

Privacy-Preserving Verifiability

A Case for an Electronic Exam Protocol

Rosario Giustolisi¹, Vincenzo Iovino² and Gabriele Lenzini²

¹*IT University of Copenhagen, Copenhagen, Denmark*

²*University of Luxembourg, Interdisciplinary Centre for Security, Reliability and Trust (SnT), Luxembourg, Luxembourg*

Keywords: Privacy-Preserving Verifiability, Electronic Exam Protocols, Analysis of Security and Privacy.

Abstract: We introduce the notion of *privacy-preserving verifiability* for security protocols. It holds when a protocol admits a verifiability test that does not reveal, to the verifier that runs it, more pieces of information about the protocol's execution than those required to run the test. Our definition of privacy-preserving verifiability is general and applies to cryptographic protocols as well as to human security protocols. In this paper we exemplify it in the domain of e-exams. We prove that the notion is meaningful by studying an existing exam protocol that is verifiable but whose verifiability tests are not privacy-preserving. We prove that the notion is applicable: we review the protocol using functional encryption so that it admits a verifiability test that preserves privacy according to our definition. We analyse, in ProVerif, that the verifiability holds despite malicious parties and that the new protocol maintains all the security properties of the original protocol, so proving that our privacy-preserving verifiability can be achieved starting from existing security.

1 INTRODUCTION

“Being verifiable” is an appreciated quality for a protocol. It says that during its execution the protocol safely stores pieces of information that can be used later as evidence to determine that certain properties have been preserved by the protocol's run. A voting system that offers its voters to “verify that their votes have been cast as intended”, for instance, generates data that allow a voter to test that his/her ballot has been registered and that it contains, with no mistake, the expression of his/her vote (Adida and Neff, 2006).

This paper is about verifiability, but it also discusses a new requirement for it. The requirement states that a curious verifier, in addition to the verifiability test's result, should learn no more about the system's execution than the pieces of information that he needs to know to run the test.

The need-to-know principle is not new (e.g., see (Department of Defence, 1987)). Not new is also the particular interpretation of the principle that link to techniques like zero knowledge proofs (De Santis et al., 1988) where a verifier is convinced of a given statement's truth without learning no more than that. However, in this paper we originally present and discuss the principle as *a requirement for verifiability*.

Our notion of *privacy-preserving verifiability*

needs to be formally defined, and we do so in §3. We contextualize the definition to the particular domain of *electronic exams*, where verifiability and privacy are sensitive properties (the first should work despite untrusted authorities, the second is not everlasting but conditional). Besides, in §4, we prove that our notion of privacy-preserving verifiability is meaningful. We show for a particular protocol called Remark! (Giustolisi et al., 2014) that several of its current universal verifiability are not privacy-preserving according to our definition.

In §5, we give evidence that achieving privacy-preserving verifiability is possible. In particular, again taking Remark! as use case, we demonstrate that by modifying the protocol's design it becomes possible to have, for one of the protocol's verifiability property, a test that is privacy-preserving. The modification we operate on the protocol is guided straightforwardly by the predicate that the verifiability test is expected to check. The other verifiability properties of Remark! rely on predicates which are, in their structure, similar to that we prove our case (i.e., they are all predicate about integrity and authentication); thus, we can easily, we claim, achieve privacy-preserving verifiability for them too.

In §6, we analyse formally the new protocol. We verify with Proverif (Blanchet, 2014) that the new de-

sign does not compromise any of the protocol's enjoyed security including all its other verifiability properties. This analysis is a significant step since we prove that privacy-preserving verifiability for a specific verifiability test can be achieved while preserving all the existing security properties of the protocol.

Before moving to the technical part of the paper it is important to anticipate that private-preserving verifiability can be potentially achieved using many of the instruments that computer security research offers today for minimize the disclosure of information. Non-interactive Zero-knowledge proofs (De Santis et al., 1988) and functional encryption schemes (Boneh et al., 2011) are the two instruments that stand out for this purpose. Here, we use the second instrument. We do not have a formal proof but we discuss in §7 that only Functional Encryption has features that make it suitable to our purpose.

2 RELATED WORK

Privacy and verifiability, apparently two contrasting properties, have been discussed in the literature mainly to show that they can be both satisfied in the same system.

Most of the works discussing privacy and verifiability in this sense are about voting systems. The literature on the subject is vast but, to make our point, we comment the work of Cuvelier *et al.* (Cuvelier et al., 2013). The authors propose a new primitive, called *Commitment Consistent Encryption*. With it, they redesign a voting scheme for an election that is verifiable while ensuring everlasting ballot privacy. The problem they solve for voting is clearly related to that we raise here: avoid that from a public audit trail a verifier could learn more than it is allowed to learn. To a certain extent, Cuvelier *et al.* propose a cryptographic instance of the problem that we formalize in this paper. But our notion of privacy-preserving verifiability is more general and not necessarily cryptographic: it applies to cryptographic protocols as well as to non-crypto protocols. It is, in other words, a functional requirement for the verifiability tests that a system offers to auditors. In this sense, our work relates more to the plentiful research that discusses requirements and definitions of privacy. Again, the literature on this subject is huge, but the work that most closely represents the notion of privacy-preservation that we advance here is that of Mödersheim *et al.* (Mödersheim et al., 2013). The work presents and discusses α - β privacy where α is what an adversary knows, and β are the cryptographic messages that the adversary sees. Then α - β privacy means that the

intruder can derive from β only what he can derive from α already. We have not tried it, but we think that our notion of privacy-preservation can be formulated in term of α - β privacy. If that were true, since Mödersheim *et al.* prove that their definition subsumes static equivalence of frames and is in some part decidable with existing formal method tools, we would gain further ways to prove our notion formally.

This paper also relates with the research on verifiability. This is a property that has been studied prevalently, but not exclusively, in secure voting, where different models and requirements have been proposed (Küsters et al., 2010; Kremer et al., 2010). For instance, *individual verifiability* is introduced as the ability that voters have to prove that their votes have been “cast as intended”, “recorded as cast”, and “counted as recorded” (Benaloh and Tuinstra, 1994; Hirt and Sako, 2000), while *universal verifiability* as the ability to verify the correctness of the tally using only public information (Cohen and Fischer, 1985; Benaloh and Tuinstra, 1994; Benaloh, 1996). Tools have been proposed to analyse the security of verifiability: Kremer *et al.* (Kremer et al., 2010) formalise individual and universal verifiability in the applied pi-calculus, and Smyth *et al.* (Smyth et al., 2010) use ProVerif to check verifiability in three voting protocols.

We also base our definition of verifiability on previous work. The formalization we propose is based on the model proposed in (Dreier et al., 2015) This choice is not a coincidence. That work is about verifiable e-exams, the same domain that we take here as a use case to demonstrate that our notion of privacy-preserving verifiability is meaningful and effective. Besides, that work proves that Remark!, an exam protocol rich in security properties (Giustolisi et al., 2014), is verifiable. Such a proof is fundamental for this paper, since we need to consider a protocol that is verifiable although we need it be not privacy-preserving. Thus, as use case, we also chose Remark!.

3 MODELS & DEFINITIONS

We refer to the specific domain of exams. We consider three fundamental roles: the *candidate*, who takes the test; the *exam authority*, who helps run the exam; the *examiner*, who grades the tests.

An exam's workflow is rigidly structured in distinct sequential phases: *Preparation*, where an exam's instance is created, the questions are selected, and the candidates enrol; *Examination*, where the candidates are identified and checked for being eligible, receive the test sheet(s), answer the questions, and quit

or submit their answers for marking; *Marking*, where the exam-tests are assessed and marked; *Notification*, where grades are notified and registered.

An Exam's Abstract Model. An exam and its execution can be modelled concisely using sets. The definitions below, slightly simplified, comes from (Dreier et al., 2015)

Definition 1. An exam is a tuple (ID, Q, A, M) of finite sets: ID are the candidates; Q are all the possible questions; A the possible answers; M all the possible marks.

Definition 2. Given an exam (ID, Q, A, M) , an exam's execution, \mathcal{E} , is a tuple of two sets

- $ID' \subseteq ID$ are the registered candidates;
- $Q' \subseteq Q$ are the official questions of the exam;

and of four relations over the exam

- $\text{AssignedQuestions} \subseteq (ID \times Q)$, links a question to a candidate, supposedly the question the candidate receives by the authority;
- $\text{SubmittedTests} \subseteq (ID \times (Q \times A))$, links an answered test to a candidate, supposedly the test submitted for marking by the candidate;
- $\text{MarkedTests} \subseteq (ID \times (Q \times A) \times M)$ links a mark to a exam-tests of a candidate, supposedly the grade given to the test;
- $\text{NotifiedMarks} \subseteq (ID \times M)$ links a mark to a candidate, supposedly the mark notified to the candidate;

These four relations model the state of the exam's execution at the end of each of the four phases. They are, in fact, execution logs. There are no requirements between the relations, and this is because the model is meant to capture also executions which are erroneous or corrupted. It may happen that a test assigned to no one is submitted fraudulently (i.e., creation of test) or that a test assigned to a candidate is duplicated and answered in two possible ways, and both copies are submitted (i.e., duplication of a test). It may happen as well as that a submitted test is never marked because it got lost (i.e., loss of a legitimately submitted and answered test).

Authentication and integrity properties are expressed as predicates over \mathcal{E} . Let c, q, a and m range over ID, Q, A and M respectively. An example of property is *Marking Integrity*, which states that no mark has changed from marking to notification, and it is written as follows:

$$\text{NotifiedMarks} \subseteq \{(c, m) : (c, (q, a), m) \in \text{MarkedTests}\}$$

Another property, *Testing Integrity* expresses that all the exams that have been marked are submitted for marking. It is as follows:

$$\{(q, a) : (c, (q, a), m) \in \text{MarkedTests}\} \subseteq \{(q, a) : (c, (q, a)) \in \text{SubmittedTests}\}$$

Properties can be easily combined. For example, *Marking and Test Integrity* is obtained by conjunction from the previous properties:

$$\begin{aligned} \text{NotifiedMarks} \subseteq \\ \{(c, m) : (c, (q, a), m) \in \text{MarkedTests} \quad (1) \\ \text{and } (c, (q, a)) \in \text{SubmittedTests}\} \end{aligned}$$

Definition 2 serves well also to express properties of verifiability of a generic security property p . An exam protocol is p -verifiable if it has a *test* for p , say test_p . This is an algorithm that given any exam's execution \mathcal{E} and some knowledge K (e.g., decryption keys) returns true if and only if p holds on \mathcal{E} . Formally, p -verifiability can be expressed as

$$\forall \mathcal{E}, \text{test}_p(\mathcal{E}, K) \text{ iff } \mathcal{E} \models p \quad (2)$$

where $\mathcal{E} \models p$ means that p holds on \mathcal{E} .

A Refined Exam's Abstract Model. Dreier et al.'s model partially serves the goal of this paper: it is too abstract to capture non-trivial notions of privacy like the one we intend to advance (see Definition 4). We need a formalism able to model encrypted logs, tests which run over them, and knowledge gained from running a test and about the exam's run. It is in the interplay between these elements where our notion of privacy-preserving verifiability will emerge.

Precisely, besides the *abstract model*, \mathcal{E} , which represents an exam's log always in clear-text, we consider a *concrete model*, $\{\mathcal{E}\}$, which models the actual and possibly encrypted logs of an exam's execution. A verifiability test is executed on the concrete model $\{\mathcal{E}\}$.

Definition 3. Let \mathcal{E} and $\{\mathcal{E}\}$ be an abstract and a concrete exam's execution respectively, and let p be a security property over \mathcal{E} . Then, p is verifiable if there exists a test for p , test_p , that given some knowledge K satisfies the following condition:

$$\forall \mathcal{E}, \text{test}_p(\{\mathcal{E}\}, K) \text{ iff } \mathcal{E} \models p \quad (3)$$

Any proof aiming at convincing that p is verifiable needs to clarify how the test that operates on $\{\mathcal{E}\}$ is sound and complete with respect to the validity of p on \mathcal{E} .

The notion of privacy-preserving verifiability is defined in respect to what a verifier, say v , can learn

about \mathcal{E} by running $\text{test}_p(\{\mathcal{E}\}, K)$. We are, however, not interested to capture what v can learn incrementally by running several and cleverly selected tests. Potentially, quite a deal can be deduced by inference attacks, see (Naveed et al., 2015), but studying the leaks due to such attacks is a goal outside the scope of this paper.

Let $[\{\mathcal{E}\}]_p$ be the portion of $\{\mathcal{E}\}$ to which a verifier needs to access to run a test for p . Since $[\{\mathcal{E}\}]_p$ can be encrypted, v does not necessarily learn about the exam's execution \mathcal{E} . It depends on what v can do from knowing $[\{\mathcal{E}\}]_p$ and K . If we assume v be a passive but curious adversary, and if we assume that the learning process works at the symbolic level of messages, the deduction relation \vdash over messages defines the ability to learn. It is the minimal relation that satisfies the following equations:

$$\begin{aligned} m_k, k^{-1} &\vdash m \\ (m, m') &\vdash m \text{ and } m' \end{aligned}$$

In the relation above, we abstract from symmetric/asymmetric encryption: k^{-1} is the key to decrypt, whatever the paradigm of reference.

Definition 4. Let \mathcal{E} be any execution of an exam protocol and test_p a test for verifiability for a property p . The test is privacy-preserving if by running it v does not learn more about the exam execution than he knows from the information he is given to run the test. Formally:

$$\{m : ([\{\mathcal{E}\}]_p \cup K) \vdash m\} \cap \mathcal{E} = ([\{\mathcal{E}\}]_p \cup K) \cap \mathcal{E}$$

Definition 4 says that v should learn about \mathcal{E} no more than what he knows already from the portion of the exam's log and from additional knowledge, i.e., from the pieces of information that he needs to know to run the test.

In Definition 4 we have slightly overloaded the operator \cap . Since \mathcal{E} is a tuple of sets it must be considered applied over each element of the tuple that is: $(A, B) \cap C = C \cap (A, B) = (A \cap C, B \cap C)$. Similarly the interpretation of \subseteq when applied to tuple of sets, must be extended to tuple, that is: $(A, B) \subseteq (C, D)$ iff $A \subseteq C$ and $B \subseteq D$.

Definition 5 (Privacy-Preserving Verifiability). A protocol is privacy-preserving verifiable with respect to a property p , if p is verifiable and admits a verifiability test test_p that is privacy-preserving.

4 MEANINGFULNESS

Claim 1. Our notion of privacy-preserving verifiability is meaningful: it can be used to distinguish verifiable protocols that are privacy-preserving from those which are not.

To prove the claim, we refer to an e-exam protocol whose verifiability (with respect to several properties) has been established formally. The protocol is called Remark! (Giustolisi et al., 2014). The protocol is structured in the four typical phases of an exam. Its message flow is recalled in Figure 1, where C is a candidate and E is an examiner. Only one candidate and one examiner are shown for conciseness, but the protocol is supposed to work for all candidates and all examiners and it is executed by them all. An exam authority, A , helps the process.

Description. The *Preparation* is the phase where pseudonyms are generated to ensure a double blind anonymity in testing and marking. Here, n exponential mixnets, M in Figure, generate the pseudonyms $\overline{\text{Pk}}_C$ for the registered candidate C . The candidate already possesses a pair of public/private keys, $(\text{Pk}_C, \text{Sk}_C)$ and uses the private key to recognize its designated pseudonym among the pseudonyms generated for all the candidates and posted on a public bulletin board (\mathcal{BB} , in Figure). Similarly, the mixnets generate pseudonyms for the examiner. A zero-knowledge proof of the generation is present on the bulletin board as proof of correctness of the generation process.

At *Examination*, the selected questions are signed by the authority and encrypted with the pseudonym of the candidates. The questions are posted on the bulletin board from where the candidates retrieve them. Then they answer the questions. In Figure, *ques* is the question and *ans* is an answer. The candidate C prepares a tuple $(\text{ques}, \text{ans}, \overline{\text{Pk}}_C)$, which he signs with its private key Sk_C . The tuple is then encrypts with the public key of the authority and send to it. The authority decrypts and re-encrypts the message with the candidate pseudonym and publishes the message on the bulletin board.

At *Marking* the authority dispatches the tests to the examiners. In Figure, A signs the answered test and uses the pseudonyms of the examiner, i.e., $\overline{\text{Pk}}_E$, as encryption keys. Hevene re-encrypts with it the answered test and publishes it on the bulletin board. From there, similarly to what the candidates did with the questions, E retrieves the test, marks them (i.e., assigns a mark M), and prepares a tuple with the answered tests signed by the authority and the marking, which E signs further with its secret key Sk_E . Encrypted this with the public key of the authority, E returns this message to the A who, in turn, decrypts the marking and re-encrypts it with the pseudonym of the candidate $\overline{\text{Pk}}_C$.

At *Notification* the candidates' anonymity is revoked and the marks can finally be registered. The

phase starts when the authority publishes all the marks together. On request the mixnets reveal (on an authenticated encrypted channel like one using TLS) the random values used to generate the pseudonyms (in Figure, \bar{r}_m , for C).

Security Properties. Remark! ensures several authentication, anonymity, privacy and verifiability properties. Here, we focus on verifiability, in particular on *universal verifiability*. Universal means that anyone only by using public knowledge and without having participated in the protocol execution can verify that a specific property holds over the execution. The public data is generally that available on the bulletin board; the pieces of additional information are explicitly provided for the test, e.g., by the authority.

Remark! has been proven to be verifiable for five properties (Dreier et al., 2014):

Registration Correctness: all accepted tests are submitted by registered candidates.

Marking Correctness: all the marks attributed by the examiners to the tests are computed correctly.

Test Integrity: all and only accepted tests are marked without any modification

Test Markedness: only the accepted tests are marked without modification

Mark Integrity: all and only the marks associated to the tests are assigned to the corresponding candidates with no modifications

The properties are verifiable only if the exam authority handles to the verifier some data *after* the exam has ended. For instance to verify *Registration Correctness* the manager must reveal the signatures inside the receipts $\{Sign_{SK_A}(H(T_C))\}_{\bar{P}_{K_C}}$ posted on the bulletin board and the random values used to encrypt the receipts. To verify *Marking Correctness*, the manager must reveal the marked exam-tests inside the evaluations $\{Sign_{SK_E, h_E}(M_C)\}_{P_{K_A}}$, the random values used to encrypt the marked tests, and the table *correct_ans* (not described in Figure 1) that defines what is the mark given a question and an answer; to verify *Test Integrity* and *Test Markedness*, the manager must reveal the marked exam-tests inside the evaluations, the random values used to encrypt the marked tests, plus the data disclosed for Registration; finally, to verify *Mark Integrity*, the manager must reveal the examiners' signatures on the marked exam-tests inside the evaluations, and the random values used to encrypt the notifications $\{Sign_{SK_E, h_E}(M_C)\}_{\bar{P}_{K_C}}$ before posting them on the bulletin board.

Discussion. Let us see where Remark!'s verifiability fails to be privacy-preserving.

To verify *Registration Correctness*, Remark! prescribes a test that takes the pseudonyms of the candidates signed by the mixnet, i.e., message $Sign_{SK_M}(\bar{P}_{K_C}, h_C)$, and the receipts of submissions generated by the exam authority i.e., messages $\{Sign_{SK_A}(H(T_C))\}_{\bar{P}_{K_C}}$. The test succeeds if for each pseudonym there is a unique receipt of submission. The verifier does not know any other information about the execution of the exam apart the pseudonyms and the hash version of the tests. Hence, the registration test satisfies Definition 4.

To verify *Marking Correctness*, the protocol requires a test that inputs $Sign_{SK_M}(\bar{P}_{K_C}, h_C)$, the table *correct_ans*, and the mark notifications signed by the examiner and published by the exam authority $\{Sign_{SK_E, h_E}(M_C)\}_{P_{K_A}}$. The test needs also the help of the exam authority that decrypt these last messages, and reveals the $M_C = \langle Sign_{SK_A}(\bar{P}_{K_E}, T_C), mark \rangle$ which reveals the pseudonym of the examiner who marked the answers, the questions, and the mark. This test does not satisfy Definition 4: the verifier learns the questions, the answers, and the link between candidate and examiner pseudonyms.

To verify *Test Integrity*, the protocol needs a test that takes message $Sign_{SK_M}(\bar{P}_{K_C}, h_C)$, the receipts of submissions $\{Sign_{SK_A}(H(T_C))\}_{\bar{P}_{K_C}}$, and the mark notifications generated by the exam authority $\{Sign_{SK_E, h_E}(M_C)\}_{\bar{P}_{K_C}}$. This test reveals to the verifier questions, answers, and the link between candidate and examiner pseudonyms, which are leaked in the mark notifications, so it does not satisfy Definition 4

To verify *Test Markedness* a test needs to input the same data as in test integrity. The same considerations outlined above applies for this test and the test does not satisfy Definition 4.

To verify *Mark Integrity*, Remark! prescribes a test that takes in the mark notifications $\{Sign_{SK_E, h_E}(M_C)\}_{\bar{P}_{K_C}}$ and allows the verifier to check if the marks that the exam authority assigned to the candidates (in Figure, Register $P_{K_C}, mark$) coincide with the marks that the examiner assigned to the candidates tests in $M_C = \langle Sign_{SK_A}(\bar{P}_{K_E}, T_C), mark \rangle$. Once more, the knowledge of questions, answers, and link between candidate and examiner pseudonyms is leaked by M_C , although those pieces of information are not necessary to evaluate mark integrity. Also this test does not respect Definition 4.

Overall, Remark! is privacy-preserving only for Registration Correctness verifiability.

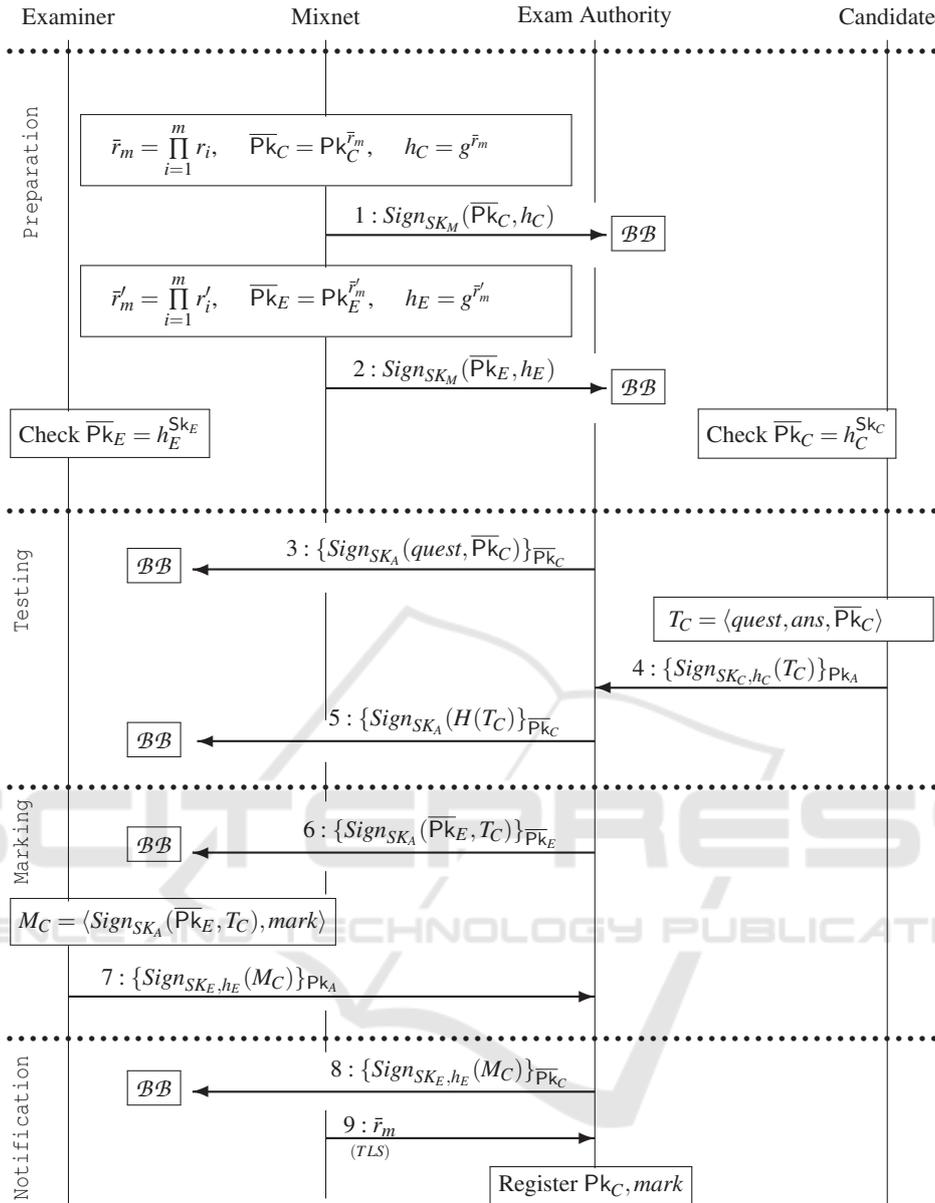


Figure 1: The message flow of Remark!

5 APPLICABILITY

Except for *Registration Correctness*, all the tests for universal verifiability that Remark! offers are not privacy preserving. Here, we show how to make them privacy-preserving. For reason of space we show the technique only that the test about *Mark and Test Integrity* verifiability. We claim, but we leave a proof for future work, that we can apply the same idea of solution to achieve privacy-preserving verifiability for all the other verifiability tests.

To achieve our goal we propose a version of Remark! that relies on *Functional Encryption* (FE). This is an encrypting paradigm that relies on the notion of FE scheme (Boneh et al., 2011).

Definition 6. A FE scheme is a 4-tuple of functions (Setup, KeyGen, Enc, Eval), where:

1. Setup($1^\lambda, 1^n$) returns a pair of keys (Pk, Msk). The first is a public key and the other a secret master key. The keys depends on two parameters: λ , a security parameter and n , a length. They are

polynomially related.

2. $\text{KeyGen}(\text{Msk}, C)$ is a function that returns a token, $\text{Tok}_C^{\text{Msk}}$. A token is a string that when evaluated (see item 4, function Eval) on specific ciphertexts encrypted with Pk returns the same output as the Boolean Circuit C (a decider) would return if it were applied on the corresponding plaintexts. We talk about Boolean Circuits (Jukna, 2012) because the “functionality” in Functional Encryption is defined on this model of computation. Here, $C \in C_n$, where C_n is the class of Boolean Circuit on inputs long n bits.
3. $\text{Enc}(\text{Pk}, m)$ encrypts the plaintext $m \in \{0, 1\}^n$ with the public key Pk .
4. $\text{Eval}(\text{Pk}, \text{Ct}, \text{Tok}_C^{\text{Msk}})$ is the function that processes the token as we anticipated in item 2. On input $\text{Ct} = \text{Enc}(\text{Pk}, m)$ the function returns $C(m)$ that is the same boolean output (accept/reject) as that of circuit C applied on string m . The key Msk embedded in the token is used retrieve the plaintext m from Ct . For any other input than the specified Ct , Eval is undefined and returns \perp .

A FE scheme should be of course instantiated cryptographically, but whatever implementation we chose it must satisfy two specific requirements: *correctness* and *secure indistinguishability*.

The requirement of *correctness* says that the functions in Definition 6 behave as described: for all $(\text{Pk}, \text{Msk}) \leftarrow \text{Setup}(1^\lambda, 1^n)$, and all $C \in C_n$ and $m \in \{0, 1\}^n$, and for $\text{Tok}_C^{\text{Msk}} \leftarrow \text{KeyGen}(\text{Msk}, C)$ and $\text{Ct} \leftarrow \text{Enc}(\text{Pk}, m)$ then $\text{Eval}(\text{Pk}, \text{Ct}, \text{Tok}_C^{\text{Msk}}) = C(m)$. Otherwise it returns \perp .

The requirement of *secure indistinguishability* says that the scheme is “cryptographically secure” against any active and strong enough adversary (i.e., uniform probabilistic polynomial time adversaries). Such an adversary, despite having seen tokens $\text{Tok}_C^{\text{Msk}}$, cannot distinguish (with a non-negligible advantage over pure guessing) two ciphertexts $\text{Enc}(\text{Pk}, m_0)$ and $\text{Enc}(\text{Pk}, m_1)$ for which $C(m_0) = C(m_1)$.

A stronger security requirement used in FE is simulation-security, but due to impossibility results for it (see (Boneh et al., 2011)), we limit our assumption to indistinguishability security.

5.1 Revising the Protocol

We modify Remark! and let it have a test for *Universal Marking and Test Integrity* (UMTI) verifiability that is privacy-preserving according to our Definition 4. We use a FE scheme to encrypt the messages that the verifier needs to process when running the

test. From §3 we recall that the test should decide the following predicate:

$$\text{test}_{UMTI}(\{\mathcal{E}\}, K) \text{ iff } (\text{NotifiedMarks} \subseteq \{(c, m) : (c, (q, a), m) \in \text{MarkedTests} \text{ and } (c, (q, a)) \in \text{SubmittedTests}\}) \quad (4)$$

The test should succeed if and only if the set of all notified marks to candidates is included in the set of exams that have been marked for those candidates and the exams marked should be those that have submitted for marking.

As we discussed in §4, in Remark! this test allows the verifier see all messages $\text{Sign}_{\text{Sk}_E, h_E}(M_C)$ which he needs to build the sets as in Equation (4). Thus, the verifier gets to know the questions and answers and their link with the candidate identifiers.

By resorting to a FE scheme we intend to achieve a version of the test that implements predicate (4) that does not reveal a candidate’s questions and answers, which is a sufficient condition to satisfy Definition 5. Figure 2 shows the new protocol, but reports only the messages that have been modified. We assume a trusted *FE authority* that generates the public Pk_{FE} from Msk . It can be any party, internal or external, but must be trusted in this particular task. It is implicit in the Figure 2, visible only when we use Pk_{FE} . Messages in step 5 and in step 8 are those which change. In 5 the exam authority encrypts with Pk_{FE} the signature of a test T_C that a candidate submitted for marking. It also post a digest of the same message, signed with the candidate’s pseudonym. To avoid that the message encrypted under Pk_{FE} and that encrypted under Pk_C hide two different values (i.e., y and T_C such that $H(T_C) \neq y$), we require a non-interactive zero knowledge proof (NIZK) of consistency, π in Figure. Similarly, in step 8 the authority encrypts with Pk_{FE} the marking and another NIZK π' is required.

5.2 Revising the Verifiability Test

Ciphertexts in 5 and in 8 will be used in combination with the tokens that the FE authority prepares for the verifier. It needs tokens that implement two functions:

The first, function f_1 , embeds the public-keys Pk_C and Pk_E and the mark $mark$ to be verified. It is supposed to input $\text{Sign}_{\text{Sk}_E, h_E}(M_C)$, the message in 8. After properly parsing the strings, checks whether the mark, say $mark'$, contained in M_C is equal to $mark$, checks the signature and checks whether the $\overline{\text{Pk}}_C'$ contained in T_C (in turn, contained in M_C) equals $\overline{\text{Pk}}_C$. It also verifies the validity of the zero-knowledge proof π . If a check fails, the function returns 0. If all checks succeed, it returns 1. Function f_1 is coded as follows:

<p>Function $f_1[\overline{Pk}_C, mark, Pk_E](\sigma)$</p> <ol style="list-style-type: none"> 1. If $VERIFY(\sigma, Pk_E) = 0$ then Return 0 <i>//verifies a signature, returns 0 on fails</i> 2. Parse σ as $Sign_{Sk_E, h_E}(M_C)$ 3. Parse M_C as $\langle Sign_{Sk_A}(\overline{Pk}_E, T_C), mark' \rangle$ (get $mark', T_C$) 4. Parse T_C as $\langle quest, ans, \overline{Pk}'_C \rangle$ (get \overline{Pk}'_C) 5. If $mark' \neq mark$ Return 0 6. If $\overline{Pk}'_C \neq \overline{Pk}_C$ Return 0 7. Return 1

The second function, f_2 , embeds the public-key \overline{Pk}_C and the message $Sign_{Sk_E, h_E}(M_C)$. It expects to input the second part of message 5, $Sign_{Sk_C, h_C}(T_C)$. It checks the signatures and whether both messages refer to the same exam test. The function returns 0 as soon as one check fails, 1 otherwise. Function f_2 is coded as follows:

<p>Function $f_2[\overline{Pk}_C, Sign_{Sk_E, h_E}(M_C)](\sigma)$</p> <ol style="list-style-type: none"> 1. If $VERIFY(\sigma, Pk_C) = 0$ then Return 0 <i>//verifies a signature, returns 0 on fails</i> 2. Parse σ as $Sign_{Sk_C, h_C}(T_C)$ (get T_C) 3. Parse M_C as $\langle Sign_{Sk_A}(\overline{Pk}_E, T'_C), mark \rangle$ (get T'_C) 4. If $T_C \neq T'_C$ Return 0 5. Return 1
--

The verifiability test uses two sets of tokens of form $Tok_{f_1[x]}^{Msk}$ and $Tok_{f_2[x]}^{Msk}$ where x ranges over the token embedded parameters.

The test then calls the function Eval over those tokens to check whether the f_1 and f_2 holds on the messages found on the bulletin board that correspond the the messages that the verifier needs to implement the test in Equation (4).

$test_{UMTI}(\{\mathcal{E}\}, K)$ first verifies whether for each notified mark per candidate there exists at least one message among those in step 8 that corresponds to the mark for a test of that candidate; if one is found the test continues and checks whether there is at least one message among those in step 5 that corresponds to that exam for that candidate submitted for marking.

The test requires to know the markings that have been notified to the candidates and PK_E the public key of the authority, so $K = \{(Pk_C, mark_C) : C \in ID_{rc}\} \cup \{Pk_E\}$. Formally the test is implemented by the following algorithm:

```

function testUMTI( $\{\mathcal{E}\}, K$ ):
  for  $(Pk_C, mark_C) \in K$  do
    Tok1  $\leftarrow$  GETTOK( $f_1[Pk_C, mark_C, Pk_E]$ )
    for  $(\{Sign_{Sk_E, h_E}(M_C)\}_{Pk_{FE}}, \pi) \in \{\mathcal{E}\}$  do
       $b_1 \leftarrow$  Eval(Tok1,  $\{Sign_{Sk_E, h_E}(M_C)\}_{Pk_{FE}}$ );
      if  $b_1$  then ExitLoop;
    if  $b_1$  then
       $b'_1 \leftarrow$  VERIFY( $\pi$  for  $R$ );
      Tok2  $\leftarrow$  GETTOK( $f_2[Sign_{Sk_E, h_E}(M_C)]$ );
      for  $(\{Sign_{Sk_C, h_C}(T_C)\}_{Pk_{FE}}, \pi') \in \{\mathcal{E}\}$  do
         $b_2 \leftarrow$  Eval(Tok2,  $\{Sign_{Sk_C, h_C}(T_C)\}_{Pk_{FE}}$ );
        if  $b_2$  then ExitLoop;
      if  $b_2$  then
         $b'_2 \leftarrow$  VERIFY( $\pi'$  for  $R'$ );
    return  $b_1 \cdot b'_1 \cdot b_2 \cdot b'_2$ 
    
```

GETTOK is used to retrieve a specific token. Here, we use the function's embedded parameters to indicate which token is being requested. However, the verifier does not necessarily know the value of the parameter to perform such request, but he must be able to indicate the token he needs e.g., by means of an index. Relations R and R' are so defined:

<p>Relation $R[\overline{Pk}_C, Pk_{FE}, H](x)$</p> <ol style="list-style-type: none"> 1. Parse x as $(Ct_1 = \{Sign_{Sk_A}(y)\}_{\overline{Pk}_C},$ $Ct_2 = \{Sign_{Sk_E, h_E}(T_C)\}_{Pk_{FE}})$ (get y and T_C) 2. If $H(T_C) \neq y$ Return 0 3. Return 1

<p>Relation $R'[\overline{Pk}_C, Pk_{FE}](x)$</p> <ol style="list-style-type: none"> 1. Parse x as $(Ct_1 = \{Sign_{Sk_E, h_E}(y)\}_{\overline{Pk}_C},$ $Ct_2 = \{Sign_{Sk_C, h_C}(T_C)\}_{Pk_{FE}})$ (get y and T_C) 2. If $T_C \neq y$ Return 0 3. Return 1

Discussion. We argue that $test_{UMTI}(\{\mathcal{E}\}, K)$ is privacy preserving. By assumption, the FE scheme satisfies the indistinguishability security. Thus, a Dolev-Yao attacker, given a ciphertext Ct encrypting a message m and a token for a function f , can only derive $f(m)$. This holds also for a curious verifier who observes the tokens for the two functions f_1 and f_2 . The crucial consideration is that, by construction, such two functions return a *Boolean* value and nothing about the messages that are considered in our model (cf. Definition 1) of an exam's execution. Moreover, among the messages that the verifier gets in input, precisely in K , there is nothing that allows him to decrypt other messages on the bulletin board. Actually, all the decryption operations that the verifier needs are realized within the function Eval. Therefore, it follows that $test_{UMTI}(\{\mathcal{E}\}, K)$ is privacy preserving according to Definition 4.

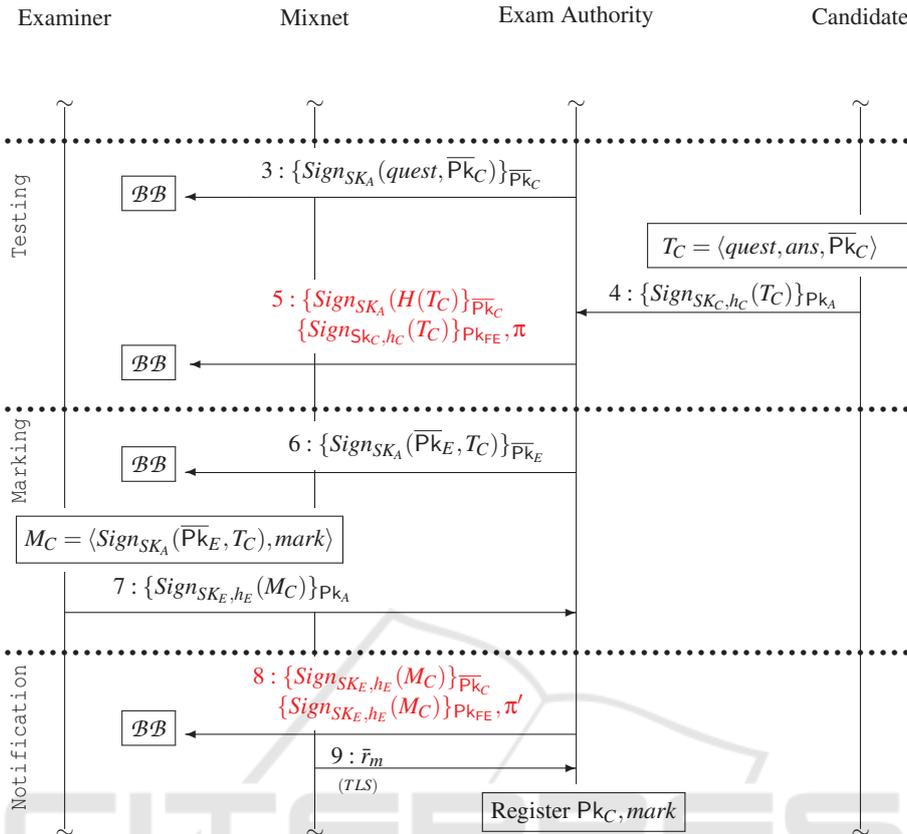


Figure 2: A revision of Remark! using FE (only phases that have been changed).

6 FORMAL ANALYSIS

We prove formally that the verifiability test is sound and complete and that the new protocol does satisfy the same security properties as the original Remark!. This exercise of formal analysis is necessary because we intend to get insurance than we achieve privacy-preserving verifiability incrementally, on top of all the authentication, integrity and privacy properties holding on the protocol and not on their expenses.

We perform the analysis in Proverif (Blanchet, 2014). Our ProVerif model of the protocol is an extension of the original Remark! model as it was presented in (Dreier et al., 2014). The extension mainly regards the following aspects:

- A ProVerif specification that reflects the new protocol description.
- A new equational theory for the Functional Encryption primitives defined in our protocol.
- A new UMTI verifiability test.

The equational theory is illustrated in Table 1 to model the cryptographic primitives of the protocol.

Table 1: Equational theory in formal analysis of the privacy-preserving verifiability version of Remark!

Name	Equation
Signature	$getmess(sig(m, k)) = m$
	$checksig(sig(m, k), spk(k)) = m$
	$checksig(sig(m, ps_pr(k, exp(rce))), ps_pub(pk(k), rce)) = m$
ElGamal	$decrypt(enc(m, pk(k), r), k) = m$
	$decrypt(enc(m, ps_pub(pk(k), rce), r), ps_pr(k, exp(rce))) = m$
	$checkps(ps_pub(pk(k), rce), ps_pr(k, exp(rce))) = true$
FE f1	$eval_{f1}(pk(k), ps_pub(pk(k), rc), ps_pub(pk(k), re), mark, enc(sig(sig(ps_pub(pk(k), re), (q, a, ps_pub(pk(k), rc)), k), mark, ps_pr(k, exp(re))), pk(k))) = true$
FE f2	$eval_{f2}(pk(k), ps_pub(pk(k), rc), enc(sig((q, a, ps_pub(pk(k), rc)), ps_pr(k, exp(rce))), pk(k))) = true$

The equations that model the digital signature are rather standard in ProVerif.

ElGamal encryption is extended with equations that model pseudonyms as public keys: the pseudonym, which also serves as test identifier, can be generated using the function ps_pub which takes in a public key and a random exponent. In fact, this function models the main feature of exponentiation mixnet. Function ps_pr can be used by a principal to decrypt or sign anonymous messages. The function takes in the private key of the principal and the new generator published by the mixnet. Function $checkps$ allows a principal to check whether a pseudonym is associated with the principal's private key. In practice, principals use this function to identify their pseudonyms published on the bulletin board. A public channel models the bulletin board.

The equations that defines $eval_f1$ and $eval_f2$ model the two Functional Encryption primitives f_1 and f_2 respectively. They return *true* if and only if the corresponding plaintext version f_1 and f_2 return *true*. Note that our equational theory of functional encryption captures the property of verifiable FE. Whenever an encrypted message has a form that is not consistent with the constructor enc , the deterministic $eval$ fails, and no rule can be applied. We do not explicitly model the generation of the token: Setup and KeyGen are generated by a trustworthy authority. It follows that the only two allowable functions are f_1 and f_2 .

Verifiability. From Definition 3, we recall that a protocol is p -verifiable if it admits a sound and complete algorithm that decides p . Soundness means that the verifiability-test returns *true* only if the property holds. Completeness means that if the property holds, the verifiability-test always returns *true*.

We use ProVerif's correspondence assertions to prove soundness. The verification strategy consists of checking that the event OK emitted by the verifiability-test when it is supposed to return *true* is always preceded by the event emitted in the part of the code where the property becomes satisfied. In case of Mark and Test Integrity, this happens when the exam authority assigns the mark to the candidate.

We resort to the unreachability of the event KO , which is emitted by the verifiability-test when it is supposed to return *false*, to prove completeness. In this case, the ProVerif model enforces only honest principals and prevents the attacker to manipulate the input data of the verifiability-test. In fact, a complete verifiability-test must succeed if its input data is correct. ProVerif confirms that our protocol meets UMTI verifiability and the verifiability-test is sound

and complete¹.

We observe that ProVerif can only prove the case in which only one entry on the bulletin board is considered. Mark Integrity verifiability requires that *all* and only the marks associated to the tests are assigned to the corresponding candidates. Thus, we need to iterate over all candidates who submitted their tests, but ProVerif cannot do this automatically as it does not support loops. We resort on the manual induction proofs provided by Dreier et al. (Dreier et al., 2014) in which the base case is given by the automated result in ProVerif, and then generalise such result to the general case with an arbitrary number of candidates, tests, and marks. Those proofs applies to the original Remark! as well as to this new version. In fact, the assumption that the auditor can check the correspondences of the assigned entries to the marked entries on the bulletin board does not change: each entry that appears on the bulletin board at step 6 of the protocol can be matched to an entry that appears on the bulletin board at step 8 of the protocol.

Authentication. The new protocol meets all the original authentication properties of Remark!. Since the new protocol differs from Remark! only in messages 5 and 8, we could reuse the same authentication definitions advanced by Dreier et al. (Dreier et al., 2014) and verify them in ProVerif.

7 CONCLUSION

In this pioneering work, we put forth a new notion of security that entails features of both verifiability and privacy. To illustrate the benefits of our framework, we proved that our definition is meaningful enough to empower applications in the concrete domain of electronic exams and in particular to an extension of an exam protocol that satisfies several verifiability properties but which is not compliantly with our privacy-preserving requirement.

To achieve our goals, we employ the cryptographic tool of Functional Encryption (FE). Only with FE we can verify obviously a property that is global over the execution of the protocol, a property that is about the data handled and processed by several agents in the protocol's run. In making this choice we considered to obtained the same effect with other tools, the most promising be zero knowledge proofs, but, we claim, using this tool would have required an authority that looks at the execution and adds

¹The full ProVerif code is available on request.

the proofs and that interacts with the several protocol agent's asking them for other Zero-Knowledge proofs, so changing too much how an exam should work. This argument requires however a proof and we plan to inquire into this question as future work.

We have not benchmarked the new protocol. At current stage, our result is only of theoretical relevance due to the high computational cost of FE for circuits that we assumed and for this reason we recognize that FE may look as be an overkill to someone. However, there is another feature of electronic exams that must be considered here. The limited dimension of the exam's audiences and the expected time of an exam's notification makes feasible implementations that rely on time-inefficient encryption schemes. Comparing with electronic voting, for instance, where a whole country is involved and where a result is nowadays expected be announced within the day, for an exam the expected audiences is definitely far more contained while waiting weeks is a perfectly acceptable time frame to get notified of the result. We defer to further research about implementing our privacy-preserving verifiability notion efficiently.

It should be stressed that although contextualized in reference to exams, our research is not bound to work in that domain only. The notion of privacy-preserving verifiability is abstract and the solution that we propose to ensure privacy-preservation is demonstrated for a universal verifiability test of a common integrity and authentication property. Therefore it seems plausible to apply our results in other domains, like voting or auction, where the verifiability properties are also about integrity and authentication. Proving this claim is future work.

We conclude by pointing to a future work for us and an open problem to whom it may be interested: to study the relation between our notion and that presented by Mödersheim *et al.* of α - β privacy (Mödersheim *et al.*, 2013). Were this correlation proved, we could gain a straightforward way to verify formally privacy-preserving verifiability through the fact α - β privacy subsumes static equivalence.

ACKNOWLEDGEMENTS

R. Giustolisi's research is supported in part by DemTech grant 10-092309, Danish Council for Strategic Research, Programme Commission on Strategic Growth Technologies; G. Lenzini's by the SnT / pEp Security partnership project "Protocol of Privacy Security Analysis". V. Iovino is supported by a FNR CORE grant (no. FNR11299247) of the Luxembourg National Research Fund.

REFERENCES

- Adida, B. and Neff, C. A. (2006). Ballot Casting Assurance. In *Proc. of the USENIX/Accurate Electronic Voting Technology*, EVT'06, p. 7–7. USENIX Association.
- Benaloh, J. (1996). *Verifiable Secret-Ballot Elections*. PhD thesis, Yale University.
- Benaloh, J. and Tuinstra, D. (1994). Receipt-free secret-ballot elections. In *26th Sym. on Theory of Computing*, STOC'94, p. 544–553, New York, NY, USA. ACM.
- Blanchet, B. (2014). Automatic Verification of Security Protocols in the Symbolic Model: the Verifier ProVerif. In *Foundations of Security Analysis and Design VII, FOSAD Tutorial Lectures*, volume 8604 of LNCS, p. 54–87. Springer.
- Boneh, D., Sahai, A., and Waters, B. (2011). Functional encryption: Definitions and challenges. In *8th Theory of Cryptography Conf.*, TCC 11, volume 6597 of LNCS, p. 253–273. Springer.
- Cohen, J. and Fischer, M. (1985). A robust and verifiable cryptographically secure election scheme. In *26th Ann. Sym. on Foundations of Computer Science*, FOCS'85, p. 372–382, Portland, Oregon, USA. IEEE Computer Society.
- Cuvelier, E., Pereira, O., and Peters, T. (2013). *Election Verifiability or Ballot Privacy: Do We Need to Choose?*, p. 481–498. Springer.
- De Santis, A., Micali, S., and Persiano, G. (1988). Non-interactive zero-knowledge proof systems. In *Advances in Cryptology*, CRYPTO87, volume 293 of LNCS, p. 52–72. Springer.
- Department of Defence (1987). DoD Personnel Security Program. DOD 5200.2 R.
- Dreier, J., Giustolisi, R., Kassem, A., Lafourcade, P., and Lenzini, G. (2015). A Framework for Analyzing Verifiability in Traditional and Electronic Exams. In *11th Int. Conf. Information Security Practice and Experience*, ISPEC 2015, volume 9065 of LNCS, p. 514–529, 2015. Springer.
- Dreier, J., Giustolisi, R., Kassem, A., Lafourcade, P., Lenzini, G., and Ryan, P. Y. A. (2014). Formal analysis of electronic exams. In *11th Int. Conf. on Security and Cryptography*, SECURE 14, August, p. 101–112. SciTePress.
- Giustolisi, R., Lenzini, G., and Ryan, P. Y. A. (2014). *Remark!: A Secure Protocol for Remote Exams*, volume 8809 of LNCS, p. 38–48. Springer.
- Hirt, M. and Sako, K. (2000). Efficient receipt-free voting based on homomorphic encryption. In *19th Ann. Conf. on the Theory and Applications of Cryptographic Techniques: Advances in Cryptology (EUROCRYPT'00)*, volume 1807 of LNCS, p. 539–556. Springer.
- Jukna, S. (2012). *Boolean Function Complexity - Advances and Frontiers*, volume 27 of *Algorithms and combinatorics*. Springer.
- Kremer, S., Ryan, M., and Smyth, B. (2010). Election verifiability in electronic voting protocols. In *15th Eur. Sym. on Research in Computer Security*,

- ESORICS'10, volume 6345 of *LNCS*, p. 389–404. Springer.
- Küsters, R., Truderung, T., and Vogt, A. (2010). Accountability: definition and relationship to verifiability. In *17th Conf. on Computer and Communications Security, CCS'10*, p. 526–535. ACM.
- Mödersheim, S. A., Groß, T., and Viganò, L. (2013). *Defining Privacy Is Supposed to Be Easy*, p. 619–635. Springer.
- Naveed, M., Kamara, S., and Wright, C. V. (2015). Inference Attacks on Property-Preserving Encrypted Databases. In *22nd ACM SIGSAC Conf. on Computer and Communications Security, CCS'15*, p. 644–655, New York, NY, USA. ACM.
- Smyth, B., Ryan, M., Kremer, S., and Mounira, K. (2010). Towards automatic analysis of election verifiability properties. In *Work. on Automated Reasoning for Security Protocol Analysis and Issues in the Theory of Security, ARSPA-WITS'10*, volume 6186 of *LNCS*, p. 146–163. Springer.

