A Browser-based Anonymous Questionnaire System with User-controlled Linkability

João Silveira
Instituto Superior Técnico, Universidade de Lisboa
École Polytechnique Fédérale de Lausanne

***Abstract*—The use of online tools for classrooms, such as quizzes and questionnaires, is becoming widespread. Their popularity is due to their ease-of-use, versatility, and ability to engage students. However, with the amount of privacy-sensitive data gathered by these systems, and the recent introduction of regulations such as the GDPR, this becomes a critical concern. We present a browser-based *privacy-enhancing questionnaire system* that supports fully anonymous and unlinkable answers. Additionally, education researchers can ask students for permission to link a subset of their answers to questionnaires. Students can then accept or reject these requests, exclusively enabling the corresponding linkability. We describe the design of this system and its implementation in a proof-of-concept demo. Furthermore, we introduce a cryptographic scheme for user-controlled linkability of questionnaire answers with a two-phase linking authorization mechanism. We also describe the implementations of this scheme, and of the underlying attribute-based credential scheme.**

I. Introduction

We see increasingly often the use of online tools to aid instructors in engaging students in class activities. One of the most common examples is the use of online questionnaires where students can easily answer on their phones or laptops questions proposed by the instructor.

On the other hand, one should consider the downsides of using these tools. An important aspect is the amount of privacy-sensitive data that these systems gather. Both instructors and the platforms they use can learn a large amount of information about the students. More recently, with the introduction of the EU General Data Protection Regulation (GDPR) [1], what can be done with this information has become very limited. One alternative is to make students fully anonymous, making their progress over time practically impossible to track.

However, when introducing education researchers into such a privacy-enhancing system, their objectives become rather impossible to achieve. Education researchers want to understand how students evolve over time, by linking their answers. In our context, as the privacy of users must be preserved, education researchers should only be allowed *privacy-preserving linking*, which implies linkability between different results but not between results and the student who produced them. However, the excessive linking of answers over time could amount to a comprehensive profiling of each student, possibly resulting in the student's de-anonymization.

In this work we present the design of a browser-based anonymous questionnaire system, that allows for user-controlled linkability of answers. Furthermore, we developed a proof-of-concept demo of the core functionality of the system. In addition, we implemented a TypeScript library of an attribute-based credential scheme for general purpose use, that we used in the implementation of the demo. Finally, we designed a scheme that allows for the cryptographic linking of answers, featuring a two-phase linking authorization. This scheme was also implemented in a TypeScript library, also used in our demo.

II. Background

A. Zero-Knowledge Proofs of Knowledge (ZKPs)

ZKPs allow a prover, to convince a verifier, that she possesses a value or set of values. However, this proof should not give away any information about the values besides the validity of the proven statement. This means that the verifier should only learn that the prover indeed possesses these values and nothing else.

To focus on the semantics of the protocol instead of its implementation, we introduce the following notation, adapted from Camenisch and Stadler [2], to represent zero-knowledge proofs in a more generic way:

$$\text{PK}\{(x_1, ..., x_n) : \text{statements about } x_1, ..., x_n\}$$

where $x_1, ..., x_n$ are secrets and any other variables used in the statements are assumed to be public. For example, the expression $PK\{(x) : C = g^x\}$, represents the proof of knowledge of a secret value x, where C and g are public, and $C = g^x$.

B. Attribute-Based Credentials

Attribute-based credentials (ABCs), also known as *anonymous credentials* allow users to authenticate to a certain service provider (SP), without the SP learning the identity of the user. The SP learns only a statement about the attributes which is necessary to grant access to the user. Here, zero-knowledge proofs are used as a building block.

III. Related Work

A. PS Credentials

In this section we describe the attribute-based credential scheme we use in our system. It is an adaptation of the randomizable signatures scheme by Pointcheval and Sanders [3, Section 6], which we refer to as PS Credentials.

The following sections explain in detail the Pointcheval and Sanders' credential scheme [3, Section 6]. First we introduce

the involved parties followed by the explanation of the scheme itself.

1) Parties: The scheme consists of the following parties which interact with each other. There exists one issuer and any number of users and verifiers.

- *Issuer*: responsible for issuing credentials to users. The issuer's signing key is public.
- *User*: uses the credentials. The user interacts with the issuer to obtain the credentials, and shows the credential to the verifier to obtain access to a service or resource. If a user successfully obtains a credential from the issuer she is considered a legitimate user.
- *Verifier*: verifies the validity of credentials shown by users. In a real-world application of the scheme, a verifier could be a service provider for example.

Regarding the threat model we need to assume the issuer only issues credentials to honest users, i.e., it does not collude with malicious users. We make otherwise no assumptions on the behaviour of users and verifiers.

2) Scheme: We now present the scheme more formally and in detail. We start by presenting the setting, followed by the algorithms and protocols run by the parties.

a) Setting: Let $(\mathbb{G}_1, \mathbb{G}_2)$ be a bilinear group pair of prime order p and $e : \mathbb{G}_1 \times \mathbb{G}_2 \to \mathbb{G}_T$ the corresponding type 3 bilinear map. The value $r \in \mathbb{N}$ is a fixed parameter of the scheme – it is the number of attributes the credential carries.

b) Key Generation: This algorithm generates a key pair for the issuer. It starts by randomly picking a generator of $\mathbb{G}_1$ and $\mathbb{G}_2$, respectively $g \in \mathbb{G}_1$ and $\tilde{g} \in \mathbb{G}_2$. Then it randomly selects $(x, y_1, ..., y_r) \in \mathbb{Z}_p^{r+1}$, and computes $(X, Y_1, ..., Y_r) = (g^x, g^{y_1}, ..., g^{y_r})$ and $(\tilde{X}, \tilde{Y_1}, ..., \tilde{Y_r}) = (\tilde{g}^x, \tilde{g}^{y_1}, ..., \tilde{g}^{y_r})$. Finally, it sets the secret key as $\mathtt{sk} \leftarrow X$, and the public key is set as $\mathtt{pk} \leftarrow (g, Y_1, ..., Y_r, \tilde{g}, \tilde{X}, \tilde{Y_1}, ..., \tilde{Y_r})$.

c) Issuing a Credential: This protocol is carried out between the user and the issuer. The user wants a signature on attributes $(a_1, ..., a_r)$. Let $\mathcal{I} \subseteq \{1, ..., r\}$ be the set of indices of attributes determined by the issuer, and $\mathcal{U} = \{1, ..., r\} \backslash \mathcal{I}$ the set of indices of attributes determined by the user. The user first picks a random $t \in \mathbb{Z}_p$ and computes a commitment on t and the attributes, $C = g^t \prod_{i \in \mathcal{U}} Y_i^{a_i}$. The user then sends C to the issuer along with a proof of knowledge of the opening of the commitment. If the proof is valid, the issuer selects a random $u \in \mathbb{Z}_p$ and returns the blind signature $\sigma' = (\sigma'_1, \sigma'_2)$ to the user, where $\sigma'_1 = g^u$ and $\sigma'_2 = \big(XC \prod_{i \in \mathcal{I}} Y_i^{a_i}\big)^u$. The user obtains the signature by unblinding σ', which yields $\sigma = (\sigma'_1, \sigma'_2 / {\sigma'_1}^t)$. Finally, the user can store the credential after confirming it is valid, by checking if

$$e\big(\sigma_1, \tilde{X} \cdot \prod_{i=1}^{r} \tilde{Y}_i^{a_i}\big) = e(\sigma_2, \tilde{g}).$$

d) Showing a Credential: This protocol is carried out between the user and the verifier. Let $\mathcal{D} \subseteq \{1, ..., r\}$ be the set of indices of attributes the user wants to disclose when showing the credential, and $\mathcal{H} = \{1, ..., r\} \backslash \mathcal{D}$ the set of indices of attributes the user wishes to hide but still prove knowledge of. The user starts by picking random values $r, t \in \mathbb{Z}_p$ and computing $\sigma' = \big(\sigma_1^r, (\sigma_2 \cdot \sigma_1^t)^r\big)$. Then the user sends σ' to the verifier, engages in the proof of knowledge

$$\pi = \text{PK}\Big\{(a_{i \in \mathcal{H}}, t) : e(\sigma'_1, \tilde{X}) \cdot \prod_{i \in \mathcal{H}} e(\sigma'_1, \tilde{Y}_i)^{a_i} \cdot \prod_{j \in \mathcal{D}} e(\sigma'_1, \tilde{Y}_j)^{a_j} \cdot e(\sigma'_1, \tilde{g})^t = e(\sigma'_2, \tilde{g})\Big\},$$

and discloses the attributes in $\mathcal{D}$, i.e., sends the verifier attributes a_j for all $j \in \mathcal{D}$. The verifier judges the validity of the credential by assessing if π is valid.

This type of credential can also be used as a signature over a message using the Fiat-Shamir heuristic [4].

B. Direct Anonymous Attestation

Direct Anonymous Attestation (DAA) was presented by Brickell et al. [5] and it is described by the authors as a group signature scheme without revocable anonymity. This scheme allows for the use of pseudonyms which give users control over the linkability of their signatures, i.e. they can decide whether a signature is linkable to other signatures. Although interesting, this feature makes signatures instantly linkable, while we look for a scheme with a two-phase linking authorization mechanism.

C. Conditional Linkability

We now describe *Vote to Link* [6]. The basic idea is to have a system where users can ask Service Providers (SPs) to perform transactions, which are anonymous. However, if a user is acting maliciously, moderators can intervene. If at least *k* moderators agree that an action is malicious, all other actions performed by the malicious author within this *epoch* (time frame during which the malicious action was performed) are linked. However, the user remains anonymous and no other actions outside the epoch are linkable.

Now we will look into how Vote to Link [6] works more informally.

a) Linking token: A user's linking token for epoch ϵ is given by $r = H(\epsilon)^x$. This token is usually a secret known only by the user. Also, for each transaction the user performs, the following two auxiliary values are published, $t_1 = h^z$ and $t_2 = \hat{e}(H(\epsilon), h^z)^x$, where $z \in \mathbb{Z}_p$ is randomly generated.

b) Performing a transaction: The user computes the linking token r and auxiliary values t_1 and t_2. Next, the user creates the encrypted linking token $T = \mathtt{TDH.Enc}(w, r, \tau)$ and, to prove that all these values were generated correctly and that they correspond to her secret key, she makes the following signature proof of knowledge over the transaction τ:

$$\pi = \text{SPK}\{(C, x, z, \alpha) : C(x) \wedge t_1 = h^z \wedge t_1^x = h^\alpha \wedge T = \mathtt{TDH.Enc}(w, H(\epsilon)^x, \tau) \wedge t_2 = \hat{e}(H(\epsilon), h)^\alpha\}(\tau)$$

where $\alpha = xz$ and $C(x)$ denotes a credential over the secret key x. The user sends the transaction record $t = (\tau, T, t_1, t_2, \epsilon, \pi)$ to the SP, who performs the transaction if π is correct.

c) Linking transactions: When at least k moderators vote to link a user's transaction of record $t = (\tau, T, t_1, t_2, \epsilon, \pi)$, the SP is able to recover the linking token r from the encrypted token T. Having r, it is now possible to know which other transactions the user performed during epoch ϵ. To do so, for every transaction $t' = (\tau', T', t'_1, t'_2, \epsilon, \pi')$ in epoch ϵ, the SP knows that τ' was performed by the same user if

$$\hat{e}(r, t'_1) = t'_2.$$

IV. System Design

In this chapter we describe the design of our anonymous questionnaire system. We try to motivate our design choices and present how they influence the system's behaviour. We also try to put the several components in context, and to illustrate the interactions that occur between them. We now start by introducing a few important concepts.

A *questionnaire* is a non-empty set of questions created by an instructor and meant for students to answer. The format of questions is not relevant – they can be multiple choice or require a paragraph as an answer. A questionnaire may have one or more questions that can be answered. However, we are not concerned with each question's answer, but with the set of answers that make up a response to an entire questionnaire. Thus, we say an *answer* is the entire response to a questionnaire. Lastly, a *study* is a set of questionnaires of which a researcher wants to be able to link the answers. For example, we can have a study involving four questionnaires, which means the researcher responsible for the study, wishes to be able to link students' answers across these four questionnaires.

A. Parties

Here we describe the parties that participate in our system, and explain what type of assumptions we made for our security model. The following entities mostly interact with the system through a web browser and are essentially the users of the system.

- *Students*, often referred to simply as *users*: participate in questionnaires. They answer questionnaires anonymously and, if they give due permission, allow education researchers to link a subset of their answers.
- *Instructors*: create the questionnaires and make them available to students. They are only interested in the class's overall performance, and not that of individual students over time. Thus, instructors need not assign results to students, nor link answers to tests by the same student across different questionnaires. Instructors are therefore not able to link any answers.
- *Education Researchers (ERs)*: are interested in linking student's answers to assess how student performance evolves over time. They ask students to link certain questionnaires and are then able to do so if the students allow it.

We now present the entities behind the servers in our system. They are responsible for serving a number of web pages which allow the above described parties to interact and take part in the system.

- *Issuer*: responsible for issuing anonymous credentials. These credentials are used by the students to unlinkably sign answers, guaranteeing their authenticity. We assume that the issuer does not collude with malicious users, which implies that the issuer only issues credentials to legitimate users.
- *e-Learning Platform (eLP)*: serves as a trusted entry point of authentication. The eLP is platform for instructors to share resources with students. A user is legitimate if she can access the eLP and gain access to the issuer through it. We must thus assume the eLP provides only legitimate users with valid access to the issuer.
- *Questionnaire System (QS)*: the central entity in our system. It hosts the questionnaires and gathers the answers and associated information.

B. Security Properties

The following are the security properties we wish to satisfy.

- *Student anonymity*: answers to questionnaires cannot be linked to the identity of the student who produced them.
- *Answer unlinkability*: without the student's authorization, two different answers made by the same student appear as related to each other as two randomly picked answers. In other words, two answers made by the same student cannot be linked without the explicit authorization of the student.
- *User-controlled linkability*: the user (or student) is in complete control of which of her answers are linked by researchers. What each different researcher is able to link is also fully controllable by the user.
- *Two-phase linking authorization*: a student can decide to include an answer in a study at an initial phase without the answer becoming instantly linkable. The linking happens at a later time with a second confirmation action by the student.
- *Answer authenticity*: a recorded answer to a questionnaire must have been produced by an authorized student.
- *One-time participation*: a student should be able to register a valid answer to a questionnaire at most once.

C. Base Scenario

Now that we have introduced the parties in our system we can introduce our basic interaction scenario. In Figure 1 we can see how the action would play out for a student answering a questionnaire – the numbers represent the steps in chronological order. The user starts by (1) requesting the eLP page to the eLP. The eLP (2) responds with the eLP page, which is rendered on the user's browser. The user (3) clicks on the link available on the eLP page to access the questionnaire, hosted on the QS server. The (4) questionnaire page is rendered, the student answers the questionnaire and (5) submits it to the QS server.

Note that a more basic scenario would involve a student directly accessing the QS. We ignore that scenario here as it is less realistic and just a simpler version of our scenario, requiring the same or less assumptions.

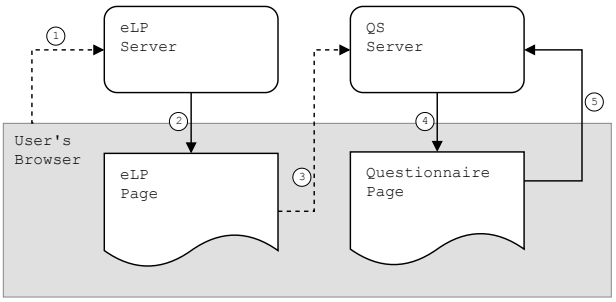


Fig. 1: Base interaction scenario for answering a questionnaire.

In the next sections we will analyse what kind of changes and assumptions we need to make in order to achieve student anonymity and unlinkability, and the authenticity of the received answers.

D. Achieving Anonymity

We now look at how we must design the interaction between the parties to guarantee that a user remains anonymous after answering a questionnaire. This means that none of the parties can link an answered questionnaire to the student who produced the answers. First we identify two levels of anonymity: *application* and *network*. We consider each one individually, but our goal is to satisfy both.

Before we go further we introduce the notion of *covert adversaries* [7]. Roughly, a covert adversary is an adversary that will only act maliciously in situations where the deviant actions cannot be detected. In case the misbehaviour is detectable, a covert adversary will correctly follow the protocol, i.e. will not actively act against honest users. This definition implies that this type of adversary may always be passively malicious. Seeing that the trust of users is at stake, as a lack of it may cause users to cease using the system, we believe that in our context this type of adversary is realistic.

However, in the next sections, unless otherwise stated, we assume all parties can have arbitrary malicious behaviour.

1) Application-level Anonymity: On the application-level we consider the identifying information that can be found on the application layer. Here we don't consider network leakage, i.e. we assume the attacker does not see IP addresses or other identifying information found on the network layer. In the next section we will see how to tackle a more powerful adversary with access to network layer data. Looking at our base scenario in Figure 1 we see that if the eLP, who knows the identity of its users, prepares the link to the QS with a request including identifying information from the user, the QS is able to learn who its users are. Additionally, if the QS constructs the questionnaire page so that submitting the answers also sends along student identifying information (obtained through the eLP), anonymity is broken.

To satisfy the anonymity of answers we assume the eLP is a covert adversary and will therefore not append any student information to the request directed at the QS. This way we guarantee the QS will not possess any identifying information to begin with and thus we need not make any assumptions on its behaviour.

2) Network-level Anonymity: Now that we can satisfy application-level anonymity, we look deeper into the network layer. We want to make sure students can answer questionnaires anonymously and unlinkably even when attackers can see the IP addresses used in communications.

In the scenario we described so far, even with a covert eLP, the QS can link each user's answer to their IP address. This breaks unlinkability as well as anonymity (if we consider the IP address as identifying information).

We fix this problem by introducing an anonymous channel for students to send their answers to the QS. Figure 2 illustrates this scenario. Steps 1 to 4 are the same as before, but now the students' answers are (5) sent to the QS server over an anonymous channel.

Now the QS cannot see the real IP address of the user when it receives the answers, therefore not being able to directly link answers to IP addresses. However, the QS can now proceed similarly as described in Section IV-D1 to break anonymity – the QS can construct the page so that it sends the student's IP address along with the answers, as it can see the IP when the page request is received (Figure 2, step 3). We propose two alternatives to mitigate this situation:

- Introduce an additional anonymous channel for students to make the initial request for the questionnaire page (Figure 2, step 3). This way the QS never sees the user's IP and thus cannot break linkability or anonymity. Note that it is important to keep the assumption that the eLP is covert in order to maintain application-level anonymity.
- Alternatively, we can assume the QS is a covert adversary. This means the QS will not try to append user information to the answers, which would be detectable behaviour. In this case we can drop the assumption that the eLP is a covert adversary – if the eLP sends user identifying information to the QS, this will not break any of our properties. Under the covert assumption we know the QS will not try to append the information from the eLP to the answers, or attempt to send the answers through a non-anonymous channel.

In view of the alternatives we opt for the latter under the principle that assumptions should be kept to a minimum as well as the complexity of the system's design.

3) Timing Attacks: The scenario we described so far is structurally vulnerable to timing attacks. This is a possible way to carry out such an attack: the QS sees the incoming page request and remembers the IP address of the requesting user; using timing information, it can now try to link that address to the answers received through the anonymous channel.

In the context of classroom questionnaires, several students will answer the same questionnaire at virtually the same time, and we can also expect every questionnaire to take different amounts of time to complete. This should introduce enough noise to make timing correlations ineffective. Taking this into

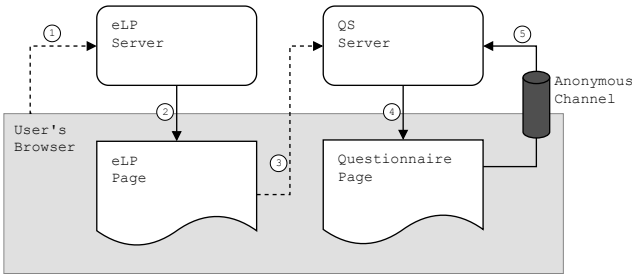


Fig. 2: Introducing an anonymous channel for students to send answers to the QS.

account, we consider it reasonable to model our system under the assumption that this type of attack is not possible.

However, if we were to allow timing attacks, we could assume the QS to be trustworthy and have it hold the answers for a certain time before making them available to researchers.

E. Achieving Authenticity

So far we have managed to design our system to guarantee student anonymity and unlinkability of answers, however we still need to ensure that the answers that are being recorded by the system come from legitimate and authorized users.

To this purpose we introduce in our scenario the use of credentials. We require that the credential does not de-anonymize the student and that several uses of the same credential are unlinkable, so that we can add authenticity without losing anonymity or unlinkability. We will use attribute-based credentials, which satisfy these requirements.

We now give a short reminder on how these credentials work. A user first interacts with an issuer through an issuance protocol, after which the user obtains the credential. The user can then interact with verifiers, who are able to check the validity of the credentials against the issuer's public key. Every time the user shows the credential to a verifier, she remains anonymous and unlinkable. It is also possible to use the showing of a credential as a signature over a message, which works as a way of proving that the message was sent by a user in possession of a valid credential. This last feature will be particularly useful in our construction.

By introducing the credentials we can now easily change our scenario to guarantee answer authenticity. Before sending the answer to the QS over the anonymous channel, the user should sign the answer with a valid credential. Assuming it trusts the issuer, the QS can now check the validity of the credential and therefore decide whether the received answers are authentic or not.

To obtain the credential the user must interact with the issuer to carry out the issuance protocol. We must carefully model this interaction to make sure our properties are not broken. We start by introducing the LTI standard as it will prove useful.

1) Learning Tools Interoperability: The *Learning Tools Interoperability (LTI)*, designed by IMS Global, is a standard aimed at facilitating and improving the security in the connection between Learning Management Systems (LMS) – e.g. Moodle or FenixEdu – and online learning applications and resources. The main idea is to make this connection seamless, as though the external resource was part of the LMS course page. The standard describes two entities: the Tool Consumer (TC), which is the LMS; and the Tool Provider (TP), representing the external resource. The specification includes a protocol for a secure and authenticated connection from the TC to the TP, as well as sharing the context between the environments. The latter allows, as an example, for the TP to see the roles set in the LMS (such as student or instructor), or for the LMS to retrieve a grade for a task a student completed using an external tool.

2) Issuing Credentials: Figure 3 schematizes the issuing process. The (1) user requests access to the eLP's page and the (2) eLP server responds with its main page which is rendered by the user's browser. The (3) user and the server then interact to carry out the necessary authentication protocol – e.g. signing in using the university credentials. Once authenticated the (4) user asks the issuer for the issuance page using the LTI standard for authentication. The (5) issuer responds with the issuance page and triggers (6) the issuance protocol which is then carried out. At the end of the protocol the (7) user stores the credential locally and it is then ready to use. Note that if the student has already authenticated with the eLP, we can skip step 3 of the procedure.

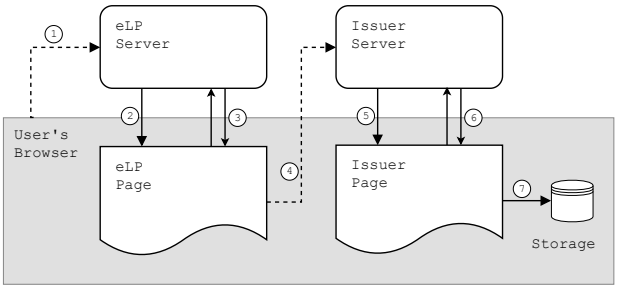


Fig. 3: Issuing credentials.

Thanks to the nature of the credentials, even the issuer has no information to link them or identify their user. This allows us to refrain from introducing additional anonymous channels and keep the scenario simple. We need, however, to make one trust assumption: the issuer must trust that the eLP only gives authorized users access to it.

F. Pseudonyms

We introduce domain-specific pseudonyms to ensure that users possessing a valid credential can only answer the same questionnaire once. Otherwise, due to our anonymity-oriented construction, a user answering the same questionnaire several times would appear to be a different user for each one of the answers.

A pseudonym is a value which is computed using a key and a domain, in our case the user's secret key and a questionnaire

identifier, respectively. Thus, for each pair of questionnaire identifier and user there exists a single possible pseudonym. More formally, a pseudonym is computed as follows. Let $\mathbb{G}_1$ be a cyclic group of prime order p, and $H : \{0,1\}^* \to \mathbb{G}_1$ a cryptographic hash function that maps strings to group elements. We compute the pseudonym as

$$nym = H(\texttt{qID})^x,$$

where $\texttt{qID} \in \{0,1\}^*$ is the questionnaire identifier, and $x \in \mathbb{Z}_p$ is the user's secret key. The user should also generate a proof that the pseudonym is well formed. This is accomplished by the signature proof of knowledge

$$\pi = SPK\{(C, x) : C(x) \wedge nym = H(\texttt{qID})^x\}(answer),$$

where $C(x)$ denotes a credential over the secret key x. Adding $C(x)$ to π proves that the key used in the pseudonym is the same as the one in the credential.

The application of pseudonyms to satisfy the one-time participation property is fairly straightforward. The user sends the pseudonym along with every answer, as well as the proof π. Upon receiving an answer, the QS verifies if the pseudonym and the proof are valid, in which case it checks if the pseudonym has already been used before. The QS finally discards the invalid answers and the ones with repeated pseudonyms.

G. Achieving Linkability

Finally we need to add the ability for students to participate in studies carried out by researchers. A student can decide whether or not to participate in a study, as well as only allowing a subset of the answers to be linked.

The instructor starts by posting a questionnaire to the QS. After this, the ER has access to the questionnaire and decides whether to include it in his study. If that is the case, the ER informs the QS. Before submitting her response to the questionnaire, the student is shown the possibility of participating or not in the study. If there are several studies interested in the same questionnaire, the student can decide which ones to participate in. This happens for every questionnaire, always giving students the option to include their answers in each study.

In the case where a student decides to include her answer in a study, she sends along with it a pair of values we call *linking information*. This pair depends on the user's secret key, a study identifier, and a random value. If correctly generated it is always different and cannot be used on its own to link answers. When linking information is appended to an answer we say the answer is *link ready*.

Complementary to the linking information there exists another value we call the *linking token*. It depends on the study identifier and the user's secret key, not having any random components. When combined with an answer's linking information pair, this token allows one to identify all the answers which were made by the student who generated the token, but only among link ready answers in the same study.

At any moment a student can send (through an anonymous channel) the linking token to the respective ER. This triggers all link ready answers (made by the student in the ER's study) to become linkable – this is what allows us to achieve two-phase linking authorization. The student makes the first phase of authorization when she creates the linking information pair and sends it to the QS. The second phase is sending the linking token to the researcher, which is what actually triggers the linking. Finally, it is by the nature of the linking scheme that we guarantee user-controlled linkability.

H. Summary

We now present the full scenario. Figure 4 shows the complete procedure for a user answering a questionnaire. Here we assume the user is already authenticated with the eLP and that she has already obtained the credentials from the issuer. The user (1) requests the eLP page and (2) the eLP responds. The user then (3) requests to access the questionnaire page through a link in the eLP page – this translates into a request to the QS. (4) The QS responds and the student receives the page where the questionnaire can be answered. After answering, the user (5) is prompted about which studies to participate in, retrieves the credential from local storage, generates and appends the necessary linking information and signs the answer. The answer is then (6) sent to the QS through an anonymous channel.

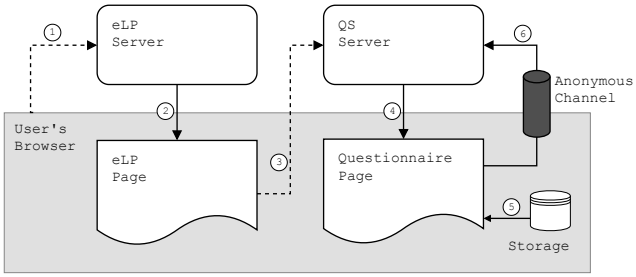


Fig. 4: Answering a questionnaire.

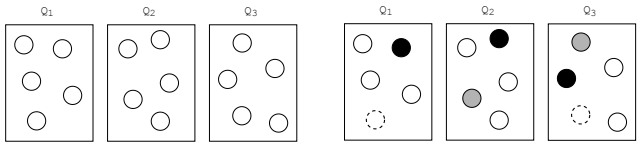
Now we go over all security properties we set out to guarantee, and summarize how we managed to accomplish that for each one.

- *Student anonymity*: we introduce an anonymous communication channel between the student and the QS, for the student to send her answers. Additionally, we assume the QS is a covert adversary, instead of arbitrarily malicious.
- *Answer unlinkability*: by guaranteeing anonymity we also manage to ensure unlinkability. Additionally, the introduction of attribute-based credentials, due to their unlinkable nature, does not break this property.
- *User-controlled linkability*: the cryptographic linking scheme we developed achieves exactly this.
- *Two-phase linking authorization*: our linking scheme achieves this with the phases for linking – the first decision to include the linking information pairs, leaving

answers link ready, and then the linking token, that actually provokes linkability.
- *Answer authenticity*: by introducing the credential scheme and signing of the answers, we are able to guarantee that answer authenticity can be verified, and thus guaranteed.
- *One-time participation*: the introduction of pseudonyms ensures that multiple participations in the same questionnaire by the same student are detectable.

V. Linking Scheme

In this chapter we present the scheme we designed to enable cryptographic linking of answers. The idea is that a student can allow a researcher to link some of her answers, while remaining anonymous. The student always has the final say about which of her answers become linkable. In Figure 5 we present an example with five students and three questionnaires, and show what happens when three of the students allow the linking of some of their answers. Each circle represents one student's answer to a questionnaire. In Figure 5a we have the view of the general parties in the system, where answers are anonymous and unlinkable. Given this view it is impossible to link answers across questionnaires. In Figure 5b we see the view of a researcher who was given permission to link some of the answers. The researcher can link all answers by one of the students (black colored circles), and link two answers of another two students (grey colored and dotted outline circles).



(a) General view (b) Researcher view

Fig. 5: An example of our linking scheme with five students and three questionnaires.

Another particularity of our scheme is that the linking happens in two phases. First the student decides whether an answer will be linkable, in which case she makes the answer ready to link and posts it – at this point the answer in not yet linkable. Later, the student decides if the set of answers that were made ready to link actually becomes linkable or not.

We dedicate a chapter to this scheme as it is one of our major contributions. In the remainder of the chapter we present the participating parties in the scheme and then define the scheme formally.

A. Parties

The scheme consists of any number of students and researchers, and a single questionnaire system.

- *Student*: answers questionnaires and decides on the linkability of her own answers.
- *Education Researcher (ER)*: is able to link students' answers if given permission by the student.
- *Questionnaire System (QS)*: the entity that aggregates all the information regarding students and researchers. Stores the questionnaires, the student's answers to them, as well as all other necessary information for the right operation of the protocols, such as the linking information and proofs.

B. Scheme

A *study* is a set of questionnaires of which a researcher wants to be able to link the answers. The same ER might conduct several different studies, but without loss of generality we will assume we have a different ER for each study.

To achieve selective linkability we use a pair of values we call *linking information*. We first roughly describe the main idea. Every time a student answers a questionnaire, they append, for each study containing the questionnaire, the linking information. On its own this pair does not allow any kind of linkability, however, if a student reveals to an ER the *linking token* corresponding to a certain study the researcher can then link the student's answers within that study. If the token is transmitted anonymously, the student manages to stay anonymous, as neither the linking token nor the linking information are sufficient to identify the student.

This scheme was for the most part based on Luek's work on Vote to Link [6]. We now explain our scheme in detail.

a) Setting: Let $(\mathbb{G}_1, \mathbb{G}_2)$ be a bilinear group pair of prime order p, $\hat{e} : \mathbb{G}_1 \times \mathbb{G}_2 \to \mathbb{G}_T$ the corresponding bilinear map and $g \in \mathbb{G}_1, h \in \mathbb{G}_2$ generators of each group. Let $H : \{0,1\}^* \to \mathbb{G}_1$ be a cryptographic hash function that maps strings to group elements, and *answer* $\in \{0,1\}^*$ represent the encoding of an answer to a questionnaire. Let x be the user's secret key and $C(x)$ the user's credential over x.

b) Answering a Questionnaire: Let $\texttt{qID} \in \{0,1\}^*$ denote the unique identifier of the questionnaire, and $\mathcal{S}$ the set of studies interested in qID in which the student wants to participate. First, the student computes the pseudonym for the questionnaire, $nym = H(\texttt{qID})^x$, and creates a zero knowledge proof π of the credential and correctness of the pseudonym, where

$$\pi = \text{SPK}\big\{(C, x) : C(x) \wedge nym = H(\texttt{qID})^x\big\}(answer).$$

Then, for each study $k \in \mathcal{S}$, the student picks a random $z_k \in \mathbb{Z}_p$ and computes $s_k = h^{z_k}$ and $t_k = \hat{e}\big(H(\texttt{studyID}), s_k\big)^x$. The pair $(s_k, t_k) \in \mathbb{G}_2 \times \mathbb{G}_T$ is the *linking information*. Finally, the student creates a signature proof of knowledge π_k for every linking information pair in order to prove that it is correctly formed. The proof also includes the showing of the credential and proves correctness of the pseudonym. It is as follows,

$$\pi_k = \text{SPK}\big\{(C, x, z_k) : C(x) \wedge nym = H(\texttt{qID})^x \wedge s_k = h^{z_k} \wedge t_k = \hat{e}\big(H(\texttt{studyID}_k), s_k\big)^x\big\}(answer),$$

where $\texttt{studyID}_k \in \{0,1\}^*$ is the unique identifier of study k. The student then sends the answer record $\alpha = \big(answer, nym, \pi, (\pi_k, s_k, t_k)_{k \in \mathcal{S}}\big)$ to the QS. Upon reception the QS verifies if π is valid and if the the pseudonym has

not been seen before. If both are true, the QS stores α. When retrieving an answer record α from the QS, an ER (conducting a study k) only stores α if π_k is valid.

c) Enabling Linking: The student computes the *linking token* $L = H(\texttt{studyID})^x$ and sends it to the ER conducting the study with identifier `studyID`. This communication should be done over an anonymous and encrypted channel. The former guarantees that the answers are only linked to each other and not to the student. The latter avoids the linking being done by other parties besides the intended ER. This token makes all student's answers associated with the study linkable. Note also that once the token is released by the student, all subsequent answers in the same study will be immediately linkable.

d) Linking Answers: Upon receiving a linking token L, an ER can go through all the answers obtained from the QS and check which ones are associated with the linking token. To do so, the ER extracts from an answer record the linking information, (s, t), relative to the ER's study. Then, an answer is associated to a linking token T if

$$\hat{e}(L, s) = t.$$

The ER can then conclude that all answers associated with a token L were made by the same student. An ER could use an associative array to group answers by the linking token associated with them. In this case, the linking token works as a pseudonym for the student within the study.

VI. Implementation

In this chapter we show the details of our implementations of the linking and credential schemes. Furthermore, we explain the architecture of our demo, give also some implementation details, and give details on its operation.

We implemented all the code in either JavaScript or TypeScript, as our objective was to create a system that would work on the browser. As all modern web browsers natively support the execution of JavaScript, ours was an easy decision.

The snippets of code shown in this chapter may have been redacted in order to reduce verbosity and improve readability. The full code can be found online at our GitHub repository (https://github.com/nowitworks/masters).

A. Credential Scheme

In this section we present the implementation of the PS Credentials scheme [3]. We implemented it in TypeScript and named it `ps-credentials`. Our code was written as a general purpose library, so that the implementation of the scheme could be used for other applications besides ours.

As the scheme requires it, we used the pairing-based cryptography library `mcl-wasm`. We note that, as the `mcl-wasm` library uses elliptic curves, it was developed with additive notation in mind for group operations (as is most common with elliptic curve cryptography), instead of the multiplicative notation we used in formalizing the scheme. Therefore, the methods from the library allude to this notation, as well as our code and comments.

The code is divided in three main classes: `Issuer`, `User`, and `Verifier`. Each class represents the respective entity from the scheme.

1) Setup: The `setup` function initializes the `mcl-wasm` library. The caller inputs which elliptic curve should be used, otherwise curve `BN254` is selected by default.

2) Key Generation: When instantiating an `Issuer` object, the constructor instantiates another object of the class `KeyPair`. This class represents the issuer's public and secret key and contains all elements making up the key pair. Both the secret and public keys are accessible individually so that the public key can be shared with other parties.

3) Issuing a Credential: First we must define the user and issuer determined attributes, and then instantiate the `User` and `Issuer` classes and retrieve the issuer's public key which we assume is available to all parties. The user triggers the beginning of the protocol by creating a commitment on the attributes and then sending it to the issuer. The issuer then blindly signs the commitment. The issuer sends the blind signature to the user, who unblinds it and checks if the resulting one is valid. The signature at the end of the protocol, `sig`, is the credential.

4) Showing a Credential: After obtaining the credential `sig`, the user can show it to a verifier. The protocol is straightforward: the user creates a proof of knowledge of the signature over her attributes and sends it to the verifier, who can check its validity.

5) Pseudonyms: In our implementation of the credential scheme we readily implemented the possibility to couple showing a credential with proving correctness of a pseudonym. We provide a method in the user class to create this proof, and another in the verifier class to assess its validity.

A pseudonym is created using a key and a domain. The domain string is mapped to an element of $\mathbb{G}_1$ and the key string to $\mathbb{Z}_p$, using respectively `mcl.hashAndMapToG1` and `mcl.hashToFr`, so that the pseudonym can then be correctly computed.

6) Internal Module: In addition to the API we described so far, we also developed an additional module, called `ps-internal`, to allow for extensions to the credential scheme. This module exposes the more low-level methods used throughout our code, mainly containing the necessary building blocks to construct zero-knowledge proofs related to the credentials and pseudonyms. This way, we can create extended proofs involving the credentials in other projects without having to build them from scratch.

B. Linking Scheme

In this section we describe our implementation of the scheme we designed to enable cryptographic linking of answers in the context of anonymous questionnaires.

We used the credential library we implemented, `ps-credentials`, in particular the low-level module `ps-internal`. We also made use of the `mcl-wasm` library for additional pairing based computations not covered by the credential library.

The code is divided in two classes, `Student` and `Verifier`, respectively representing the roles of the student and education researcher (ER). We did not create a questionnaire system (QS) class as it can be seen in this context as a verifier, from the credential scheme.

1) Answering a Questionnaire: The `tagAnswer` method in the `Student` class encapsulates the generation of all necessary information to correctly answer a questionnaire. The parameter `answerObj` is the object representation of the student's answer to the questionnaire with identifier `questionnaireID`. The array of strings `studyIDs` is expected to include all the identifiers of the studies in which the student wants to participate with that answer. The method thus creates the pseudonym, the linking information pairs for each study, and the respective proofs of knowledge (which always include showing the credential).

In the `Researcher` class we have method `checkAnswerCorrectness`, which checks the validity of the credential, pseudonym, and linking information.

2) Enable Linking: To generate the linking token the student calls the method `getLinkToken`, which takes as input solely the study identifier for which the student wishes to generate the token. It also uses the student's key, which is a property of the `Student` class.

3) Linking Answers: After collecting several answers and linking tokens, the researcher tries to establish wish answers are linked to each other. We do this by grouping answers by the linking token they are associated to.

C. System Demo

In this section we present the demo we built as a proof of concept of our system. It exemplifies the use of the credential and linking scheme, and instantiates our system design, while addressing the challenge of limiting user software to her web browser.

The demo is made up of four main components. Each of them was implemented as a server and they are as follows:

- `moodle`: represents the eLearning platform. We chose Moodle as it is the course management system used at EPFL, and it also supports LTI. We did not implement this server, but instead used a ready to use Docker image.
- `cred`: is the server hosting the credential and linking library, as well as all pages that contain code that directly manipulates the user's credential.
- `issuer`: represents the issuer from the credential scheme, and is thus responsible for issuing credentials to users.
- `qs`: the questionnaire system which hosts the questionnaires

The user is represented by the web browser, thus we need not implement a component that represents this entity.

The three servers (`cred`, `issuer`, and `qs`) were implemented using the Express framework in JavaScript and are run using Node.js. In the following sections we develop the explanation of each component, and we describe the protocol for parent-iframe communication.

1) moodle: For instantiating the eLearning platform server we used a Docker image containing a Moodle instance. This way we did not have to go through the trouble of setting up a Moodle server from scratch.

Once the instance is running we can access it locally through the web browser. We then play the role of the instructor to create and setup a course page, and give students access to it by enrolling them in the course. We can then add an external resource using LTI, so that users can access the issuance page. The security of the LTI authentication relies on a key shared between Moodle and the tool provider, which we input when setting up the LTI tool on Moodle. Afterwards, a link becomes available on the course page through which users can then access the resource.

We then also add in the course page several links to each one of the questionnaires, so that the students can access them. These questionnaires are hosted on the `qs` server (see section VI-C4).

2) cred: Named for its focus on the user's credential, this is one of the most critical components, as it provides users with the necessary code for their secure participation in the system. To guarantee that the parties in the system have no access to a student's credential, we introduced the `cred` server as the only fully trusted entity – this is an imported assumption due to `localStorage`'s vulnerability to XSS attacks. Users trust this server to provide a page – `user.html` – that accesses and does computations on the credential (using script `user.js` along with our credential and linking library), but never exposes it to other parties. This page is then included inside an iframe by the `issuer` and `qs`, so that they can also interact with the credential. As the credential is stored in the `localStorage` from `cred`'s domain, no other party's scripts have direct access to it.

The `user.html` page also contains the implementation (`parent_comm.js`) of the communication protocol for the iframe to be able to interact with the parent page (see Section VI-C5).

The server also hosts the `helper.js` script, which implements the parent side of the communication protocol, so that the `issuer` and `qs` can interact with the credential page in the iframe.

3) issuer: Implements a tool provider from the LTI standard. The issuance page can only be accessed if through LTI, which works as the authentication mechanism. To implement the tool provider we used JavaScript library `ims-lti`, which provides an intuitive interface to carry out the LTI authentication protocol with Moodle.

Once a student has gained proper access to the issuance page, they engage in the issuance protocol. If run successfully, the student should at the end be in possession of a valid signature. The issuer computations are done on the server side and using our implementation of the credential and linking schemes.

4) qs: Hosts questionnaires, and stores all information related to them – students' answers, pseu-donyms, linking information pairs, and related proofs. It starts by serving the

questionnaire page to the user, who then answers the questions. Pressing the submit button triggers the page to send the answer to the credential iframe, as well as the list of studies interested in the questionnaire. The iframe then outputs the pseudonym, linking information pairs, and the proofs. This is sent through an anonymous channel to the server who then checks the proof before storing the data.

Furthermore, for students to send the linking tokens to researchers, the `qs` hosts a page showing several links. Each link corresponds to a researcher, and clicking the link triggers the generation of the linking token, which is then sent to the researcher via an anonymous channel.

The anonymous channels we mentioned above were implemented using *Lightnion*[1], a JavaScript library developed at EPFL's Spring Lab which provides a lightweight method of making web requests through the Tor network, while using a regular web browser, there being no need for the installation of any additional software.

For the purposes of this demo, we did not find it necessary to create different components for the instructor or researchers. Instead we simulate the necessary instructor and researcher actions in the `qs` server. This reduces the complexity of the demo, without compromising its validity. Namely, we create and host the questionnaires directly on the `qs` server, as we did not implement a feature to enable instructors to create questionnaires. Additionally, we perform the simulated linking of answers on the `qs`, as well as receiving the linking tokens from students. Even though important in a production context, these are aspects we do not consider fundamental in a proof-of-concept scenario, in which we want only to demonstrate the core functionality of the system we designed.

5) Parent-iframe communication: Implemented in the scripts `helper.js` and `parent_comm.js`, we developed a protocol for the communication between a page inside an iframe and the parent page. The pages communicate using the postMessage interface. Note that we often abuse language and refer to the `user.html` page (which always appears inside an iframe) simply by *iframe*.

It starts with the parent page sending a request to the iframe for some value involving the credential. The iframe then prompts the user for permission before doing any computations. If the user allows, the iframe performs the necessary computations using the credential, and returns the resulting value to the parent page. We have a predefined set of actions the parent page can ask the iframe to perform, therefore protecting the credential by not allowing arbitrary (and possibly security-compromising) operations to be executed on it.

For clarity, the `parent_comm.js` script allows the following operations to be triggered on the browser: for the issuance protocol the creation of the initial commitment, unblinding of the signature, and then its storage; creating the "showable" credential; and signing answers, which includes creating pseudonyms, linking information pairs, and respective proofs.

[1]https://github.com/spring-epfl/lightnion – accessed 30-October-2019

VII. Conclusion

In this document we present a browser-based privacy-friendly system that enables users to answer online questionnaires anonymously. The student's several answers are unlinkable by the instructor. We introduce the role of the education researcher who asks students for permission to link some of their answers. The student then decides which of these requests to approve. The approved requests make the corresponding answers linkable by the researcher.

We explain the design of this system by addressing the several challenges we had to overcome to guarantee our security properties. Namely, we address how to guarantee that students stay anonymous, and also unlinkable if they choose to. Additionally, we ensure the authenticity of answers (an issue that often plagues anonymous systems), without compromising anonymity and unlinkability, by introducing an anonymous credential scheme. This scheme then works as one of the foundations for the cryptographic scheme we designed to allow the selective linking of answers, controlled by the user, and featuring a two-phase linking authorization mechanism.

Furthermore, we describe the implementation of the credential scheme, built as a general purpose TypeScript library, which can be used in other completely unrelated applications. Similarly, we present the implementation of our linking scheme. These implementations were then used to instantiate our system design in a proof-of-concept demo of the core functionality of the system.

References

[1] "Regulation (EU) 2016/679 of the European Parliament and of the Council of 27 April 2016 on the protection of natural persons with regard to the processing of personal data and on the free movement of such data, and repealing Directive 95/46/EC (General Data Protection Regulation)," *Official Journal of the European Union*, vol. L119, pp. 1–88, May 2016.

[2] J. Camenisch and M. Stadler, "Efficient group signature schemes for large groups (extended abstract)," in *Advances in Cryptology – CRYPTO'97*, ser. Lecture Notes in Computer Science, B. S. Kaliski Jr., Ed., vol. 1294. Santa Barbara, CA, USA: Springer, Heidelberg, Germany, Aug. 17–21, 1997, pp. 410–424.

[3] D. Pointcheval and O. Sanders, "Short randomizable signatures," in *Topics in Cryptology – CT-RSA 2016*, ser. Lecture Notes in Computer Science, K. Sako, Ed., vol. 9610. San Francisco, CA, USA: Springer, Heidelberg, Germany, Feb. 29 – Mar. 4, 2016, pp. 111–126.

[4] A. Fiat and A. Shamir, "How to prove yourself: Practical solutions to identification and signature problems," in *Advances in Cryptology – CRYPTO'86*, ser. Lecture Notes in Computer Science, A. M. Odlyzko, Ed., vol. 263. Santa Barbara, CA, USA: Springer, Heidelberg, Germany, Aug. 1987, pp. 186–194.

[5] E. F. Brickell, J. Camenisch, and L. Chen, "Direct anonymous attestation," in *ACM CCS 04: 11th Conference on Computer and Communications Security*, V. Atluri, B. Pfitzmann, and P. McDaniel, Eds. Washington D.C., USA: ACM Press, Oct. 25–29, 2004, pp. 132–145.

[6] W. Lueks, M. H. Everts, and J. Hoepman, "Vote to link: Recovering from misbehaving anonymous users," in *Proceedings of the 2016 ACM on Workshop on Privacy in the Electronic Society, WPES 2016, Vienna, Austria, October 24, 2015*. ACM, 2016, pp. 111–122.

[7] Y. Aumann and Y. Lindell, "Security against covert adversaries: Efficient protocols for realistic adversaries," in *TCC 2007: 4th Theory of Cryptography Conference*, ser. Lecture Notes in Computer Science, S. P. Vadhan, Ed., vol. 4392. Amsterdam, The Netherlands: Springer, Heidelberg, Germany, Feb. 21–24, 2007, pp. 137–156.