitment is correct or `false` otherwise.

Our protocol uses a generalised Pedersen commitment scheme [36], which guarantees unconditional hiding and allows the commitment to many values at once. For example, we take advantage of the Pedersen commitment scheme at notification. The administrator generates a commitment of the mark of the candidate. Once the candidate reveals her identity to learn the mark, she can verify that the administrator notifies her the committed mark. This deters the administrator to notify the candidate with a mark that would differ from the one the examiner assigned to candidate's test.

4.3. Oblivious transfer

Oblivious transfer schemes allow a chooser to pick some pieces of information from a set a sender offers him, in such a way that (a) the sender does not learn which pieces of information the chooser picks, and (b) the chooser learns no more than the pieces of information he picks. Our protocol adopts Tzeng's oblivious transfer scheme [37]. In Tzeng's scheme, the chooser commits to some elements from a set and sends the commitments to the sender. This, in turn, obfuscates all the elements of the set, and the chooser will be able to de-obfuscate only the elements he committed to. Tzeng's scheme guarantees unconditional security for the receiver's choice, and it is efficient since it works with the sender and receiver's exchanging only two messages.

4.4. Threat model and assumptions

In addition to the threats derived from a standard Dolev-Yao attacker, we consider the following specific threats coming from the exam roles we specified above.

- Corrupted candidates, who may want to register for an exam without being eligible or on behalf of someone else; or be assigned with a mark higher than what the examiner assigns to their test.
- Corrupted authorities (i.e., administrator, invigilator, and examiner), who may want to assign an unfair mark to a specific candidate, namely to over-mark or under-mark her, or assign no mark at all.
- An attacker, who may want to get any private information or tamper with tests and marks.

Like any other security protocol, ours is not designed to withstand every possible threat. For example, it cannot cope with plagiarism, but assumes appropriate invigilation during testing. Principals may still collude and communicate via subliminal channels, for example by using steganography. Although it is hard to rule out completely such a threat, steganalysis techniques can be of some help here. Other countermeasures may be needed against collusion attacks that exploit covert channels. We thus specify five assumptions conveniently for the goals of our protocol. In particular, we assume that:

1. Administrator and examiner hold a long-term public/private pair of keys.
2. The candidate is invigilated during testing to mitigate cheating.
3. The model answers are kept secret from the candidates until after testing. The examiners may be provided with the model answers at marking.
4. An authenticated append-only bulletin board is available. It guarantees everyone to see the same data, though write access might be restricted to appropriate entities [38]. (An implementation of a bulletin board and its formal analysis exists [39].)
5. A TLS channel that ensures integrity and confidentiality of messages is available. Remote communications between administrator and candidate occur via TLS.

While we do not focus on a particular form of written exam (e.g., multiple choice, short answer, essay), we assume that the administrator populates each test with a set of questions, and that the pile of tests is available at exam venue. Each candidate will pick a random test from the pile and will answer the corresponding questions while supervised by an invigilator.

We note that if a candidate and the administrator collude, the latter may reveal the exam questions in advance, a scenario that is difficult to address in a security protocol, and that collides with our assumption 4. However, some methods can mitigate the effect of such scenario. One is to generate different piles of tests. Tests belonging to the same pile contain the same questions. At exam venue, one pile is randomly chosen and its tests serve for the exam. In doing so, each candidate receives the same set of questions, but without knowing which one until after testing. Another solution is to populate each test with different questions that are chosen from a very large, possibly public, set of questions. In doing so, each candidate receives a different set of questions.

4.5. Description of the protocol in detail

We describe our protocol in reference to the four exam phases. In the description we assume a few public parameters, namely:

n	length of the candidate's pseudonym
$\Sigma = \{s_1, \dots, s_k\}$	alphabet of pseudonym's characters
$c_j \in \{0,1\}^{t \times u},\ j = 1, \dots, k$	$(t \times u)$-pixel representation of a character
idC	candidate ID
ex	exam code
SPK_A	signing key of the administrator
SPK_E	signing key of the examiner
M	set of possible marks
$g, h \in_{\mathcal{R}} \mathbb{G}_q$	generators for bit-commitments

4.6. Preparation

The goal of preparation is to generate a candidate's pseudonym, which is a string of n characters taken from alphabet Σ, and to encode it into two visual cryptographic shares. No one can know the pseudonym until candidate and invigilator meet at testing, when the candidate learns her pseudonym by overlapping the administrator's share with hers. The underlying idea is that the candidate provides a commitment to an index into an array. The administrator fills the array with a secret permutation of the characters, and only when the two secrets are brought together is the selection of a character determined.

Part of this phase is inspired by one of the schemes used to print a secret, proposed by Essex *et al.* [40]. We tailor that scheme so that it can generate a pseudonym as detailed in Figure 3. This modification also supports the dispute resolution algorithm, as we shall see below. The main technical differences between our preparation phase and the original scheme to print a secret are: (a) a modified oblivious transfer protocol that copes with several secret messages in only one protocol run; (b) the generation of signatures that will be used for accountability in the resolution of disputes.

In the following, we refer to the steps described in Figure 3. The protocol begins with the candidate providing a sequence of l commitments y_i to an index into an array of length k (steps 1-2). More precisely, the parameter l is chosen so that the $l - n$ elements can be later used for a cut-and-choose audit. The administrator can challenge the candidate to check whether the committed choices are in fact in the interval $[1, k]$. Otherwise, the administrator generates a sequence of randomly chosen $t \times u$ images, indicated as $\alpha_1, \dots, \alpha_l$ in Figure 3. A sequence of k images $\beta_{i1}, \dots, \beta_{ik}$ are generated from α_i and each possible character c_j. The sequence is randomly permuted and repeated for all i, resulting in l sequences of images $(\beta_{11}, \dots, \beta_{1k})$, $\dots$, $(\beta_{l1}, \dots, \beta_{lk})$. The secret permutation and the commitment allow the selection of a character to be determined only when the two secrets are brought together.

The administrator then generates the obfuscation ω_{ij} from each β_{ij} and a commitment on each α_i, indicated as *com* (step 3), which is signed and sent with the sequences of obfuscations $(\omega_{11}, \ldots, \omega_{1k})$,..., $(\omega_{l1}, \ldots, \omega_{lk})$ to the candidate (step 4). The obfuscation allows the candidate to retrieve only the elements whose indexes correspond to the choices $y_1, \ldots, y_l$ she committed to in step 1.

The candidate performs a cut-and-choose audit, selecting a random set of $l - n$ sequences amongst the ω. By doing so, she can check whether the administrator generated the sequence of images correctly. The remaining substitutions $\sigma_1, \sigma_2, \ldots, \sigma_n$ select the indexes of the images that make the pseudonym. Thus, the visual share of the administrator consists of the concatenated images $\alpha_{\sigma_1}, \ldots, \alpha_{\sigma_n}$ (steps 5-6).

The administrator then generates the proofs for the cut-and-choose audit and prints the visual share and the candidate's details in the transparency printout `transp`. This also includes the secret value s used for the commitment of all the elements $\alpha_1, \ldots, \alpha_l$ (step 7). The secret value is represented in the form of a QR code. The administrator generates a signature that contains the candidate's commitments $y_1, \ldots, y_l$, the sequence of images used for the cut-and-choose audit $\alpha_1, \ldots, \alpha_m$, and the sequences of selected obfuscations $(\omega_{\sigma_1 1}, \ldots, \omega_{\sigma_1 k}), \ldots (\omega_{\sigma_n 1}, \ldots, \omega_{\sigma_n k})$. The administrator then sends the signature and the proofs to the candidate (step 8). In turn, the candidate checks the signature and the proofs, de-obfuscates the elements ω, and retrieves the visual share consisting of the concatenated image $\beta_{\sigma_1}, \beta_{\sigma_2}, \ldots, \beta_{\sigma_n}$. She finally prints the share, together with the two signatures, on a `paper` printout (step 9). At this point, both candidate and administrator have a visual share each; it is these two shares that, once overlapped, return an intelligible sequence of characters that serves as candidate's pseudonym.

The candidate's paper printout includes two QR codes (*QR1* and *QR2*) while the administrator's transparency only one (*QR3*). All these codes refer to the same candidate identity *idC* and exam identifier *ex*. The QR codes *QR1* and *QR2* notably encode the two signatures of the administrator respectively, while *QR3* encodes the secret value s. This phase concludes with the administrator handing the transparency to the invigilator (step 10).

4.7. Testing

The steps of this phase are described in Figure 4. The invigilator leaves a pile of tests on a desk at the exam venue. The candidate brings the paper printout at exam venue, while the invigilator brings the transparencies. The invigilator authenticates the candidate by checking her identity document (steps 11-12). He then gives the candidate her corresponding transparency and invites the candidate to pick a test randomly (steps 13-14). The candidate overlaps her paper printout with the transparency and learns her pseudonym, which she writes on the test. If no pseudonym appears, then this may happen only if the candidate or the administrator misprinted their printouts, and the Testing Dispute Resolution algorithm outlined in Algorithm 1 reveals the principal that is accountable for the misbehaviour. At the end of the phase, the candidate returns the filled test by inserting it anywhere in the pile of tests (step 15), and takes both transparency and paper printouts home. The randomness exercised in dispatching the tests (to the candidates) and in returning the filled tests (to the invigilator) thwarts the risk that the invigilator builds visual associations between a test and a candidate.

4.8. Marking

At marking (see Figure 5), the invigilator hands the filled tests to an examiner (step 16). For each filled test, the latter evaluates the answers, assigns a mark, and generates a signature on the triplet formed by pseudonym, answers and mark (step 17). Then, the examiner sends the triplet and the signature to the administrator (step 18). The administrator generates a commitment on the assigned mark (step 19), signs pseudonym, answers, and the commitment, and finally publishes the signature on the bulletin board (step 20).

4.9. Notification

We refer to the steps described in Figure 6 for notification. This phase opens for a fixed time during which the candidate can remotely request to learn her mark and having it registered. She has to send the ordered sequences of $\beta_1, \ldots, \beta_n$, her pseudonym, and all the signatures she collected so far to the administrator (step 21). The administrator checks the signatures, overlaps the given sequence with the corresponding sequences of $\alpha_1, \ldots, \alpha_n$, and checks the pseudonym. Again, if no registered pseudonym appears, Dispute Resolution can reveal the principal who misbehaved. The administrator signs mark, pseudonym, and the secret parameter used to commit the mark (step 22), and sends the signature to the candidate (step 23). By doing so, the candidate can verify the correctness of the mark by looking at the bulletin board.

4.10. Dispute resolution

An interesting feature of our protocol is the support for dispute resolution during testing. The combination of signatures and visual cryptography guarantees an easy procedure to find the culprit if the candidate and/or the administrator misbehave. Therefore, Dispute Resolution qualifies as an accountability requirement as it enables a judge to blame the principal who misbehaved in the execution of the protocol.

In our protocol the judge is the invigilator, and the dispute originates if no intelligible pseudonym can be read when the candidate overlaps the paper sheet with the transparency sheet. Should such a dispute arise, the invigilator could then quickly resolve it by following Algorithm 1, which corrects the previous version [8] in case both principals misbehave. We assume that an electronic device with a camera is available at the exam venue. It could be a smartphone or a tablet, and it should store the public key of the administrator. The input of the algorithm are the two QR codes printed on the paper printout (QR1 and QR2) and the QR code printed on the transparency (QR3), all scanned with the camera of the electronic device. Note that the trustworthiness of dispute resolution comes from the correctness (soundness and completeness) of the algorithm, which is open to anyone for checking. Some trust is still required on the device used to resolve the dispute; however, it is always possible to run dispute resolution on any other device.

The goal of the algorithm is to reconstruct the correct visual shares, if possible. First, the algorithm checks the correctness of the signatures encoded in QR1 and QR2. If any of the checks fails, then the correct visual shares cannot be reconstructed. However, this reveals that the candidate misprinted her paper printout, thus she is the culprit. Otherwise, the algorithm reconstructs the correct visual share of the candidate by checking the candidate's commitments and the obfuscation — both signed by the administrator. If the check reveals that the reconstructed visual share matches the one printed by the candidate, then the culprit is the administrator. Otherwise, the algorithm reconstructs the administrator's visual share by checking the correctness of the administrator's commitment using the secret value encoded in QR3. If the check succeeds, then the candidate is the culprit, otherwise both candidate and examiner misprinted their visual shares.

Algorithm 1: Testing dispute resolution

```
Data: Public parameters: $(C, n, g_i, h, idC, ex, SPK_A)$
    - $paper = (\beta_{\sigma_1}, \beta_{\sigma_2}, \ldots, \beta_{\sigma_n}), idC, ex, sign1, sign2, (x_{\sigma_1}, x_{\sigma_2}, \ldots, x_{\sigma_n}),$
          $(\gamma_{\sigma_1}, \gamma_{\sigma_2}, \ldots, \gamma_{\sigma_n})$ where
        - $sign1 = Sign_{SSK_A}\{idC, ex, com_A\}.$
        - $sign2 = Sign_{SSK_A}\{idC, ex, (y_{\sigma_1}, y_{\sigma_2}, \ldots, y_{\sigma_n}), (\alpha_{\chi_1}, \alpha_{\chi_2}, \ldots, \alpha_{\chi_m}),$
              $(\omega_{\sigma_1 1}, \ldots, \omega_{\sigma_1 k}), \ldots (\omega_{\sigma_n 1}, \ldots, \omega_{\sigma_n k})\}.$
    - $transp = (\alpha_{\sigma_1}, \alpha_{\sigma_2}, \ldots, \alpha_{\sigma_n}), idC, ex, s.$

Result: Corrupted participant(s)
if sign1 is verifiable with $SPK_A$ and sign2 is verifiable with $SPK_A$ then
    if $y_{\sigma_j} \neq g^{x_{\sigma_j}} h^{\gamma_{\sigma_j}}$ or $\beta_{\sigma_j} \neq \frac{b_{\sigma_j \gamma_j}}{(a_{\sigma_j \gamma_j})^{x_{\sigma_j}}}$ $[j = 1, 2, \ldots n]$ then
        if $com_A \neq h^s \prod_{i=1}^{l} {g_i}^{\alpha_i}$ then
            return Administrator and Candidate
        else
            return Candidate
    else
        return Administrator
else
    return Candidate
```

5. Analysis

We analyse our protocol in ProVerif[1], a security protocol verifier that supports the automatic analysis of authentication and privacy properties in the Dolev-Yao attacker model [41]. The input language of ProVerif is a variant of the applied π-calculus.

5.1. Modelling Choices

We model TLS and face-to-face communications between the roles using the cryptographic primitive of probabilistic symmetric encryption rather than using ProVerif's private channels. The attacker cannot monitor communications via ProVerif's private channels and cannot even know if any communication takes place, and we think this would be an overly strong assumption that could miss attacks. By renouncing to private channels, we achieve stronger security guarantees for the analysis of the protocol. Moreover, our choice has a triple advantage: i) it gives the attacker more discretional power because he can observe when a candidate registers for the exam, when she is given the test, when she submits the answers, and when she is notified with a mark; ii) it allows modelling either corrupted candidate or examiner by just sharing the private key with the attacker; iii) it increases the chances that verification attempts in ProVerif terminate.

We use the equational theory illustrated in Table 1 to model the cryptographic primitives of the protocol. The theory for probabilistic symmetric key consists of two functions *senc* and *sdec*. A message encrypted with a private key can only be decrypted using the same private key. Note that the randomness r on the encryption algorithm causes that the same message encrypted several times outputs different ciphertexts. The equational theory for the digital signature is rather standard in ProVerif.

We introduce a novel theory to model oblivious transfer and visual cryptography. The function *obf* allows the examiner to obfuscate the elements $\beta_1, \ldots, \beta_i$, while the function *deobf* returns the correct element β_{sel} to the candidate, depending on the choice she committed to. We also provide the theory for the Pedersen commitment scheme with the function *commit*. Finally, we model the generation of a visual cryptography share with *gen_share*, and their overlapping with the function *overlap*.

We model an unbounded number of corrupted candidates who can register for the exam. All the processes are augmented with the events that allow for the verification of authentication requirements.

5.2. Verification Choices

We verify Anonymous Examiner in presence of corrupted candidates and Anonymous Marking in presence of corrupted administrator, examiner and co-

[1]The full ProVerif code is available at `https://sites.google.com/site/sarogiustolisi/cose16.tar.gz`

candidates, which means that the attacker can control an unbounded number of candidates while some are honest. To verify Question Indistinguishability, we consider corrupted candidates, while for both Mark Privacy and Mark Anonymity we consider corrupted eligible candidates who can register for the exam but cannot participate at testing.

To verify Mark Integrity Verifiability, we model the verifiability-test *testMI* as in Figure 2. It takes in, via a private channel, the pseudonym *pid*, the secret value v, and the mark notified to the candidate. It also takes as input the signed notification *sign3* containing the pseudonym, the answers, and the committed mark from the bulletin board. The verifiability-test checks if the administrator's signature is correct and the disclosure of the commitment contained in the signed notification reveals the mark provided by the candidate. We verify the soundness of the test in the presence of corrupted administrator, examiner, invigilator, and co-candidates. The verifiability-test *testTIV* depicted in Figure 3 takes in the pseudonym *pid* and the answer submitted by the candidate. It also takes as input the signed notification *sign3* containing the pseudonym, the answers and the committed mark from the bulletin board. The verifiability-test checks if the administrator's signature is correct and the signed answer matches the answer submitted by the candidate. We also verify the soundness of this test in the presence of corrupted administrator, examiner, invigilator, and co-candidates.

Algorithm 2: The verifiability-test for Mark Integrity Verifiability

```
Data: Public parameters: $(g, h, SPK_A)$
    - $sign3 = Sign_{SSK_A}(pid, answers, c)$
    - $idC, pid', mark, v.$
Result: Whether the candidate was notified with the mark assigned to
        her test.
if $pid = pid'$ and $c = g^v h^{mark}$ then
  | return true
else
  | return false
```

5.3. Limitations

A limitation of the formal model is the specification of the cut-and-choose audit due to the powerful ProVerif's attacker model. In fact, if the attacker plays the cutter's role, he might cut the set of elements such that the subset audited by the chooser is correct, while the other subset is not. Although in reality the probability of success of this attack for a large set of elements is small, it is a valid attack in ProVerif irrespective of the number of elements. In our case, the chooser is the candidate and the cutter is the examiner. ProVerif thus finds a false attack when the examiner is corrupted, namely controlled by the attacker. We resolve this false-positive by allowing the candidate to check

Algorithm 3: The verifiability-test for Test Integrity Verifiability

```
Data: Public parameters: $(g, h, SPK_A)$
    - $sign3 = Sign_{SSK_A}(pid, answers, c)$
    - $idC, pid', answers', v$.
Result: Whether the candidate was notified with the mark assigned to
        her test.
if $pid = pid'$ and $answers = answers'$ then
    return true
else
    return false
```

all the elements of the set. This is sound because the candidate plays the role of the chooser, thus she is honest and follows the protocol although she knows the extra information.

Another limitation concerns the analysis of the soundness of dispute resolution with respect to corrupted administrator and candidate. The classic way to model the scenario of corrupted principals is to let the attacker control them. However, we observe that such a modelling does not work in this case because the conflicting goals of administrator and candidate are prerequisites for dispute resolution. Ideally, we would need administrator and candidate to be controlled by two different attackers that are only limited by the conflicting goals. Unfortunately, this scenario cannot be specified in ProVerif as the tool operates in the Dolev-Yao attacker model. Thus, we check the soundness of dispute resolution in the case of corrupted administrator and candidate by generating two correct printouts and by sharing both with the attacker. We then prove that if the attacker feeds `dispute` with any two printouts that are both different from the correct ones, the events `Cguilty` and `Aguilty` are emitted in no execution trace of the exam protocol.

5.4. Results

Table 2 outlines the results of our analysis and the execution times of ProVerif over an Intel Core i7 2.6 GHz machine with 8 GB RAM. ProVerif confirms that the protocol guarantees all the authentication requirements even under an unbounded number of corrupted, eligible co-candidates. Thus, the protocol ensures authentication although the attacker can register to the exam. Partitioning the examiner role as administrator, examiner and invigilator enables a finer analysis: we can now prove in ProVerif that the protocol ensures most of the authentication requirements when considering both corrupted administrator and examiners. This was not possible to prove in the conference version [8] of the present article. However, we cannot still prove Candidate Authorisation and Mark Authentication because it is not possible to model in ProVerif the related correspondence assertions assuming corrupted administrators.

Regarding privacy, ProVerif proves that the protocol guarantees all the privacy requirements. In particular, the protocol meets Anonymous Marking considering corrupted administrator and examiner, and Anonymous Examiner considering corrupted candidates;

ProVerif confirms that the verifiability-tests *testMIV* and *testTIV* are sound and complete, thus it can be claimed that the exam protocol is Mark Integrity and Test Integrity verifiable.

Finally, the exam protocol ensures Dispute Resolution: ProVerif shows that the protocol does not blame honest principals when the `dispute` algorithm is executed or blames those who are corrupted (soundness). In particular, it blames both administrator and examiner when they are both corrupted. This result is possible thanks to the updates to the exam protocol. ProVerif also shows that the pseudonym is always revealed, namely the dispute algorithm is not run, when both examiner and candidate are honest (completeness).

6. Conclusions and Future Work

This paper draws its motivation by observing that exam security has a major role in the widespread acceptance of exams, especially when they are assisted by computers. It has found that it is possible to provide a secure exam protocol with the design principle of minimising the reliance on the trusted parties. Notably, the protocol guarantees some form of accountability without relying on a TTP. The underlying idea of the protocol is to combine oblivious transfer and visual cryptography to generate a pseudonym that anonymises the tests while they are marked. This protocol is more detailed than the previous one and meets a larger set of security requirements with only minimal assumptions; this was confirmed using a formal approach whereby the security of the protocol was analysed extensively.

Future work can be envisaged over various directions. One is to analyse the protocol in the computational model to achieve finer security guarantees, perhaps using computer-assisted tools like CryptoVerif [42] or EasyCrypt [43]. It should be possible to study compositional proofs that integrate computational proofs of the cryptographic primitives used in our protocol with the symbolic ones obtained in ProVerif. Another direction is to complement the protocol with techniques to detect plagiarism and candidate cheating at testing. This is important when testing is done via computers. For example, techniques similar to those described by Pieczul and Foley [44] could be useful for this purpose. Another extension may be the support for collaborative marking, in which the questions are categorised by subject, and examiners evaluate only the answers that pertain to the examiner subject area.

We believe that the idea behind the protocol may be useful for the design of similar systems, such as for public tenders, project reviews, and conference management systems. We feel that, by promoting a fairer assessment of knowledge and skills through the many application scenarios that award a qualification or a post to people, this line of research can significantly contribute to advancing modern meritocracy.

Acknowledgements. The authors thank the FNR, CORE project C11/IS/1183245 "Socio-Technical Analysis of Security and Trust".

References

[1] M. Kazin, R. Edwards, A. Rothman, The Princeton Encyclopedia of American Political History. (Two volume set), Princeton University Press, 2009.

[2] E. Flock, APS embroiled in cheating scandal, Washington Post, July, 2011.

[3] R. Watson, Student visa system fraud exposed in BBC investigation, `http://www.bbc.com/news/uk-26024375` (February 2014).

[4] B. News, Vyapam: India's deadly medical school exam scandal, `http://www.bbc.com/news/world-asia-india-33421572` (July 2015).

[5] K. Liptak, U.S. Navy discloses nuclear exam cheating, `http://edition.cnn.com/2014/02/04/us/navy-cheating-investigation/` (February 2014).

[6] A. Press, Navy kicks out 34 for nuke test cheating, `http://www.foxnews.com/us/2014/08/21/navy-kicks-out-34-for-nuke-cheating/` (August 2014).

[7] B. Blanchet, An Efficient Cryptographic Protocol Verifier Based on Prolog Rules, in: CSFW '01, IEEE, 2001, pp. 82–96.

[8] G. Bella, R. Giustolisi, G. Lenzini, P. Y. Ryan, A secure exam protocol without trusted parties, in: ICT Systems Security and Privacy Protection, Vol. 455 of IFIP Advances in Information and Communication Technology, Springer International Publishing, 2015, pp. 495–509. `doi:10.1007/978-3-319-18467-8_33`.

[9] J. Dreier, R. Giustolisi, A. Kassem, P. Lafourcade, G. Lenzini, A framework for analyzing verifiability in traditional and electronic exams, in: Information Security Practice and Experience, Vol. 9065 of Lecture Notes in Computer Science, Springer International Publishing, 2015, pp. 514–529. `doi:10.1007/978-3-319-17533-1_35`.

[10] E. Weippl, Security in E-learning, Vol. 6 of Advances in Information Security, Springer Science + Business Media, 2005.

[11] S. Furnell, P. D. Onions, M. Knahl, P. W. Sanders, U. Bleimann, U. Gojny, H. F. Röder, A security framework for online distance learning and training, Internet Research 8 (3) (1998) 236–242. `doi:10.1108/10662249810217821`.

[12] J. Dreier, R. Giustolisi, A. Kassem, P. Lafourcade, G. Lenzini, P. Y. A. Ryan, Formal Analysis of Electronic Exams, in: SECRYPT 2014, SciTePress, 2014.

[13] S. N. Foley, J. L. Jacob, Specifying Security for Computer Supported Collaborative Working, J. of Computer Security 3 (1995) 233–253.

[14] J. Castella-Roca, J. Herrera-Joancomarti, A. Dorca-Josa, A Secure E-Exam Management System, in: ARES 2006, IEEE, 2006.

[15] G. Bella, R. Giustolisi, G. Lenzini, Secure Exams Despite Malicious Management, in: PST '14, IEEE, 2014, pp. 274–281.

[16] A. Huszti, A. Pethö, A secure Electronic Exam System, Publicationes Mathematicae Debrecen 77 (3-4) (2010) 299–312.

[17] R. Giustolisi, G. Lenzini, P. Ryan, Remark!: A Secure Protocol for Remote Exams, in: Security Protocols XXII, Vol. 8809 of LNCS, Springer, 2014, pp. 38–48.

[18] Test of English as a Foreign Language., `http://www.ets.org/toefl`.

[19] T. Lewin, Students Rush to Web Classes, but Profits May Be Much Later, `http://www.nytimes.com/2013/01/07/education/massive-open-online-courses-prove-popular-if-not-lucrative-yet.html?_r=0` (January 2013).

[20] ETS, Tests and Products, `https://www.ets.org/tests_products/alpha` (July 2015).

[21] Pearson, Pearson VUE delivered exams, `http://home.pearsonvue.com/test-taker/All-Tests.aspx` (July 2015).

[22] E. P. S. Office, Careers with the european union, `http:\europa.eu/epso/apply/how_apply/index_en.htm` (November 2013).

[23] M. Maffei, K. Pecina, M. Reinert, Security and Privacy by Declarative Design, in: CSF 2013, IEEE, 2013, pp. 81–96.

[24] S. Hohenberger, S. Myers, R. Pass, a. shelat, Anonize: A large-scale anonymous survey system, in: Proceedings of the 2014 IEEE Symposium on Security and Privacy, SP '14, IEEE Computer Society, Washington, DC, USA, 2014, pp. 375–389. `doi:10.1109/SP.2014.31`.

[25] S. Kanav, P. Lammich, A. Popescu, A Conference Management System with Verified Document Confidentiality, in: CAV 2014, Vol. 8559 of LNCS, Springer, 2014, pp. 167–183.

[26] M. Arapinis, S. Bursuc, M. Ryan, Privacy supporting cloud computing: Confichair, a case study, in: Principles of Security and Trust (POST), Vol. 7215 of LNCS, Springer-Verlag, 2012, pp. 89–108.

[27] M. Arapinis, S. Bursuc, M. Ryan, Privacy-supporting cloud computing by in-browser key translation., J. of Computer Security 21 (6) (2013) 847–880.

[28] INFOSAFE, http://www.anonymousmarking.com/.

[29] J.-P. Moussette, Method for anonymous computerized processing of documents or object. Patent, eP 1430439 B1 (08 2009).
URL http://www.patentlens.net/patentlens/patent/EP_1430439_B1/en/

[30] M. Abadi, C. Fournet, Mobile Values, New Names, and Secure Communication, in: POPL 01, ACM, 2001.

[31] T. Y. Woo, S. S. Lam, A semantic model for authentication protocols, in: Research in Security and Privacy, 1993. Proceedings., 1993 IEEE Computer Society Symposium on, 1993, pp. 178–194.

[32] J. Dreier, H. Jonker, P. Lafourcade, Defining verifiability in e-auction protocols, in: Proceedings of the 8th ACM SIGSAC symposium on Information, computer and communications security, ASIA CCS '13, ACM, New York, NY, USA, 2013, pp. 547–552.

[33] S. Delaune, S. Kremer, M. Ryan, Coercion-resistance and receipt-freeness in electronic voting, in: Proceedings of the 19th IEEE Workshop on Computer Security Foundations, CSFW '06, IEEE Computer Society, Washington, DC, USA, 2006, pp. 28–42. doi:10.1109/CSFW.2006.8.

[34] M. Naor, A. Shamir, Visual cryptography, in: EUROCRYPT, 1994.

[35] T. Pedersen, Non-Interactive and Information-Theoretic Secure Verifiable Secret Sharing, in: EUROCRYPT 91, Vol. 576 of LNCS, Springer, 1991, pp. 129–140.

[36] T. P. Pedersen, Non-interactive and information-theoretic secure verifiable secret sharing, in: J. Feigenbaum (Ed.), CRYPTO, LNCS, Springer, 1992, pp. 129–140.

[37] W.-G. Tzeng, Efficient 1-out-of-n Oblivious Transfer Schemes with Universally Usable Parameters, IEEE Trans. on Computers 53 (2) (2004) 232–240.

[38] J. Benaloh, P. Y. Ryan, V. Teague, Verifiable postal voting, in: Security Protocols XXI, Vol. 8263 of Lecture Notes in Computer Science, Springer Berlin Heidelberg, 2013, pp. 54–65. doi:10.1007/978-3-642-41717-7_8.

[39] C. Culnane, S. Schneider, A peered bulletin board for robust use in verifiable voting systems, in: Computer Security Foundations Symposium (CSF), 2014 IEEE 27th, 2014, pp. 169–183. doi:10.1109/CSF.2014.20.

[40] A. Essex, J. Clark, U. Hengartner, C. Adams, How to Print a Secret, in: HotSec 09, USENIX Association, 2009.

[41] D. Dolev, A. C. Yao, On the Security of Public Key Protocols, IEEE Trans. on Information Theory 29 (2) (1983) 198–208.

[42] B. Blanchet, A computationally sound mechanized prover for security protocols, IEEE Transactions on Dependable and Secure Computing 5 (4) (2008) 193–207.

[43] G. Barthe, B. Grégoire, S. Zanella Béguelin, Formal certification of code-based cryptographic proofs, SIGPLAN Not. 44 (1) (2009) 90–101. `doi:10.1145/1594834.1480894`.

[44] O. Pieczul, S. Foley, Collaborating as normal: detecting systemic anomalies in your partner, in: Proc. of 22nd Int. Security Protocols Workshop, Cambridge, 2014.

Primitive	**Equation**
Prob. symmetric enc.	$sdec(senc(m,k,r),k) = m$
Digital signature	$getmess(sign(m,ssk)) = m$
	$checksign(sign(m,ssk),spk(ssk)) = m$
Oblivious transfer	$deobf(obf(r,m,\texttt{sel},commit(r',\texttt{sel})),r') = m$
Visual cryptography	$overlap(share,gen\_share(m,share)) = m$
	$overlap(share,share) = share$

Table 1: Equational theory to model our protocol

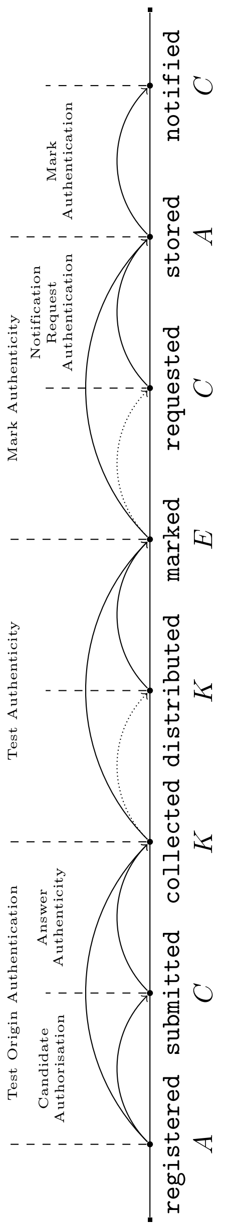

Figure 1: A graphical representation of the authentication requirements for exams

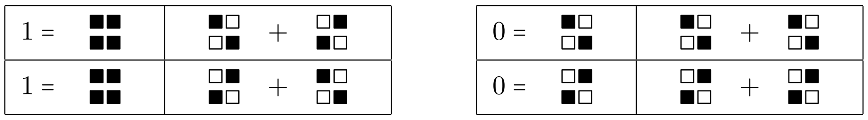

Figure 2: Representation of bits using visual cryptography

No.	**Requirement name**	**Result**	**Time**
1.	Candidate Authorisation	✓	6 s
2.	Answer Authenticity	✓	5 s
3.	Test Origin Authentication	✓	5 s
4.	Test Authenticity	✓	6 s
5.	Mark Authenticity	✓	6 s
6.	Notification Request Auth.	✓	6 s
7.	Mark Authentication	✓	6 s
8.	Question Indistinguishability	✓	<1 s
9.	Anonymous Marking	✓	1m 5s
10.	Anonymous Examiner	✓	2 m 19 s
11.	Mark Privacy	✓	32 m 23 s
12.	Mark Anonymity	✓	9 m 12 s
13.	Mark Integrity Verifiability	✓	<1s
14.	Test Integrity Verifiability	✓	<1s
15.	Dispute Resolution	✓	<1s

Table 2: Summary of the analysis of our protocol

1. C calculates $y_i = g^{x_i} h^{\gamma_i}$ where:
   - $x_i \in_{\mathcal{R}} \mathbb{Z}_q^*$.
   - $\gamma_i \in_{\mathcal{R}} [1, k]$.
   - $i = 1, 2, \dots, l$ with $l > n$.

2. $C \to A$: $y_1, y_2, \dots, y_l$.

3. A calculates $\beta_{ij} \leftarrow_{\pi_{\mathcal{R}}} (\alpha_i \oplus c_j)$, $\omega_{ij} = \langle a_{ij}, b_{ij} \rangle \leftarrow \langle g^{r_{ij}}, \beta_{ij} \left(\frac{y_i}{h^j}\right)^{r_{ij}} \rangle$, $com_A = h^s \prod_{i=1}^{l} {g_i}^{\alpha_i}$, and $sign1 = Sign_{SSK_A}(idC, ex, com_A)$ where:
   - $\alpha_i \in_{\mathcal{R}} [0, 1]^{t \times u}$.
   - $s, r_{ij} \in_{\mathcal{R}} \mathbb{Z}_q^*$.
   - $g_i \in_{\mathcal{R}} \mathbb{G}_q$.
   - $i = 1, 2, \dots, l$.
   - $j = 1, 2, \dots, k$.

   or runs the challenge procedure against $y_1, y_2, \dots, y_l$.

4. $A \to C$: $(\omega_{11}, \dots, \omega_{1k}), \dots (\omega_{l1}, \dots, \omega_{lk})$ and $sign1$.

5. C calculates $\chi_i \in [1, l]$ and $\sigma_j \in [1, l]$ where:
   - $i = 1, 2, \dots, m$.

6. $C \to A$: $\chi_1, \chi_2, \dots, \chi_m$ and $\sigma_1, \sigma_2, \dots, \sigma_n$.

7. A calculates $ev_{\chi_i} = \langle \alpha_{\chi_i}, (\beta_{\chi_i 1}, \beta_{\chi_i 2}, \dots, \beta_{\chi_i k}), (r_{\chi_i 1}, r_{\chi_i 2}, \dots, r_{\chi_i k}) \rangle$ and $sign2 = Sign_{SSK_A}(idC, ex, (y_{\sigma_1}, y_{\sigma_2}, \dots, y_{\sigma_n}), (\alpha_{\chi_1}, \alpha_{\chi_2}, \dots, \alpha_{\chi_m}),$ $(\omega_{\sigma_1 1}, \dots, \omega_{\sigma_1 k}), \dots (\omega_{\sigma_n 1}, \dots, \omega_{\sigma_n k})\}$ where
   - $i = 1, 2, \dots, m$.
   - $j = 1, 2, \dots, k$.

   and prints `transp` $= \langle (\alpha_{\sigma_1}, \alpha_{\sigma_2}, \dots, \alpha_{\sigma_n}), idC, ex, QR3 \rangle$ where
   - $QR3 = idC, ex, s$.

8. $A \to C$: $ev_{\chi_1}, ev_{\chi_2}, \dots, ev_{\chi_m}$ and $sign2$.

9. C checks ev_{χ_i}, calculates $\beta_{\sigma_j} = \frac{b_{\sigma_j \gamma_j}}{(a_{\sigma_j \gamma_j})^{x_{\sigma_j}}}$ where
   - $i = 1, 2, \dots, m$.
   - $j = 1, 2, \dots, n$.

   and prints `paper` $= \langle (\beta_{\sigma_1}, \beta_{\sigma_2}, \dots, \beta_{\sigma_n}), idC, ex, QR1, QR2 \rangle$ where
   - $QR1 = idC, ex, sign1, com_A, (x_{\sigma_1}, x_{\sigma_2}, \dots, x_{\sigma_n}), (\gamma_{\sigma_1}, \gamma_{\sigma_2}, \dots, \gamma_{\sigma_n})$.
   - $QR2 = idC, ex, sign2, (y_{\sigma_1}, y_{\sigma_2}, \dots, y_{\sigma_n}), (\alpha_{\chi_1}, \alpha_{\chi_2}, \dots, \alpha_{\chi_m}),$ $(\omega_{\sigma_1 1}, \dots, \omega_{\sigma_1 k}), \dots (\omega_{\sigma_n 1}, \dots, \omega_{\sigma_n k})$.

10. $A \xrightarrow{hands} K$: `transp`, `test`

Figure 3: Preparation phase

11. $C \xrightarrow{hands} K$: `id_doc`

12. K checks `id_doc`

13. $K \xrightarrow{hands} C$: `transp`

14. C picks a random `test`, calculates $pid = (\alpha_1, \alpha_2, \dots, \alpha_n) \oplus (\beta_1, \beta_2, \dots, \beta_n)$ and
    if pid is unintelligible then C writes $\texttt{test}_{\texttt{filled}} = (questions, answers, pid)$
    otherwise C runs the Testing Dispute Resolution algorithm.

15. $C \xrightarrow{hands} K$: $\texttt{test}_{\texttt{filled}}$

Figure 4: Testing phase

16. $K \xrightarrow{hands} E$: $\texttt{test}_{\texttt{filled}}$

17. E calculates $sign_E = Sign_{SSK_E}(pid, answers, mark)$ where:
    - $mark \in M$.

18. $E \rightarrow A$: $pid, answers, mark, sign_E$

19. A calculates $c = g^v h^{mark}$ and $sign3 = Sign_{SSK_A}(pid, answers, c)$
    - $v \in_{\mathcal{R}} \mathbb{Z}_q^*$.

20. $A \rightarrow \mathcal{BB}$: $pid, answers, c, sign3$

Figure 5: Marking phase

21. $C \rightarrow A$: $(\beta_1, \beta_2, \dots, \beta_n), pid, sign1, sign2, sign3$

22. A calculates $sign4 = Sign_{SSK_A}(idC, ex, pid, mark, v)$

23. $A \rightarrow C$: $idC, ex, pid, mark, v, sign4$

Figure 6: Notification phase