A Framework for Analyzing Verifiability in Traditional and Electronic Exams

Jannik Dreier[1], Rosario Giustolisi[2], Ali Kassem[3]
Pascal Lafourcade[4], and Gabriele Lenzini[2]

[1] Institute of Information Security, ETH Zurich
[2] SnT/University of Luxembourg
[3] Université Grenoble Alpes, CNRS, VERIMAG, Grenoble, France
[4] Université d'Auvergne, LIMOS

Abstract. The main concern for institutions that organize exams is to detect when students cheat. Actually more frauds are possible and even authorities can be dishonest. If institutions wish to keep exams a trustworthy business, anyone and not only the authorities should be allowed to look into an exam's records and verify the presence or the absence of frauds. In short, exams should be *verifiable*. However, what verifiability means for exams is unclear and no tool to analyze an exam's verifiability is available. In this paper we address both issues: we formalize several *individual* and *universal verifiability properties* for traditional and electronic exams, so proposing a set of verifiability properties and clarifying their meaning, then we implement our framework in ProVerif, so making it a tool to analyze exam verifiability. We validate our framework by analyzing the verifiability of two existing exam systems – an electronic and a paper-and-pencil system.

1 Introduction

Not a long time ago, the only way for a student to take an exam was by sitting in a classroom with other students. Today, students can take exams using computers in test centers or even from home and can be graded remotely. This change is possible thanks to computer-aided or computer-based exams, generally called electronic exams (e-exams). E-exams are integrated in Massive Open Online Courses (MOOC), platforms to open a worldwide access to university lectures. E-exams are also trialled at university exams: at the University Joseph Fourier, exams in pharmacy have been organized electronically in 2014 using tablet computers, and all French medicine exams are planned to be managed electronically by 2016 [13].

All such diverse exam and e-exam protocols should provide a comparable guarantee of security and not only against students that cheat, the main concern of exam authorities, but also against other frauds and the frauds perpetrated by the authorities themselves. More or less effective mitigations exist but to really address the matter exams must be *verifiable*. Verifiable exams can be checked for the presence or the absence of irregularities and provide evidence about the fairness and the correctness of their grading procedures. And they should be welcome by authorities since exam verifiability is also about to be transparent about an exam's being compliance with regulations as well as being able to inspire public trust. Specially, some recent scandals [14, 17] show that frauds do not come only from students, but also from exam authorities.

Ensuring verifiability is generally hard and for exams and e-exams one part of the problem lays in the lack of clarity about what verifiability properties they should offer. Another part comes from the absence of a framework to check exams for verifiability. This paper proposes a solution for both.

Contributions. We provide a clear understanding of verifiability for exam protocols and propose a methodology to analyze their verifiability: we define a formal framework where we model traditional paper-and-pencil and electronic exams. We formalize eleven verifiability properties relevant for exams and, for each property, we state the conditions that a sound and complete verifiability test has to satisfy. Following a practice already explored in other domains [2,3,7,20], we classify our verifiability properties into individual and universal, and we formalize them within our framework. Finally, we implement the verifiability tests in the applied π-calculus and we use ProVerif [5] to run an automated analysis. We validate the effectiveness and the flexibility of our framework by modelling and analyzing two different exam protocols: a paper-and-pencil exam currently used by the University of Grenoble, and an internet-based exam protocol called Remark! [16]. We check whether they admit sound and complete verifiabile tests and discuss what exam roles are required to be honest.

Outline. The next section comments the related work. Section 3 provides definitions and models for exam protocols. Section 4 describes and formalizes eleven verifiability properties, and develops a framework of analysis for them. Section 5 validates the framework. Section 6 draws the conclusions and outlines the future work.

2 Related Work

To the best of our knowledge, there is almost no research done on verifiability for exams. A handful number of papers list informally a few security properties for e-exams [6, 15, 21]. Only one offers a formalization [11]. We comment them shortly.

Castella-Roca *et al.* [6] discuss a secure exam management system which is claimed to provide authentication, privacy, correction and receipt fullness, properties that are described informally. Huszti & Pethő [21] refine these notions as security requirements and propose a cryptographic protocol that is claimed to fulfil the requirements, but the claims are only sustained informally. Bella *et al.* [15] comment a list of requirements which are desirable for electronic and traditional exams, and similar to the previous works, they do not formalize the properties they propose. Instead, Dreier *et al.* [11] propose a model for several authentication and secrecy properties in the formal framework of the Applied π-Calculus [1]. No paper outlined above addresses verifiability.

However, verifiability has been studied in other domains than exams, specially in voting and in auctions. In these domains formal models and definitions of security properties exist stably [12, 22, 23]. In voting, *individual verifiability* ensures that a voter can verify her vote has been handled correctly, that is, cast as intended, recorded as cast, and counted as recorded [3, 20]. The concept of *universal verifiability* has been introduced to express that voters and non-voters can verify the correctness of the tally using only public information [2, 3, 7]. Kremer *et al.* [22] formalize both individual and universal verifiability in the Applied π-Calculus [1]. They also consider *eligibility verifiability*, a specific universal property assuring that any observer can verify that the set of votes from which the result is determined originates only from eligible voters, and that each eligible voter has cast at most one vote. Smyth *et al.* [25] use ProVerif to check different verifiability notions that they express as reachability properties, which ProVerif processes natively. In this paper, we also use ProVerif for the analysis, but the model and the definitions here proposed are more general and constrained neither to the Applied π-Calculus nor to ProVerif.

Verifiability for e-auction is studied in Dreier *et al.* [12]. The manner in which they express sound and complete tests for their verifiability properties has been a source of inspiration for what we present here.

Notable notions related to verifiability are *accountability* and *auditability*. Küsters *et al.* [23] study accountability ensuring that, when verifiability fails, one can identify the participant responsible for the failure. They also give symbolic and computational definitions of verifiability, which they recognize as a weaker variant of accountability. However, their framework needs to be instantiated for each application by identifying relevant verifiability goals. Guts *et al.* [18] define auditability as the quality of a protocol that stores sufficient evidence to convince an honest judge that specific properties are satisfied. Auditability revisits the universal verifiability defined in this paper: anyone, even an outsider without knowledge of the protocol execution, can verify the system relying only on the available pieces of evidence.

3 Exam Model

Any exam, paper-and-pencil or electronic, involves at least two roles: the *candidate* and the *exam authority*. The exam authority can have several sub-roles: the *registrar* registers candidates; the *question committee* prepares the questions; the *invigilator* supervises the exam, collects the answers, and dispatches them for marking; the *examiner* corrects the answers and marks them; the *notification committee* delivers the marking.

Exams run generally in phases, commonly four of them: *Registration*, where the exam is set up and candidates enroll; *Examination*, where candidates answer the questions, give them to the authority, and have them accepted officially; *Marking*, where the exam-tests are marked; and *Notification*, where the grades are notified. Usually, each phase ends before the next one begins, an assumption we embrace.

Assuming such roles and such phases, our model of exam consists of four sets — a set of candidates, a set of questions, a set of answers (questions and answers together are called *exam-tests*) and a set of marks. Three relations link candidates, exam-tests, and marks along the four phases: `Accepted`, `Marked`, and `Assigned`. They are assumed to be recorded during the exam or build from data logs such as registers or repositories.

Definition 1 (Exam). *An exam E is a tuple (I, Q, A, M, α) where I of type $\mathcal{I}$ is a set of candidate identities, Q of type $\mathcal{Q}$ is a set of questions, A of type $\mathcal{A}$ is a set of answers, M of type $\mathcal{M}$ is a set of marks, and α is the set of the following relations:*

- $\texttt{Accepted} \subseteq I \times (Q \times A)$*: the candidates' exam-tests accepted by the authority;*
- $\texttt{Marked} \subseteq I \times (Q \times A) \times M$*: the marks delivered on the exam-tests;*

- `Assigned` $\subseteq I \times M$*: the marks assigned (*i.e., *officially linked) to the candidates;*
- `Correct` : $(\mathcal{Q} \times \mathcal{A}) \to \mathcal{M}$*: the function used to mark an exam-test;*

Definition 1 is simple but expressive. It can model electronic as well as paper-and-pencil exams, and exams executed honestly as well as exams with frauds. It is the goal of verifiability to test for the absence of anomalies. For this aim we recognize two specific subsets: (a) $I_r \subseteq I$ as the set of candidates who registered for the exam (thus, $I \setminus I_r$ are the identities of the unregistered candidates who have taken the exam), and (b) $Q_g \subseteq Q$ as the questions that the question committee has prepared (thus, $Q \setminus Q_g$ are the additional and illegitimate questions that appear in the exam).

The function `Correct` models any objective mapping that assigns a mark to an answer. This works well for single-choice and with multiple-choice questions, but it is inappropriate for long open questions. Marking an open question is hardly objective: the ambiguities of natural language can lead to subjective interpretations by the examiner. Thus, independently of the model, we cannot hope to verify the marking in such a context. Since in our framework the function `Correct` is used to verify the correctness of the marking, exams that do not allow a definition of such a function cannot be checked for that property; however, all other properties can still be checked.

4 Verifiability Properties

To be verifiable with respect to specific properties, an exam protocol needs to provide tests to verify these properties. A test t is a function from $\mathcal{E} \to$ `bool`, where $\mathcal{E}$ is the set of data used to run the test. Abstractly, a verifiable property has the format $t(e) \Leftrightarrow c$, where t is a test, e is the data used, and c is a predicate that expresses the property the test is expected to check. Direction $\Rightarrow$ says that the test's success is a sufficient condition for c to hold (soundness); direction $\Leftarrow$ says that the test's success is a necessary condition for c to hold (completeness).

Definition 2. *An exam for which it exists a test for a property is* testable *for that property. An exam is* verifiable *for that property when it is testable and when the test is* sound *and* complete*.*

To work, a test needs pieces of data from the exam's execution. A verifier, which is the entity who runs the test, may complementarity use personal knowledge about the exam's run if he has any. We assume data to be taken after the exam has ended, that is, when they are stable and not subject to further changes.

To be useful, tests have to be sound even in the presence of an attacker or of dishonest participants: this ensures that when the test succeeds the property holds despite any attempt by the attacker or the participants to falsify it. However, many sound tests are not complete in such conditions: a misbehaving participant can submit incorrect data and, in so doing, causing the test to fail although the property holds. Unless said differently, we check for soundness in presence of some dishonest participants (indeed we seek for the maximal set of dishonest participants that preserve the soundness of the test), but we check for completeness only with honest participants.

A verifiability test can be run by the exam participants or by outsiders. This brings to two distinct notions of verifiability properties: *individual* and *universal*. In exams, individual verifiability means verifiability from the point of view of a candidate. She can feed the test with the knowledge she has about the exam, namely her personal data (identity, exam-test, mark) and the messages she exchanged with the other participants during the exam. Universal verifiability means verifiability from the point of view of an external observer. In practical applications this might be an auditor who has no knowledge of the exam: he has no candidate ID, he has not seen the exam's questions and answered any of them, and he did not receive any mark. Besides, he has not interacted with any of the exam participants. In short, he runs the test only using the exam's public pieces of data available to him.

In Table 1 we select six individual (left) and five universal (right) relevant verifiability properties. These properties cover the verifiability of all phases of a typical exam. We define one property about registration verifiability, one about the validity of questions, two about the integrity of exam-test, two about the process of marking, and one about the integrity of notification. More details are given in the reminder of the section.

Generally speaking, an exam is fully (individual or universal) verifiable when it satisfies all the properties. Of course, on an exam each property can be verified separately to clearly assess its strengths and weaknesses.

Individual Verifiability Properties (I.V.): Here is the candidate that verifies the exam. She knows her identity i, her submitted exam-test q and a, and her mark m. She also knows her perspective p of the exam run, that is, the messages she has sent and received. Her data is a tuple (i, q, a, m, p). Note that the candidate's perspective p is not necessary to define the properties, that's why it does not appear in the right-hand-side of the equivalent (see Table 1). However, it might be necessary to implement the test depending on the case study.

	Individual Verifiability	**Universal Verifiability**
Registration		$\mathtt{R}_{\mathtt{UV}}(e) \Leftrightarrow$ $I_r \supseteq \{i : (i, x) \in \mathtt{Accepted}\}$
Question Validity	$\mathtt{QV}_{\mathtt{IV}}(i, q, a, m, p) \Leftrightarrow$ $(q \in Q_g)$	
Marking Correctness	$\mathtt{MC}_{\mathtt{IV}}(i, q, a, m, p) \Leftrightarrow$ $(\mathtt{Correct}(q, a) = m)$	$\mathtt{MC}_{\mathtt{UV}}(e) \Leftrightarrow$ $(\forall (i, x, m) \in \mathtt{Marked},$ $\mathtt{Correct}(x) = m$
Exam-Test Integrity	$\mathtt{ETI}_{\mathtt{IV}}(i, q, a, m, p) \Leftrightarrow$ $\big((i, (q, a)) \in \mathtt{Accepted}$ $\wedge \exists m' : (i, (q, a), m') \in \mathtt{Marked}\big)$	$\mathtt{ETI}_{\mathtt{UV}}(e) \Leftrightarrow$ $\mathtt{Accepted} =$ $\{(i, x) : (i, x, m) \in \mathtt{Marked}\}$
Exam-Test Markedness	$\mathtt{ETM}_{\mathtt{IV}}(i, q, a, m, p) \Leftrightarrow$ $(\exists m' : (i, (q, a), m') \in \mathtt{Marked}))$	$\mathtt{ETM}_{\mathtt{UV}}(e) \Leftrightarrow$ $\mathtt{Accepted} \subseteq$ $\{(i, x) : (i, x, m) \in \mathtt{Marked}\}$
Marking Integrity	$\mathtt{MI}_{\mathtt{IV}}(i, q, a, m, p) \Leftrightarrow$ $\exists m' : \big((i, (q, a), m') \in \mathtt{Marked}$ $\wedge (i, m') \in \mathtt{Assigned}\big)$	$\mathtt{MI}_{\mathtt{UV}}(e) \Leftrightarrow$ $\mathtt{Assigned} =$ $\{(i, m) : (i, x, m) \in \mathtt{Marked}\}$
Marking Notification Integrity	$\mathtt{MNI}_{\mathtt{IV}}(i, q, a, m, p) \Leftrightarrow$ $(i, m) \in \mathtt{Assigned}$	

Table 1. Individual and Universal Verifiability

There is no individual verifiability property about registration as a candidate knows whether she has registered, and she might even have a receipt of it. Instead, what a candidate does not know, but wishes to verify, is whether she got the correct questions, and whether she got her test correctly marked. To verify the validity of her question, we propose the property *Question Validity* which ensures that the candidate receives questions actually generated by the question committee. This is modeled by a test which returns true, if and only if, the questions q received by the candidate belong to the set of the valid questions Q_g generated by the question committee. To verify that her mark is correct, the candidate can check the property *Marking Correctness* which ensures that the mark received by the candidate is correctly computed on her exam-test. Verifying *Marking Correctness* could e.g. be realized by giving access to the marking algorithm, so the candidate can compute again the mark that corresponds to her exam-test and compare it to the mark she received. As discussed in Section 3, this is feasible with multiple-choice questions or short open-questions, but rather difficult in other cases such as the case of long and open questions. In this case, a candidate may wish to verify more properties about her exam test, and precisely that the integrity of the candidate's exam-test is preserved till marking, and that the integrity of the candidate's mark is preserved from delivery till reception. Preserving the integrity of the exam-test and that of the mark is sufficient for the candidate to be convinced that she got the correct mark, provided the examiner follows the marking algorithm correctly.

Each of the remaining four individual properties covers a different step from exam-test submission till mark reception. This allows to identify in which step the error happened in case of failure. The first property, *Exam-Test Integrity*, is to ensure that the candidate's exam-test is accepted and marked as she submitted it without any modification. Running the *Exam-Test Integrity* test after the end of the exam does not invalidate the property since if an exam-test is lost or modified before being marked, it remains modified also after the exam is over. But the event consisting of an exam-test that is first changed before the marking, and then restored correctly after marking, is not captured by *Exam-Test Integrity*. However, such an event can still be detected by verifying *Marking Correctness*. Another property that also concerns the integrity of the exam-test is *Exam-Test Markedness* which ensures that the exam-test submitted by a candidate is marked without modification. Note that if *Exam-Test Integrity* (*i.e.,* the test $\mathtt{ETI}_{\mathtt{IV}}$) succeeds, then *Exam-Test Markedness* (*i.e.,* the test $\mathtt{ETM}_{\mathtt{IV}}$) also succeeds, namely $\mathtt{ETI}_{\mathtt{IV}}(i, q, a, m, p) \Rightarrow \mathtt{ETM}_{\mathtt{IV}}(i, q, a, m, p)$. However, if the test $\mathtt{ETI}_{\mathtt{IV}}$ fails, but the test $\mathtt{ETM}_{\mathtt{IV}}$ succeeds, this would mean that the candidate's exam-test is modified upon acceptance by the authority, but then restored to its correct version before marking. The latter case could be not relevant to the candidate as her exam-test was unmodified when marked; however such an error can be reported to the responsible authority to investigate the problem and see where the error comes from. Moreover, we might have a protocol that does not provide a test for $\mathtt{ETI}_{\mathtt{IV}}$, but a test for $\mathtt{ETM}_{\mathtt{IV}}$, this could depend on the available data at the end of the exam execution. The remaining two properties ensure that the integrity of the mark attributed to a candidate's exam-test by the examiner is preserved. The property *Mark Integrity* ensures that the mark attributed to a candidate's exam-test is assigned to that candidate by the responsible authority without any modification; and the property *Mark Notification Integrity* ensures that the candidate receives the mark assigned to her by the authority.

Universal Verifiability Properties (U.V.): These properties are designed from the viewpoint of a generic observer. In contrast to the individual viewpoint, the observer does not have an identity and does not know an exam-test or a mark, because he does not have an official exam role. The observer runs the test on the public data available after a protocol run. Hence, we simply have a general variable e containing the data.

In the universal perspective, properties such as Question Validity and Mark Notification Integrity are not relevant because the external observer has no knowledge of the questions nor of the markings received by the candidates. However, an observer may want to verify other properties revealing whether the exam has been carried out correctly, or he may want to check that the exam authorities and examiners have played by the rules. Precisely, an observer would be interested in verifying that only eligible candidates can submit an exam-test, and this is guaranteed by *Registration*, which ensures that all accepted exam-tests are submitted by registered candidates. An observer may wish to test that all the marks attributed by the examiners to the exam-tests are computed correctly. This property, *Marking Correctness*, raises the same practical questions as the individual case and therefore the same discussion applies here. However, even in case of open questions, to increase their trustworthiness, universities should allow auditors to access their log for an inspection to the marking process. It may be also interested in checking that no exam-test is modified, added, or deleted till the end of the marking phase: this *Exam-Test Integrity*, which ensures that all and only accepted exam-tests are marked without any modification. Another property that could be useful for an observer is *Exam-Test Markedness*. This ensures that all the accepted exam-tests are marked without modification. Thus, if *Exam-Test Integrity* fails but *Exam-Test Markedness* succeeds, then there is at least one extra marked exam-test which is not included in the set of accepted exam-test by the exam authority. Finally, the observer may wish to check that all and only the marks assigned to exam-tests are assigned to the corresponding candidates with no modifications. This is guaranteed by *Mark Integrity*.

5 Validation

We validate our framework and show its flexibility with two different use cases: a paper-and-pencil exam procedure and an internet-based exam protocol. We analyze their verifiability fully. The modeling and the analysis is done in ProVerif. For the full treatment of the case studies, we refer the reader to our technical report [10]. The ProVerif code is available on line[5]. We consider the Dolev-Yao [9] intruder model that is used in ProVerif. Dishonest roles, when needed, are processes controlled by the intruder.

5.1 Use Case # 1: The Grenoble Exam

The first exam that we analyze is the paper-and-pencil procedure used to evaluate undergraduate students at the University of Grenoble. It involves candidates (C), an examiner (E), a question committee (QC), and an exam authority (EA). It has four phases:

[5] `apsia.uni.lu/stast/codes/exams/proverif_ispec15.tar.gz`

Registration: All the students of the course are automatically registered as candidates for the exam; they are informed about the exam's date, time and location. EA assigns a fresh pseudonym to each C. The QC, the course's lecturer(s), prepares the questions and hands them to EA.

Examination: After EA authenticates all Cs, EA lets them take a seat. There, each C finds a special exam paper: the top-right corner is glued and can be folded. Each C signs it, and writes down her name and student number in such a way that the corner, when folded, hides them. Each C also writes down visibly their pseudonyms. Then, EA distributes the questions, and the exam begins. At the end, EA collects the exam-tests, checks that all copies have been returned, that all corners are correctly glued, and gives the exam-tests to E.

Marking: E evaluates the exam-tests: each pseudonym is given a mark. E returns them, along with the marks, to EA.

Notification: EA checks that the corner is still glued and maps the pseudonyms to real identities (names and student numbers) without opening the glued part. Then, EA stores the pairs student numbers / marks and publishes them. C can review her exam-test in presence of E to check the integrity of her exam-test and verify the mark. If, for instance, C denies that the exam-test containing her pseudonym belongs to her, the glued part is opened.

Formal Model We model the Grenoble protocol in ProVerif. EA, QC, E and the Cs are modeled as communicating processes that exchange messages over public or private channels. They can behave honestly or dishonestly. We detail later when we need private vs. public channels and honest vs. dishonest participants.

Data sets I, Q, A and M are as in Definition 1. Each set is composed by a selection of messages taken from the data generated by the processes, possibly manipulated by the attacker. For example, Q are all the messages that represent a question. Q_g, subset of Q, are all the messages representing a question that are generated by the QC. The exam's relations are also as in Definition 1. `Accepted` contains all the messages $(i, (q, a))$ (*i.e.,* identity and exam-test) that EA has collected. If the EA is honest, it accepts only the exam-tests submitted by registered candidates. `Marked` contains all the messages $(i, (q, a), m)$ (*i.e.,* identity, exam-test, and mark) that the E has generated after having marked the exam-tests. If E is honest, he marks only exam-tests authenticated by EA. `Assigned` contains all the messages (i, m) originating from the EA when it assigns mark m to candidate i. If EA is honest, it assigns a mark to C only if E notifies that it is the mark delivered on C's exam-test. `Correct` is a deterministic function that outputs a mark for a given exam-test.

We made a few choices when modeling the Grenoble exam's "visual channels". These are face-to-face channels that all the participants use to exchange data (exam-sheets, student pseudonyms, marks). Intrinsically, all such communications are mutually authenticated. To model visual channels in ProVerif, we could have used private channels, but this would have made the channels too strong, preventing the attacker even from knowing if a communication has happened at all. More appropriately, visual channels are authenticated channels, where authentication is expressed by an equational theory similar to the one commonly used for cryptographic signatures, but with the assumption that the verification key is only known to the intended receiver, namely: $\mathtt{openauth}(\mathtt{auth}(m, s)) = m$, and $\mathtt{authcheck}(\mathtt{auth}(m, s), \mathtt{generate}(s)) = m$. Function `auth` takes as input a message m and a secret s that only the sender knows, and outputs an authenticated value. The verification key that corresponds to this secret, $\mathtt{generate}(s)$, is possessed only by the receiver/verifier. Anyone can get the message, m, but only the owner of $\mathtt{generate}(s)$ can verify its origin.

Analysis of Individual Verifiability We model individual verifiability tests as processes in ProVerif, guided by the properties defined in Table 1. Each test emits two events: the event `OK`, when the test succeeds, and the event `KO`, when the test fails. We use correspondence assertions, *i.e., "if an event e is executed the event e' has been previously executed"* [24], to prove soundness, and resort to unreachability of `KO` to prove completeness. We also use unreachability to prove soundness for Marking Correctness.

A sound test receives its input via public channels. This allows an attacker to mess with the test's inputs. Participants can be dishonest too. Thus, we check that the event `OK` is always preceded by the event emitted in the part of the code where the predicate becomes satisfied. Below, we describe how this works for Question Validity.

A complete test receives its input via private channels and by honest participants. The intruder cannot change the test's input this time. Then, we check that the test does not fail, that is, the event `KO` is unreachable.

Figure 1 reports the result of the analysis. All properties hold (✓) despite the intruder, but often they hold only assuming some roles to be honest : there are attacks otherwise. All properties but Marking Correctness have sound tests (Figure 1, middle column) only if we assume at least the honesty of the exam authority (EA), or of the examiner (E), or of both. This in addition to the honesty of candidate, who must be necessarily honest because he is the verifier. The minimal assumptions for all the properties are reported in brackets. All properties have complete tests (Figure, right columns) but all roles except the intruder have to be honest for them to hold.

Property	Sound	Complete
Question Validity	✓(EA)	✓(all)
Exam-Test Integrity	✓(EA, E)	✓(all)
Exam-Test Markedness	✓(E)	✓(all)
Marking Correctness	✓	✓(all)
Mark Integrity	✓(EA, E)	✓(all)
Mark Notification Integrity	✓(EA)	✓(all)

Fig. 1. I.V. properties for the Grenoble exam.

Due to the limited space, we only comment in detail on how we tested one verifiability property: Question Validity. It must be said that, in reality, the Grenoble exam does not provide any means for a candidate to verify question validity. The questions she receives from the EA comes without any proof that they actually were generated by the QC: there is no QC's signature or stamp. But, if we assume an honest EA, a simple test exists: the candidate can authenticate EA when it distributes the questions.

The QV test inputs the verification value `ver_AC`, which is used to authenticate the exam authority. On channel `chTest`, the test inputs the authenticated question `auth_q`, which it checks for origin-authenticity. The test succeeds if the question is authenticated by the EA, it fails otherwise. The test emits the event `OK` when it succeeds, otherwise emits the event `KO`. Namely,

```
let test(chTest, ver_AC) =
 in(chTest, (auth_q)); let question = openauth(auth_q)
 in if authcheck(auth_q, Ver_AC) = question
 then event OK else event KO.
```

In the proof for soundness, we modified the ProVerif code for EA in such way to emit an event `valid` just after the process receives the question from QC and checks its origin-authenticity, and just before EA sends the question to the C. ProVerif shows, in case of honest EA, that any `OK` is preceded by `valid`: the test outputs true only if the question is generated by QC. Note that any tampering that QC can perform on the questions (for example, generating dummy questions or by trashing them after having generated them) does not violate question validity *per se*: according to this property the questions that C received are still those generated, honestly or dishonestly, by the QC: the origin of the question is not compromised.

In the proof for completeness, ProVerif shows that the event `KO` is unreachable. All participants are assumed to be honest in this case.

Analysis of Universal Verifiability Universal verifiable tests should use some public data. But, since the Grenoble exam is a paper-and-pencil based exam, in general, there is no publicly available data. Thus, originally Grenoble exam does not satisfy any of the universal verifiability properties. To be universally testable, an auditor has to be given access to the following data: (1) for Registration verifiability, he can read the list of registered candidates and the set of accepted exam-tests. Thus, he can check whether all accepted exam-tests are submitted by registered candidates; (2) for Exam-Test Markedness, in addition to the accepted exam-tests, he knows the set of marked exam-tests. Then, he can check whether all the accepted exam-tests are marked; (3) for Exam-Test Integrity, he knows the same data as in Exam-Test Markedness. The auditor has to check that all and only the accepted exam-tests are marked; (4) for Marking Correctness, he knows the correction algorithm and the marked exam-tests together with the delivered marks. The test is to run the correction algorithm again on each exam-test and check if the obtained mark is the same as the delivered one; finally, for (5) Mark Integrity, in addition to the delivered marks, he can access the assigned marks. The auditor can check whether the assigned marks are exactly the ones delivered and whether they are assigned to the correct candidates. Having access to such significant data mentioned above could break candidate's privacy (for instance identities, answers, and marks can be disclosed to the auditor); that noticed, discussing the compatibility between the universal verifiability and privacy is not in the scope of this paper.

Similar to what we did for the individual verifiability tests, we use correspondence assertions to prove soundness and unreachability of a `KO` event to prove completeness.

Property	Sound	Complete
Registration	✓(EA)	✓(all)
Exam-Test Integrity	✓(EA, E)	✓(all)
Exam-Test Markedness	✓(EA, E)	✓(all)
Marking Correctness	✓(E)	✓(all)
Mark Integrity	✓(EA, E)	✓(all)

Fig. 2. U.V. properties for the Grenoble exam.

Figure 2 depicts the result of the analysis. We must report that in our testing universal verifiability, not for all tests we were able to run a fully automatically analysis in the general case requiring any number of participants. This is because ProVerif does not support loops and to prove the general case we would have needed to iterate over all candidates. For these tests we ran ProVerif only for the base case, that where we have only one accepted exam-test or one assigned mark; then we completed a manual induction proof that generalizes this result to the general case with an arbitrary number of candidates.

5.2 Use Case #2: Remark!

The second exam system that we analyze is an internet-based cryptographic exam protocol called Remark! [16]. It aims to guarantee several authentication, privacy, and verifiability properties. Remark! has been only proved formally in [11] to ensure authentication and privacy properties, without discussing the protocol's verifiability properties.

Remark! engages the typical exam roles: the candidate (C), the examiner (E), and the exam authority (EA) (called *manager* in the original paper). The protocol uses two particular building blocks: an *exponential mixnet* (NET) [19] and an *append-only bulletin board* (BB) [8]. The protocol's only trust assumption is that there is at least one honest server in NET. We briefly describe the four phases of Remark! below. For the full description we refer to [16].

Registration: The NET generates pseudonyms for C and E using their public keys. The pseudonyms also serve as public encryption and verification keys, and allow C and E to communicate anonymously still guaranteeing some form of authentication. The NET publishes the pseudonyms on the BB. Only C and E can identify their pseudonyms.

Examination: The EA encrypts the question with C's pseudonym, and publishes them on the BB. C retrieves the question, answers it, signs the pair "question and answer" (*i.e.,* the exam-test) using the private key that corresponds to her pseudonym, and submits anonymously the exam-test. Once the EA receives the signed exam-test, it publishes a hashed version of the exam-test on the BB (a *receipt*). The receipt is signed by the EA and can only be verified by C because it is encrypted with C's pseudonym. Then, the EA signs the exam-test and publishes a new post on the BB encrypted with E's pseudonym, which is meant to mark the exam-test.

Marking: E retrieves the exam-test, marks, signs, and sends it back to the EA. The EA encrypts the marked exam-test with the C's pseudonym, and publishes it on the BB.

Notification: When all the marked exam-tests have been posted on the BB, the NET de-anonymizes C's pseudonyms, while E's pseudonyms remain anonymous. To do so, each server of the NET reveals the secret exponents used to generate C's pseudonyms. Finally, the EA can register the mark for the corresponding C.

Formal Model All roles are modelled as communicating processes, except the BB which is a public channel; the equational theory is the following:

$$\begin{aligned}
\texttt{checkpseudo}(\texttt{pseudo\_pub}(\texttt{pk}(k), rce), \texttt{pseudo\_priv}(k, \texttt{exp}(rce))) &= \texttt{true} \\
\texttt{decrypt}(\texttt{encrypt}(m, \texttt{pk}(k), r), k) &= m \\
\texttt{decrypt}(\texttt{encrypt}(m, \texttt{pseudo\_pub}(\texttt{pk}(k), rce), r), \texttt{pseudo\_priv}(k, \texttt{exp}(rce))) &= m \\
\texttt{getmess}(\texttt{sign}(m, k)) &= m \\
\texttt{checksign}(\texttt{sign}(m, k), \texttt{pk}(k)) &= m \\
\texttt{checksign}(\texttt{sign}(m, \texttt{pseudo\_priv}(k, \texttt{exp}(rce))), \texttt{pseudo\_pub}(pk(k), rce)) &= m
\end{aligned}$$

Data sets I, Q, A and M are as in Definition 1. Set I contains the C's pseudonyms rather than their identities. This replacement is sound because any candidate is uniquely identified by her pseudonym and the equational theory preserves this bijection. The sets Q, A, and M are the messages that correspond to questions, answers, and marks generated during the protocol's run. The relations are built from the posts that appear on the BB. Precisely, the tuple $(i, (q, a))$ of `Accepted` is built from the receipts that EA publishes at Examination. The tuples $(i, (q, a), m)$ and (i, m) of `Marked` and `Assigned` respectively consist of the posts that EA publishes at Marking. Precisely, the tuple $(i, (q, a), m)$ is built from the marked exam-test signed by E, while the tuple (i, m) is built from the encryption of the marked exam-test that EA generates. In fact, the encryption requires a pseudonym, and officially links C with their identities. This replacement is sound because C is uniquely identified by her key and the marked exam-test. `Correct` is the algorithm used to mark the exam-tests and is modeled using a table.

Analysis of Individual Verifiability Similarly to what we did in the previous analysis we use assertions to prove soundness and unreachability of the event `KO` to prove completeness. In checking the soundness of a test, we assumed an honest C (the verifier), in addition to the honest NET. The roles of E and co-candidates (*i.e.,* candidates other than the verifier) are dishonest for all tests. The input of a test consists of the data sent via private channel from C, the data sent via public channel from EA, and the data posted on BB. To check the completeness of a test, we model all roles as honest. They send their data via private channel to the test, whose input also includes data posted on BB.

Remark! originally mandates only two individual verifiability properties: Mark Notification Integrity and a weaker version of Exam-Test Integrity. However, we checked which assumptions Remark! needs in order to ensure all our properties. For the sake of space, we only report Question Validity here. Remark! assumes that EA both generates the questions and sends them to the candidates. EA has to be honest to avoid that it sends to C questions that are different from the ones generated.

```
let testQV(pkA,pch, bb)=
 in (bb, eques);
 in (pch, (ques, priv_C:skey));
 let(ques', sques)=decrypt(eques, priv_C)
 in let (ques'', pseudoC)=checksign(sques, pkA)
 in if ques'=ques && ques''=ques' then event OK else event KO.
```

Property	Sound	Complete
Question Validity	✓ (EA)	✓(all)
Exam-Test Integrity	✓	✓(all)
Exam-Test Markedness	✓	✓(all)
Marking Correctness	✓ (EA)	✓(all)
Mark Integrity	✓	✓(all)
Mark Notification Integrity	✓	✓(all)

Fig. 3. I.V. properties for Remark!

The test receives the question published on the BB (`eques`), C's question and her private key (`ques, priv_C`) on a private channel. The test checks whether C actually received the question published on BB from EA. To model soundness in ProVerif, we insert the event `generated(ques)` where the EA process generates the question, and the event `accepted(ques)` into the test process, exactly inside the `if` branch where the test succeeds. The test is sound if each occurrence of the event `accepted(ques)` is preceded by the event `generated(ques)`. ProVerif confirms the test is sound and complete, so Remark! is question validity verifiable.

Figure 3 summarizes the result of our analysis. In the column about the soundness of the verifiability properties the minimal trust requirements to ensure the properties are in brackets. The honesty of the verifier, that is, the candidate, is assumed implicitly. In the column reporting the completeness results all roles are assumed to be honest.

Analysis of Universal verifiability We verify most universal verifiability tests using a different approach compared to the individual ones. This is needed because C can be dishonest, in contrast to the case of individual verifiability, thus no sufficient events can be insert in any process to model correspondence assertions. In general, the idea of this approach is that every time the test succeeds, which means that it emits the event `OK`, we check if the decryption of the concerned ciphertext gives the expected plaintext. If not, the event `KO` is emitted, and we check soundness of the tests using unreachability of the event `KO`. We can still model soundness using correspondence assertions for Registration, because the NET is honestly emitting events when registration concludes.

Since all the bulletin board posts are encrypted with C's or E's pseudonyms, no public data can be used as it is. Moreover, the encryption algorithm is a probabilistic encryption, thus the random value used to encrypt a message is usually needed. So, like Grenoble exam, Remark! does not originally provide universal verifiability. Remark! can be universally testable if EA gives an auditor access to some data after the exam concludes. Again, this might affect candidate's privacy. Here we assume the auditor is given the following data: (1) for Registration, the EA reveals the signatures inside the receipts posted on BB and the random values used to encrypt the receipts. By looking at the bulletin board, the auditor can check that EA only accepted tests signed with pseudonyms posted by the NET during registration; (2) for Exam-Test Integrity, the EA reveals the marked exam-test and the random values used to encrypt them in addition to the data given for Registration. In so doing, the auditor can check if pseudonyms, questions, and answers access the same data outlined above for Exam-Test Integrity. However, since Remark! is exam-test integrity universally verifiable, it is easy to show that the protocol is exam-test markedness universally verifiable too; (4) for Marking Correctness, the EA reveals the marked exam-test, the random values used to encrypt the marked exam-test, and a table that maps a mark to each answer, after the exam concludes. The auditor can thus check if the mark of each exam-test corresponds to the mark of the given table provided the answer; finally, for (5) Mark Integrity, the EA reveals the examiners' signatures on the marked exam-test and the random values that EA used to encrypt them before posting on the BB. In so doing, the auditor can check that each mark notified to the candidate has a correct signature.

Property	Sound	Complete
Registration	✓	✓(all)
Exam-Test Integrity	✓	✓(all)
Exam-Test Markedness	✓	✓(all)
Marking Correctness	✓ (EA)	✓(all)
Mark Integrity	✓	✓(all)

Fig. 4. U.V. properties for Remark!

Figure 4 summarizes the results of our analysis. Marking Correctness is sound only if EA is honest. ProVerif shows an attack if EA is dishonest because it can change the table that the auditor uses to check the correctness of the marks.

Similar to the analysis outlined for the Grenoble exam, ProVerif is unable to handle the general case for the universal verifiability properties also for Remark!. We thus prove in ProVerif the case with one candidate, and also rely on manual induction proofs for the general case.

6 Conclusion

This paper studies verifiability for exam protocols, a security feature that has been studied for voting and auctions but not for exams. In this domain, verifiability properties revealed to be peculiar and required novel definitions. We defined several properties which we organized in individual and universal verifiability properties; moreover, we developed a formal framework to analyze them. As far as we know, we are the first to have developed a framework for verifiability of exam protocols.

Our properties and our methodology work for both to electronic (*e.g.,* cryptographic) exam protocols and to traditional (*e.g.,* paper-and-pencil) exams. Thus, in addition to cryptographic protocols, our methodology is applicable to systems that handle physical security measures, such as a face-to-face authentication. In ProVerif, where we implement our framework, such communications are tricky and we developed ad-hoc equational theories to capture their properties.

We have validated our framework by analyzing the verifiability of two existing exams, one paper-and-pencil-based and the other Internet-based. We run most of the tests automatically; but where ProVerif could not handle the verification of the general case for some universal verifiability properties, we proved manually the tests by induction.

The paper-and-pencil exam has been proved to satisfy all the verifiability properties under the assumption that authorities and examiner are honest. This seems to be peculiar to paper-and-pencil exams, where log-books and registers are managed by the authorities that can tamper with them. Only *Marking Correctness* holds even in presence of dishonest authorities and examiner: here, a candidate can consult her exam-test after marking, thus verifying herself whether her mark has been computed correctly.

The result of the analysis of the Internet-based protocol are somehow complementary. All properties but three are sound without assuming that the exam's roles are honest. But *Marking Correctness*, which worked without assumption in the paper-and-pencil case, holds only assuming an honest exam authority. In fact, a student can check her mark by using the exam table, but this is posted on the bulletin board by the exam authority who can nullify the verification of correctness by tampering with the table.

Since the interest in exam protocols is growing, as future work we plan to analyze more protocols and corroborate more extensively the validity of our framework. Another future work regards the use of tools. Since ProVerif's equational theories, used to model cryptographic primitives, introduce an abstraction which may bring the analyst to miss attacks, we intend to trail CryptoVerif [4] and achieve stronger proofs. In a framework extended with the new tool, we plan to show how to set up an analysis based on the computational model.

References

1. M. Abadi and C. Fournet. Mobile values, new names, and secure communication. In *POPL'01*, pages 104–115, New York, 2001. ACM.
2. J. Benaloh. *Verifiable Secret-Ballot Elections*. PhD thesis, Yale University, December 1996.
3. J. Benaloh and D. Tuinstra. Receipt-free secret-ballot elections (extended abstract). In *STOC '94*, pages 544–553, New York, NY, USA, 1994. ACM.
4. B. Blanchet. A Computationally Sound Mechanized Prover for Security Protocols. In *Proc. of the IEEE Symposium on Security and Privacy, Oakland, May 2006*, pages 140–154.
5. B. Blanchet. An Efficient Cryptographic Protocol Verifier Based on Prolog Rules. In *CSFW*, pages 82–96, Cape Breton, Canada, June 2001. IEEE Computer Society.
6. J. Castellà-Roca, J. Herrera-Joancomartí, and A. Dorca-Josa. A Secure E-Exam Management System. In *ARES*, pages 864–871. IEEE Computer Society, 2006.
7. J. Cohen and M. Fischer. A robust and verifiable cryptographically secure election scheme (extended abstract). In *FOCS'85*, pages 372–382. IEEE Computer Society, October 1985.
8. C. Culnane and S. Schneider. A peered bulletin board for robust use in verifiable voting systems. In *CSF*, 2014.
9. D. Dolev and Andrew C. Yao. On the security of public key protocols. *Information Theory, IEEE Transactions on*, 29(2):198–208, 1983.
10. J. Dreier, R. Giustolisi, A. Kassem, P. Lafourcade, and G. Lenzini. On the verifiability of (electronic) exams. Technical Report TR-2014-2, Verimag, April 2014.
11. J. Dreier, R. Giustolisi, A. Kassem, P. Lafourcade, G. Lenzini, and P. Y. A. Ryan. Formal analysis of electronic exams. In *SECRYPT'14*. SciTePress, 2014.
12. J. Dreier, Jonker H., and P. Lafourcade. Defining verifiability in e-auction protocols. In *ASIACCS'13*, pages 547–552. ACM, 2013.
13. Le Figaro. Etudiants: les examens sur tablettes numériques appellés a se multiplier. Press release, January 2015. Available at `goo.gl/ahxQJD`.
14. Elizabeth Flock. Aps (atlanta public schools) embroiled in cheating scandal. Retrieved from `goo.gl/fqzBBR`, November 2011.
15. R. Giustolisi, G. Lenzini, and G. Bella. What security for electronic exams? In *CRiSIS'13*, pages 1–5. IEEE, 2013.
16. R. Giustolisi, G. Lenzini, and P.Y.A. Ryan. Remark!: A secure protocol for remote exams. In *Security Protocols XXII*, volume 8809 of *LNCS*, pages 38–48. Springer, 2014.
17. François Guénard. *La Fabrique des Tricheurs: La fraude aux examens expliquée au ministre, aux parents et aux professeurs*. Jean-Claude Gawsewitch, 2012.
18. N. Guts, C. Fournet, and F. Zappa Nardelli. Reliable evidence: Auditability by typing. In *ESORICS'09*, volume 5789 of *LNCS*, pages 168–183. Springer, 2009.
19. R. Haenni and O. Spycher. Secure internet voting on limited devices with anonymized dsa public keys. In *EVT/WOTE'11*. USENIX, 2011.
20. M. Hirt and K. Sako. Efficient receipt-free voting based on homomorphic encryption. In *EUROCRYPT'00*, volume 1807 of *LNCS*, pages 539–556. Springer, 2000.
21. A. Huszti and A. Pethõ. A secure electronic exam system. *Publicationes Mathematicae Debrecen*, 77:299–312, 2010.
22. S. Kremer, M. Ryan, and B. Smyth. Election verifiability in electronic voting protocols. In *ESORICS'10*, volume 6345 of *LNCS*, pages 389–404. Springer, 2010.
23. R. Küsters, T. Truderung, and A. Vogt. Accountability: definition and relationship to verifiability. In *CCS'10*, pages 526–535. ACM, 2010.
24. P. Y. A. Ryan, S. A. Schneider, M. Goldsmith, G. Lowe, and A. W. Roscoe:. *The Modelling and Analysis of Security Protocols: The CSP Approach*. Addison-Wesley Professional, 2000.
25. B. Smyth, M. Ryan, S. Kremer, and K. Mounira. Towards automatic analysis of election verifiability properties. In *ARSPA-WITS'10*, volume 6186 of *LNCS*. Springer, 2010.