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

João Pedro Pinto Silveira

Thesis to obtain the Master of Science Degree in

Computer Science and Engineering

Supervisor(s): Prof. Miguel Nuno Dias Alves Pupo Correia
Dr. Wouter Lueks

Examination Committee
Chairperson: Prof. Alexandre Paulo Lourenço Francisco
Supervisor(s): Prof. Miguel Nuno Dias Alves Pupo Correia
Member of the Committee: Prof. Ricardo Jorge Fernandes Chaves

December 2019
Acknowledgments

I will start by thanking everyone who reads this thesis. I hope by the end you will have learned something interesting.

Now I especially want to thank Wouter, for teaching me so many interesting things and always guiding me in the right direction. It was always fun and extremely rewarding to work with you, and you exceeded my expectations of how cool a supervisor can be. I also want to thank Prof. Carmela Troncoso for allowing me to do my thesis at the SPRING lab, where people are doing amazing and inspiring research to help create a better and safer world.

I would like to thank Prof. Miguel Correia, who gave me most valuable feedback and support. Your help was essential.

Finally, I definitely have to thank my family and friends. You kept me motivated and helped me through all of it, making it completely impossible to imagine any of it without you.
Resumo

A utilização de ferramentas on-line em salas de aula, como questionários e quizzes, está a tornar-se cada vez mais popular. Isto deve-se à facilidade de uso, versatilidade e capacidade para envolver os alunos. No entanto, a quantidade de dados privados recolhidos por esses sistemas, e a recente introdução de regulamentos como o RGPD, suscitam algumas preocupações. Neste trabalho apresentamos um sistema de questionários completamente anónimo para browsers. Para além disso, investigadores podem pedir aos alunos permissão para relacionar um subconjunto das suas respostas aos questionários. Por sua vez os alunos podem aceitar ou rejeitar esses pedidos. Descrevemos o design deste sistema e respetiva implementação numa demonstração de funcionalidade. Introduzimos ainda um esquema criptográfico de relação controlada de respostas a questionários, com um mecanismo de autorização de relações em duas fases. Finalmente, descrevemos as implementações desse sistema criptográfico e do esquema de credenciais anónimas que o suporta.

Palavras-chave: anonimato, privacidade, relação controlada por utilizador, credenciais anónimas, criptografia.
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.

Keywords: anonymity, privacy, user-controlled linkability, attribute-based credentials, cryptography.
## Contents

Acknowledgments .................................................. v  
Resumo ........................................................... vii  
Abstract ......................................................... ix  
List of Tables ..................................................... xv  
List of Figures ................................................... xvii

1 Introduction  
1.1 Motivation ................................................... 1  
1.2 Contributions ............................................... 2  
1.3 Thesis Outline ............................................. 3

2 Background  
2.1 Groups ....................................................... 5  
2.2 Pairings ....................................................... 6  
2.3 Cryptographic Assumptions ................................ 7  
2.4 Zero-Knowledge Proofs of Knowledge ..................... 8  
2.5 Attribute-Based Credentials ................................ 11  
2.6 Anonymous and Confidential Channels ................... 13

3 Related Work  
3.1 PS Credentials ............................................. 16  
3.1.1 Parties .................................................. 16  
3.1.2 Scheme .................................................. 17  
3.2 Group Signatures .......................................... 18  
3.3 Direct Anonymous Attestation ............................. 19  
3.4 Conditional Linkability ................................... 21  
3.5 Anonymous blacklisting ................................... 22  
3.6 Discussion .................................................. 25
4 System Design

4.1 Parties ........................................... 27
4.2 Security Properties ................................. 28
4.3 Base Scenario .................................... 29
4.4 Achieving Anonymity .............................. 30
  4.4.1 Application-level Anonymity ................. 30
  4.4.2 Network-level Anonymity ..................... 31
  4.4.3 Timing Attacks ................................. 31
4.5 Achieving Authenticity ......................... 32
  4.5.1 Learning Tools Interoperability ............ 33
  4.5.2 Issuing Credentials .......................... 33
4.6 Pseudonyms ................................... 34
4.7 Achieving Linkability ............................ 35
4.8 Summary ...................................... 36

5 Linking Scheme .................................. 39

5.1 Parties ........................................... 40
5.2 Scheme ........................................ 40

6 Implementation ................................... 43

6.1 Credential Scheme ............................... 43
  6.1.1 Setup ....................................... 44
  6.1.2 Key Generation ............................... 44
  6.1.3 Issuing a Credential ......................... 44
  6.1.4 Showing a Credential ......................... 45
  6.1.5 Pseudonyms ................................ 45
  6.1.6 Internal Module .............................. 46
6.2 Linking Scheme ................................ 46
  6.2.1 Answering a Questionnaire .................. 47
  6.2.2 Enable Linking ............................... 47
  6.2.3 Linking Answers ............................. 48
6.3 System Demo .................................... 48
  6.3.1 Browser Security ............................ 48
  6.3.2 Storing Credentials ......................... 49
  6.3.3 Architecture ................................ 50
List of Tables

3.1 Comparison of related work. .................................................. 25

6.1 Comparison of storage technologies in the browser. ....................... 55

7.1 Sizes of the elements in our credential and linking schemes. ............. 57
7.2 Sizes in bytes of the elements from the mcl-wasm library. ............... 58
7.3 Sizes in bytes of the elements stored in the user’s browser. ............. 58
7.4 Running times of the libraries’ most relevant operations. ................. 59
# List of Figures

<table>
<thead>
<tr>
<th>Figure</th>
<th>Description</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>2.1</td>
<td>Sigma protocol where Peggy proves to Victor the knowledge of $x$.</td>
<td>9</td>
</tr>
<tr>
<td>2.2</td>
<td>Sigma protocol where Peggy proves to Victor the knowledge of $x$ and $y$.</td>
<td>10</td>
</tr>
<tr>
<td>4.1</td>
<td>Base interaction scenario for answering a questionnaire.</td>
<td>29</td>
</tr>
<tr>
<td>4.2</td>
<td>Introducing an anonymous channel for students to send answers to the QS.</td>
<td>32</td>
</tr>
<tr>
<td>4.3</td>
<td>Issuing credentials.</td>
<td>34</td>
</tr>
<tr>
<td>4.4</td>
<td>Answering a questionnaire.</td>
<td>36</td>
</tr>
<tr>
<td>4.5</td>
<td>Sequence diagram showing the interaction between the system’s several parties.</td>
<td>38</td>
</tr>
<tr>
<td>5.1</td>
<td>An example of our linking scheme with five students and three questionnaires.</td>
<td>39</td>
</tr>
<tr>
<td>6.1</td>
<td>Code example of credential issuance protocol.</td>
<td>45</td>
</tr>
<tr>
<td>6.2</td>
<td>Methods involved in showing a credential.</td>
<td>46</td>
</tr>
<tr>
<td>6.3</td>
<td>Signature of method $\text{tagAnswer}$.</td>
<td>47</td>
</tr>
<tr>
<td>6.4</td>
<td>Signature of method $\text{checkAnswerCorrectness}$.</td>
<td>47</td>
</tr>
<tr>
<td>6.5</td>
<td>System design for credential issuance in the browser.</td>
<td>54</td>
</tr>
</tbody>
</table>
Chapter 1

Introduction

1.1 Motivation

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 and quizzes – for example Kahoot\textsuperscript{1} or Socrative\textsuperscript{2}. Students can easily answer on their phones or laptops questions proposed by the instructor, who gets the results instantly and can then adjust the flow of class accordingly. These tools can be used in multiple ways and by anyone at any time, turning out to be quite versatile and thus widely used.

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. In solutions of this kind, users can answer questionnaires while remaining completely anonymous, making their progress over time practically impossible to track. This scenario might be acceptable for instructors, who are often only interested in the performance of the class as a whole, and do not need to be able to track individual student behaviour.

However, when introducing education researchers into such a privacy-enhancing system, their objectives become rather impossible to achieve. Education researchers want to evaluate how the system is affecting students and understand how they evolve over time. To do this, one would need the ability to understand that two or more answers were produced by the same student – this is called linking \[2, 3\]. In our context, as the privacy of users must be preserved, education

\textsuperscript{1}https://kahoot.com/ – accessed 30-October-2019

\textsuperscript{2}https://www.socrative.com/ – accessed 30-October-2019
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 linking of results does not come without its drawbacks. The excessive linking of answers over time could amount to a comprehensive profiling of each student, possibly resulting in the student’s de-anonymization. Therefore, measures should be taken to prevent said situation.

This topic is also relevant in the broader context of internet privacy, where users are becoming increasingly more aware of the information gathered about them online, and how to take measures to protect their privacy. However, internet users often do not feel in control of the data they are sharing with organizations, which is also what prompted policymakers to implement such regulations as the previously mentioned GDPR.

In this work we present an anonymous questionnaire system that allows for selective linkability of answers. The student always remains anonymous and has full control over which answers are linkable by researchers. As one of the main reasons for the popularity of these questionnaire tools is their ease of use, we made our solution completely browser-based, which means that a student does not need to install any additional software on their device, being a common web browser sufficient for participation in the system.

This work was developed at the Security and Privacy Engineering (SPRING) lab at the École Polytechnique Fédérale de Lausanne (EPFL), and in cooperation with Instituto Superior Técnico (IST). I worked under the supervision of Dr. Wouter Lueks (from EPFL), and Prof. Miguel Pupo Correia (from IST).

1.2 Contributions

Our main contribution is the design of a browser-based anonymous questionnaire system, that allows for user-controlled linkability of answers to education researchers. 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 system 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.

³https://www.epfl.ch/labs/spring/
1.3 Thesis Outline

The remainder of the document is structured as follows. Chapter 2 introduces the cryptographic tools and other background needed to understand the constructions presented in this work. In Chapter 3 we present and describe previous work related to our topic. Chapter 4 details the overall architecture of our system, as well as the motivation behind our design choices. Chapter 5 presents the cryptographic linking scheme we designed. In Chapter 6 we explain how we implemented the credential and linking schemes, as well as the system demo. Chapter 7 presents several measurements and experimentally obtained results to assess the performance of the implemented system. Finally, Chapter 8 presents the conclusions we reached upon the completion of this work, and some ideas for future work on the thesis topics.
Chapter 2

Background

This section presents background information on cryptographic topics that are necessary for this work. For a more extensive treatment of these sections, we refer to Katz and Lindell [4] and Smart [5].

The section is organized as follows. We start by introducing groups in Section 2.1, and pairings in Section 2.2. We then follow with the necessary cryptographic assumptions in Section 2.3. We give an overview of zero-knowledge proofs in Section 2.4 and attribute-based credentials in Section 2.5. Finally, we define anonymous and confidential communication channels in Section 2.6.

2.1 Groups

We briefly introduce the notion of groups here and refer to Katz and Lindell [4] for a more detailed introduction.

Definition 2.1. A group is a tuple \((G, \circ)\) where \(G\) is a set and \(\circ\) is an operation that maps \(G \times G\) to \(G\), such that it satisfies the following properties:

1. (Closure) For all \(g, h \in G\), \(g \circ h \in G\).

2. (Existence of an identity) There exists an element \(e \in G\), such that for all \(g \in G\), \(g \circ e = e \circ g = g\).

3. (Existence of inverses) For all \(g \in G\) there exists an element \(b \in G\) (inverse of \(g\)), such that \(g \circ b = b \circ g = e\).

4. (Associativity) For all \(a, b, c \in G\), we have \((a \circ b) \circ c = a \circ (b \circ c)\).

We usually refer to a group \((G, \circ)\) simply as \(G\). The order of a group \(G\), denoted by \(|G|\), is the number of elements in the group. A group with finite order is called a finite group. Typically
groups are of prime order $p$. We will often use $\mathbb{Z}_p$ to denote the integers modulo $p$. We now define abelian groups.

**Definition 2.2.** A group $G$ is abelian if it satisfies the following additional property:

5. (Commutativity) For all $g, h \in G$, we have $g \circ h = h \circ g$.

In this work we use multiplicative notation for the group operation, thus we write $1$ for the identity element. We call a group element $g \in G$ a generator of $G$ if for all $h \in G$, we can write $h = g^i$ for some $i \in \mathbb{Z}$. If such a generator exists for a group $G$, we say $G$ is cyclic. A cyclic group is necessarily abelian. Now we define group homomorphism and isomorphism.

**Definition 2.3.** Given two groups $(G, \circ)$ and $(H, \triangleleft)$, we say a function $f : G \rightarrow H$ is a group homomorphism if, for all $g, g' \in G$,

$$f(g \circ g') = f(g) \triangleleft f(g').$$

Additionally, if $f$ is also a bijection, we call $f$ a group isomorphism.

To help us define the cryptographic assumptions of Section 2.3, we use a polynomial-time algorithm $G$ that generates a description of a group. On input $1^\ell$, $G$ outputs a 3-tuple $(G, p, g)$ which is a description of a group $G$, of order $p$, generated by $g$, and where $p$ is of size $\ell$ bits.

### 2.2 Pairings

We give here a short introduction to pairings, presenting the most important facts and definitions. We refer to Galbraith, Paterson and Smart [6] for an in-depth explanation.

Let $(G_1, G_2)$ be a pair of cyclic groups, both of prime order $p$, and $g, h$ be generators of $G_1$ and $G_2$ respectively. Let $G_T$ be a group of prime order $p$, which we will call the target group.

**Definition 2.4.** We say $(G_1, G_2)$ is a bilinear group pair if there exists a bilinear map $\hat{e} : G_1 \times G_2 \rightarrow G_T$ such that,

- $\hat{e}$ is bilinear: $\hat{e}(g^a, h^b) = \hat{e}(g, h)^{ab}$ for all $a, b \in \mathbb{Z}_p$;

- $\hat{e}$ is non-degenerate: $\hat{e}(g, h)$ is a generator of $G_T$;

- $\hat{e}(x, y)$ can be computed efficiently (i.e., in polynomial time) for all $x \in G_1$ and $y \in G_2$.

This map is typically called a pairing.

There exist four types of pairings. We present the first three here as they are the most common in the context of pairing-based cryptography.
Type 1 \( G_1 = G_2 \).

Type 2 \( G_1 \neq G_2 \) but there exists an efficiently computable homomorphism \( \phi : G_2 \to G_1 \).

Type 3 \( G_1 \neq G_2 \) and there exist no efficiently computable homomorphisms between groups \( G_1 \) and \( G_2 \).

Similar to what we did for cyclic groups, we define algorithm \( IG \) that on input \( 1^{\ell} \), outputs the tuple \( (\hat{e}(\cdot, \cdot), G_1, G_2, G_T, p, g, h, g_T) \), describing pairing \( \hat{e} : G_1 \times G_2 \to G_T \), bilinear group pair \( (G_1, G_2) \) with generators \( g \) and \( h \), target group \( G_T \) generated by \( g_T \), and the \( \ell \) bit group order \( p \).

2.3 Cryptographic Assumptions

Most of the works presented here rely on certain cryptographic assumptions which are based on the hardness of mathematical problems. These assumptions are the basis of most cryptographic systems used today and allow us to create provably secure systems, as long as the assumptions hold. Once again we refer to Katz and Lindell [4] for a more comprehensive explanation of this topic.

We start by giving the intuition about the first two assumptions, then we define negligible function as it will be useful to formally define the assumptions, and finally we present the definition of the assumptions.

The discrete logarithm assumption says that given a group element \( h = g^x \) and generator \( g \), it is difficult to find the exponent \( x \). The decisional Diffie-Hellman assumption intuitively states that given \( g^x \) and \( g^y \) it is difficult to determine if a third element is of the form \( g^{xy} \).

**Definition 2.5.** A function \( f : \mathbb{N} \to \mathbb{R}_+^+ \) is called negligible if for every positive polynomial \( p \), there exists an \( N \) such that, for all \( n > N \), \( f(n) < p(n)^{-1} \).

**Definition 2.6.** The discrete logarithm (DL) problem in a cyclic group \( G \) is to output \( x \) on input \((X, g) \in G^2\), where \( g \) is a generator of \( G \) and \( X = g^x \).

**Definition 2.7.** We say the DL problem is hard relative to \( G \) if there exists a negligible function \( \text{negl} \) such that, for all probabilistic polynomial-time (PPL) algorithms \( A \),

\[
\Pr[x \leftarrow A(G, p, g, X) \land X = g^x] \leq \text{negl}(\ell)
\]

where the probability is over the output \((G, p, g)\) of \( G(1^{\ell}) \), the random choice of \( X \in G \) and the random bits of algorithm \( A \). We further say that the DL assumption holds if there exists a \( G \) for which the DL problem is hard.
Definition 2.8. The decisional Diffie-Hellman (DDH) problem in a cyclic group $G$ is to output 1 (resp. 0), on input $(g, X, Y, Z) \in G^4$, if $z = xy$ (resp. $z \neq xy$), where $X = g^x$, $Y = g^y$ and $Z = g^z$.

Definition 2.9. We say the DDH problem is hard relative to $G$ if there exists a negligible function $\text{negl}$ such that, for all PPL algorithms $A$,

$$\left| \Pr[A(G, p, g, g^x, g^y, g^z) = 1] - \Pr[A(G, p, g, g^x, g^y, g^x) = 1] \right| \leq \text{negl}(\ell)$$

where the probabilities are over the output $(G, p, g)$ of $G(1^\ell)$, the random choices of $x, y, z \in \mathbb{Z}_p$ and the random bits of algorithm $A$. We say the DDH assumption holds if there exists a $G$ for which the DDH problem is hard.

Finally we define the $q$-strong Diffie-Hellman assumption. This assumption is commonly used in systems built on top of pairings. The following definitions are due to Boneh and Boyen [7].

Definition 2.10. The $q$-Strong Diffie-Hellman ($q$-SDH) problem in a bilinear group pair $(G_1, G_2)$, with respective generators $g$ and $h$, and both groups of prime order $p$, is defined as follows. On input of tuple $(g, g^x, g^{x^2}, ..., g^{x^q}, h, h^x) \in G_1^{q+1} \times G_2^2$, output pair $(c, g_c^{x+c}) \in \mathbb{Z}_p \times G_1$ for a freely chosen value $c \in \mathbb{Z}_p \{−1\}$.

Definition 2.11. We say the $q$-SDH problem is hard relative to $IG$ if there exists a negligible function $\text{negl}$ such that, for all PPL algorithms $A$,

$$\Pr[A(\gamma, g^x, g^{x^2}, ..., g^{x^q}, h^x) = (c, g_c^{x+c})] \leq \text{negl}(\ell)$$

where the probabilities are over the output $\gamma = (\hat{e}(\cdot, \cdot), G_1, G_2, G_T, p, g, h, g_T)$ of $IG(1^\ell)$, the random choices of $x \in \mathbb{Z}_p$ and the random bits of algorithm $A$. We say the $q$-SDH assumption holds if there exists an $IG$ for which the $q$-SDH problem is hard.

2.4 Zero-Knowledge Proofs of Knowledge

Zero-knowledge proofs of knowledge were introduced by Goldwasser, Micali and Rackoff [8] in 1985. Using such a proof, a prover, often called Peggy, can convince a verifier, Victor, 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 Peggy indeed possesses these values and nothing else.
Zero-knowledge proofs are a very powerful tool. For example, Peggy could use it to prove to Victor that she knows the private key $x \in \mathbb{Z}_p$ corresponding to a public key $h = g^x$, where $g, h \in G$ and $G$ is a cyclic group of prime order $p$. Indeed, Victor should exclusively learn that Peggy knows this key and that it is valid. The sigma protocol in Figure 2.1 constitutes a zero-knowledge proof of such a statement.

To focus on the semantics of the protocol instead of its implementation, we introduce the following notation, adapted from Camenisch and Stadler [9], to represent zero-knowledge proofs in a more generic way:

$$\text{PK}\{(x_1, ..., x_n) : \text{conjunction and/or disjunction of statements about } x_1, ..., x_n\}$$

where $x_1, ..., x_n$ are secrets and any other variables used in the statements are assumed to be public.

We can also prove more complicated statements over groups. These constructions are frequently used by the related work, and in our solution as well. For the following examples we take as setting a cyclic group $G$ of prime order $p$.

For example, to instantiate

$$\text{PK}\{(x, y) : h = g_1^x \land C = g_2^x \cdot g_3^y\},$$

we can have the protocol in Figure 2.2.

In a zero-knowledge proof, the following properties should be satisfied [5]:

- Completeness: If the proven statement is true, then Victor should accept Peggy’s proof
Common knowledge: $G, p$ and $g_1, g_2, g_3, h, C \in \mathbb{G}$

<table>
<thead>
<tr>
<th><strong>Peggy</strong></th>
<th><strong>Victor</strong></th>
</tr>
</thead>
<tbody>
<tr>
<td>knows $x, y \in \mathbb{Z}_p$ such that $h = g_1^x$ and $C = g_2^y \cdot g_3^y$</td>
<td>only knows the common knowledge</td>
</tr>
</tbody>
</table>

pick random $k_x, k_y \in \mathbb{Z}_p$

$r_h := g_1^{k_x}$

$r_C := g_2^{k_x} \cdot g_3^{k_y}$

$(r_h, r_C)$

pick random $c \in \mathbb{Z}_p$

$c$

$s_x := k_x + cx$

$s_y := k_y + cy$

$(s_x, s_y)$

verify:

$g_1^{s_x} = h^c \cdot r_h$ and $g_2^{s_x} \cdot g_3^{s_y} = C^c \cdot r_C$

Figure 2.2: Sigma protocol where Peggy proves to Victor the knowledge of $x$ and $y$. with probability one.

- **Soundness**: If the statement is false, Victor should be convinced of Peggy’s proof only with small probability.

- **Zero-knowledge**: The verifier should learn nothing else from the proof besides its validity.

An important aspect of interactive zero-knowledge proofs is that the transcript of the proof cannot be used to prove the validity of the statement to other parties that did not participate in the protocol.

Any of these interactive zero-knowledge proofs can be transformed into non-interactive proofs using the Fiat-Shamir heuristic [10]. Roughly, the idea is to remove the role of the verifier by using the digest of a hash function as the challenge. This new value should depend on the random value generated in the beginning of the protocol and the public values. Assuming that the hash function is not invertible, this is important in order to guarantee the validity of the proof, otherwise it would be easy to prove false statements.

We now show an example of the non-interactive instantiation of $\text{PK}\{(x) : h = g^x\}$:

- **Proof generation**:

1. Select random $k$
2. $r := g^k$
3. \( c := H(h||g||r) \)

4. \( s := k + cx \)

5. Output \((c, s)\)

Verification:

1. \( r' := g^s \cdot h^{-c} \)

2. Verify: \( c = H(h||g||r') \)

Where \( H \) is a non-invertible hash function.

In contrast to the interactive version, non-interactive zero-knowledge proofs can be used to prove knowledge of a statement by simply showing the transcript of the protocol, as it can be verified by anyone. This is used to create a signature scheme by including a message in the input of the hash function. The signature proof of knowledge of \( x \) over the message \( m \), such that \( h = g^x \), is represented by

\[ \text{SPK}\{(x) : h = g^x\}(m). \]

We based this introduction on Smart’s book [5, Chapter 25], which we refer to for more details and examples.

### 2.5 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 the attributes which are necessary to grant access to the user. In fact, these credential systems can provide an even higher level of privacy. Instead of revealing the actual requested attributes, the user could simply reveal the necessary facts about those attributes. For example, when using a credential to prove that she is an adult, a user could just reveal the fact that her age is over 18, instead of revealing her actual age. The idea is to prove that the attribute in question satisfies a certain property. Here, zero-knowledge proofs are used as a building block.

In a credential we typically have the following roles [11, 12]:

- **Issuer**: a trusted entity that issues credentials to users. The Issuer is responsible for making sure that the credential it issues contains correct information.

- **User**: entity that keeps the credentials issued to her. These credentials might have been issued by several Issuers. They are used with a Service Provider in order to grant the
user access to a service, either by disclosing certain attributes or proving statements about them.

- **Verifier or Service Provider**: entity which is able to provide some service to an authorized user. It is responsible for requesting the necessary attribute disclosure or proof of statements to the user, and for their subsequent validation.

- **Revocation Authority**: Not always present in an ABC scheme. It is responsible for the revocation of credentials and for preventing the use of revoked credentials.

The following actions are commonly supported [12]:

- **Issuance**: The process through which the user receives a credential issued by an issuer. This might be preceded by a series of checks to guarantee the information contained in the credential is correct.

- **Presentation/Showing/Verification**: In this phase the verifier requests the disclosure of attributes by the user or the proof of statements about them. The verifier then chooses to grant or refuse the user access to the requested service.

- **Revocation**: One of the scheme’s participants requests the revocation of a credential. The revocation authority stores this information and makes sure it is available in order to prevent the future usage of the revoked credential. The revocation can be initiated by the user, *user-initiated revocation*, or the system, *system initiated revocation*.

A number of properties are usually required by systems that use ABCs. ABC schemes offer different properties depending on their implementation and the applications they are meant for. The properties we present next [11, Section 2.7] are very common and often essential in existing schemes:

- **Unforgeability**: It is not possible for any party in the system to forge a credential over a new key $x$ without the help of the issuer.

- **Unlinkability**: The showing of a credential gives no information about its owner to the verifier. In particular, when any two different users disclose the same set of attributes, a verifier cannot distinguish between the two users when she is shown a credential of one of them.

- **Zero-knowledge proofs**: The showing of a credential can be combined with a zero-knowledge proof that proves a statement about the secret key $x$ associated with the credential.
There exist several works on how to construct ABCs. Some of them include the blind signature-based U-Prove [13], Idemix [14] (which is based on zero-knowledge proofs) and the elliptic curve cryptography variant of Idemix [15]. One can also construct ABCs using a signature scheme such as BBS+ [16] and PS signatures [17], which we describe in Section 3.1. In the setting where user and verifier use a shared key, message authentication codes can also be used as the basis for the system [18].

2.6 Anonymous and Confidential Channels

Throughout this work we often mention anonymous and confidential communication channels. By anonymous channel we mean a channel that guarantees it is hard for the receiver of a message (or other adversary) to identify the sender of the message. Tor [19] is an example of a system that implements this type of channel. When referring to confidential channels we wish to ensure a communication where no other than the participants (sender and receiver) are able to decipher the contents of the sent message. As an example, HTTPS\textsuperscript{1} is a protocol that achieves this.

Chapter 3

Related Work

In this section we give an overview of some works related to our topic. We describe them and give some details about the constructions in order to show how the more relevant parts work and are specified. We do not give comprehensive descriptions and we refer to the original papers for the full details.

As a preliminary note, we define the concept of linkability \[2, 3\] as it is crucial to the understanding of the following sections. We say “items are linkable if they are more or less related than they are without any knowledge of the system” \[3\]. In the context of anonymous communication systems, if two messages are linkable, it means that one knows they were sent by the same user – regardless of whether this user remains anonymous. If a message is linkable to a user, that translates into the user losing her anonymity.

In this section, our objective is to understand which solutions currently exist and what forms of linkability they offer, i.e., what can be linked and which entities wield this power. Some of the schemes we present here offer a number of interesting properties, that even though relevant in their context, we overlook here. This allows us to focus on linkability which is our main concern.

The remainder of this section follows with the presentation of the ABC scheme we used in our work, in Section 3.1. We then give an introduction to group signatures (Section 3.2) and direct anonymous attestation schemes (Section 3.3). In Section 3.4 we introduce conditional linkability and describe Vote to Link, a scheme offering this type of linkability. In Section 3.5 we describe BLAC, a privacy-enhanced revocation scheme using blacklists. Finally, in Section 3.6, we shortly discuss and compare the presented schemes.
3.1 PS Credentials

As explained in Section 2.5, attribute-based credentials (or anonymous credentials) allow users to show a verifying entity that they have the necessary authorization to access a certain resource while maintaining their identity hidden and only revealing the absolutely necessary information. Additionally, the same credential can be used several times, while keeping the multiple showings unlinkable.

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 [17, Section 6], which we refer to as PS Credentials. We defer to Section 6.1 for the details on how it was implemented. We start by giving an intuitive high-level description of how the scheme works.

Let Alice be the user who wishes to obtain a credential, Ivan the issuer of credentials, and Victor the entity that verifies the validity of a credential (e.g. a service provider). Alice starts by asking Ivan to make a blind signature on a set of values – this means Alice gets the signature on the values without having to send them to Ivan. This is done by having Alice send a commitment on the values she wants signed and proving in zero-knowledge the correctness of the commitment. If Ivan is convinced of the proof, he blindly signs the commitment and sends the blind signature to Alice. She can now unblind the signature obtaining what we call the credential. Once Alice owns the credential, she can prove its possession to a verifier while remaining anonymous and unlinkable. To do so she starts by randomizing the credential and sending it to Victor. Additionally, she proves in zero-knowledge that the credential is well formed and that it was correctly signed with Ivan’s key (which is public). Now Victor can attest the authenticity of the credential by checking the validity of the proof.

The following sections explain in detail the Pointcheval and Sanders’ credential scheme [17, Section 6]. First we introduce the involved parties followed by the explanation of the scheme itself.

3.1.1 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.

3.1.2 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.

Setting Let \((G_1, G_2)\) be a bilinear group pair of prime order \(p\) and \(e: G_1 \times G_2 \to G_T\) 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.

Key Generation This algorithm generates a key pair for the issuer. It starts by randomly picking a generator of \(G_1\) and \(G_2\), respectively \(g \in G_1\) and \(\tilde{g} \in G_2\). 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 \(sk \leftarrow X\), and the public key is set as \(pk \leftarrow (g, Y_1, ..., Y_r, \tilde{g}, \tilde{X}, \tilde{Y}_1, ..., \tilde{Y}_r)\).

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 \(I \subseteq \{1, ..., r\}\) be the set of indices of attributes determined by the issuer, and \(U = \{1, ..., r\} \setminus I\) 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, \(C = g^t \prod_{i \in I} Y_i^{a_i}\). 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 \(\sigma'_2 = (XC \prod_{i \in I} Y_i^{a_i})^u\). The user obtains the signature by unblinding \(\sigma'\), 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

\[
e(c_1, \tilde{X} \cdot \prod_{i=1}^r \tilde{Y}_i^{a_i}) = e(c_2, \tilde{g})
\]

Showing a Credential This protocol is carried out between the user and the verifier. Let \(D \subseteq \{1, ..., r\}\) be the set of indices of attributes the user wants to disclose when showing the credential, and \(H = \{1, ..., r\} \setminus D\) 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 \( \sigma’ = (\sigma_1^r, (\sigma_2^t \cdot \sigma_1^r)^r) \). Then the user sends \( \sigma’ \) to the verifier, engages in the proof of knowledge

\[
\pi = \text{PK}\{(a_i \in \mathcal{H}, t) : e(\sigma_1', \tilde{X}) \cdot \prod_{i \in \mathcal{H}} e(\sigma_1', \tilde{Y}_i)^{a_i} \cdot \prod_{j \in \mathcal{D}} e(\sigma_1', \tilde{Y}_j)^{a_j} \cdot e(\sigma_1', \tilde{g})^t = e(\sigma_2', \tilde{g}) \}
\]

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 \( \pi \) is valid.

This type of credential can also be used as a signature over a message. To do this we just have to apply the Fiat-Shamir heuristic [10] (see Section 2.4) to the proof of knowledge \( \pi \) from the showing of a credential. In an attempt to improve readability in the other sections, we often use \( C(x) \) to denote a credential over a secret key \( x \). For example, the proof

\[
\text{PK}\{(C, x) : C(x) \land X = h^x \}
\]

proves knowledge of a credential \( C \) over a secret value \( x \), where this value was used to compute a known value \( X \) using a public element \( h \). This simplification also highlights the fact that our other constructions do not depend on this specific credential scheme, leaving the possibility to have it replaced by other schemes.

### 3.2 Group Signatures

The first type of linkability we look at is the one provided by *group signatures*, which, as introduced by Chaum and Heyst [20], are a signature scheme that guarantees the following basic properties:

1. only members of the group can sign messages;
2. a receiver can verify that a message’s signature is valid but cannot tell which member of the group signed the message;
3. under certain circumstances, the signature can be “opened” in order to reveal the identity of the signer.

Thus, group signatures provide privacy in the sense that users are anonymous within the group they are a part of. However, as we can see by the third property, users can be fully de-anonymized, which is the strongest form of linkability. The power to undo this anonymity
usually belongs to a special party called the tracing agent. Thus, we see that privacy here fully depends on the trustworthiness of the tracing agent.

However, group signatures can be constructed in order for the power to reveal the identity of a signer to be distributed [21]. In these systems, several group members have to cooperate in order to recover the identity of a signer. The security of the scheme now depends on the number of malicious tracing agents.

Recent work [22] has introduced in the context of group signatures for the possibility of creating fully unlinkable signatures at first, which can then become linkable later. However, this construction relies on a central entity wielding the power to link.

We refer to Manulis et al. [23] for a in-depth overview of group signatures.

3.3 Direct Anonymous Attestation

Direct Anonymous Attestation (DAA) was presented by Brickell et al. [24] 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. Finally, it allows for the identification of signatures that were produced with known keys.

There have been several publications improving on the original scheme by Brickell et al. [24], mostly focused on updating its security model [25, 26, 27] and efficiency [28, 29]. However, none of them seem to introduce the ability for two-phase linking. Thus, we will focus on describing the original simpler scheme, as it is enough to demonstrate the idea of the family of schemes.

The context of the scheme is a trusted platform module (TPM) who wants to authenticate to a verifier while remaining anonymous. The TPM has limited resources but is included in a host machine, thus delegating most computations to the host. We will give an informal explanation of the scheme, without going into mathematical detail. First, the system’s participants are:

– the Trusted Platform Module (TPM) is the central entity in the scheme. It would correspond to a user in the context of a credential scheme, for example. The TPM wishes to authenticate to a verifier by proving that it received a signature from the issuer;

– the host is the machine in which the TPM is embedded;

– the issuer is responsible for issuing the signature that the TPM uses for authentication;

– and finally, the verifier is the entity responsible for checking TPMs’ signatures.

The scheme is mainly made up of the following protocols and algorithm:
**Join**  This protocol involves the TPM, its host and the issuer, and consists of the TPM asking the issuer to sign a pair of secret values. The TPM, with the help of the host, creates a commitment on the secret values and sends it to the issuer, while also proving in zero knowledge that the commitment is correct. The issuer then checks the proof, and if convinced creates a signature over this commitment – i.e., a blind signature over the secrets (see Section 3.1.2 for an example on how to create blind signatures). The issuer sends the signature to the TPM and proves that it is correctly formed. If the proof is correct, the TPM unblinds the signature and stores it.

**Sign**  Run between the TPM and the host, this protocol is used to create a “showable” signature derived from the one obtained with the *Join* protocol. Essentially, this means that the TPM and the host create a proof of knowledge of a valid signature made by the issuer. This proof is then showed as a form of authentication to the verifier who checks its validity by running the *Verify* algorithm. The created signature also includes a *pseudonym*. This pseudonym can be used to link several different authentications of the same TPM, but without requiring the TPM to give up anonymity, or the linking of all its authentications. In a cyclic group setting, the pseudonym is created by first computing a group generator \( \zeta = H(\text{bsn}) \), where \( \text{bsn} \) is a basename, and \( H \) is a cryptographic hash function that maps strings to group elements. Then, the TPM computes the pseudonym \( N = \zeta^f \), where \( f \) is the user’s secret. This way, we can see that if the TPM generates two different signatures using the same basename, the pseudonym \( N \) is the same, and therefore the two signatures become linkable. If TPMs wish to remain unlinkable, they simply select a random \( \zeta \) instead of computing it using the basename.

**Verify**  The verification algorithm is run by the verifier and takes as input the signature generated using with the *Sign* protocol by the TPM and its host. The verifier then uses the issuer’s public key to check the validity of the signature. The verifier also checks if the pseudonym is correctly formed and if it has been used before – making sure it has not been created using a key extracted from a malicious TPM.

We draw attention to the use of pseudonyms in this scheme. One of our objectives is to allow the users of the system to choose which of their answers are linkable. This is precisely what is achieved in this scheme by giving TPMs the choice to use the same pseudonym to enable linkability, and to compute the pseudonym with the random generator to remain unlinkable. This is indeed an effective method to achieve this. However, this would imply that a user’s answer becomes immediately linkable once it is retrieved by the researcher, as the pseudonym enables
linkability right away. As we wish to provide users with a two-phase control over their answers’ linkability, this solution is not adequate. A possibility would be to encrypt the pseudonyms (using a non-deterministic encryption scheme such as El-Gamal) with a different key for every researcher, and later disclosing these keys to the respective researchers. We will see in Chapter 5 that our solution follows a similar idea.

3.4 Conditional Linkability

In revocation schemes for group signatures we see that usually, when a user is revoked membership, all past actions become linkable. To address this excessive linking, Nakanishi and Funabiki [30] introduced a revocation scheme for group signatures with backward unlinkability. In this scheme, time is divided into time intervals and users who are revoked membership become linkable only from the interval in which their access was revoked - all their past actions before this remain unlinkable. To this purpose the scheme uses tickets which are created to allow linking of actions.

We call this type of linkability, conditional linkability. The term comes from the fact that even though the linking of a user’s actions is possible, it only happens in certain conditions. In this case, linkability is only possible after a specific time, whereas all previous actions remain ununlinkable.

We now describe Vote to Link [31], a more complex scheme drawing inspiration from Nakanishi and Funabiki’s work [30]. The basic idea is to have a system where users can ask Service Providers (SPs) to perform transactions. These transactions are anonymous and underlying is a group signature scheme. However, if a user is acting maliciously, moderators can intervene. The scheme works in such a way that, 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 [31] deals with conditional linkability. This work uses a variant of the Shoup and Gennaro’s $\text{TDH2}$ threshold encryption scheme [32]. Here we give only a brief high level description. We define $\text{TDH.Enc}(w, r, \tau)$ as the encryption of linking token $r$, labeled with transition $\tau$ with the moderators’ public key $w$ using a $k$-out-of-$n$ threshold encryption scheme. Briefly, this scheme enables the encryption of a message which can only be decrypted if $k$ or more moderators (corresponding to the given public key) participate in the decryption.
Setting  Let \((G_1, G_2)\) be a bilinear group pair of prime order \(p\), \(\hat{e} : G_1 \times G_2 \to G_T\) the corresponding bilinear map and \(g \in G_1, h \in G_2\) generators of each group. Let \(x\) be the user’s secret key and \(H : \{0,1\}^* \to G_1\) a cryptographic hash function. \(\epsilon\) is a description of an epoch, and \(H\) is used to transform \(\epsilon\) into a group generator - \(H(\epsilon)\) is a generator of \(G_1\).

Linking token  A user’s linking token for epoch \(\epsilon\) 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.

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 \(T = \text{TDH.Enc}(w, r, \tau)\) 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 \(\tau\):

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

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 \(\pi\) is correct.

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 \(\epsilon\). To do so, for every transaction \(t' = (\tau', T', t'_1, t'_2, \epsilon, \pi')\) in epoch \(\epsilon\), the SP knows that \(\tau'\) was performed by the same user if

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

3.5 Anonymous blacklisting

We present under this section the notion of anonymously blacklisting misbehaving users. The goal is to revoke access to users that misbehave, but still learning nothing about these users. The revocation entity is not able to identify the user or link any of her messages. When the focus of the main system is privacy, this is a very relevant property.

Next, we describe one of these systems called BLAC [33] and we refer to Henry and Goldberg [34] for more examples and analysis of these systems.
Blacklistable Anonymous Credentials (BLAC) [33] enables revocation of access to repeatedly misbehaving users without de-anonymizing them. This is a property the authors refer to as privacy-enhanced revocation. Revocation happens after misbehaviour has been detected a $d$ number of times. The systems does not rely on a trusted third party who has the power to arbitrarily de-anonymize users. Also, misbehaviour can be judged subjectively. That is why threshold-based approaches - such as double spending in cryptocurrencies - are not suitable for this kind of system.

BLAC requires an underlying credential scheme that must allow revocation of credentials such that:

1. the anonymity of users is preserved,
2. the revocation can be based on subjective definitions of misbehavior,
3. does not rely on a trusted third party.

We now give a brief overview of the way BLAC works. Users are issued credentials by an entity called the Group Manager (GP). The users have a unique secret credential, known only by the user. Users anonymously authenticate to Service Providers (SPs) who will serve users that have not misbehaved more than $d$ times. To verify this, the SP keeps a blacklist with each entry representing an instance of misbehaviour. These entries are called tickets and the SPs cannot link them to users, therefore not being able to learn the identity of misbehaving users. Thus the user has to prove to the SP that she appears in the blacklist fewer than $d$ times, otherwise authentication fails. When authentication succeeds, a ticket is produced. This ticket is used to tag actions done by the user during the authenticated session. The ticket will be different the next time the user authenticates, therefore not allowing the SP or other entities to link between actions by the same user across different authentication sessions. This ticket can then be used to label an action as misbehaviour by simply adding the ticket to the blacklist. Furthermore, the blacklisting can be done long after the misbehaviour took place and can be judged subjectively. Also, SPs can decide to remove tickets from the blacklist as way to forgive past misbehaviour.

A ticket is a tuple $(s, t)$, where $s$ is randomly picked by the user and $t = H(s||\text{sid})^x$, where $\text{sid}$ is the target server’s identity, $x$ is from the user’s credential and $H$ is a secure cryptographic hash function.

We now give a more detailed explanation of the different phases of the system.

Parameters  Let $\lambda$ and $\ell$ be sufficiently large security parameters. Let $(G_1, G_2)$ be a bilinear group pair, with both groups of prime order $p$ (of $\lambda$ bits) and computable isomorphism $\psi$. Let
$g_0, g_1, g_2 \in G_1$ and $h_0 \in G_2$ be generators of $G_1$ and $G_2$ respectively such that $g_0 = \psi(h_0)$. We assume the relative discrete logarithms of the generators is unknown. We also have $G$, a group of prime order $p$ where DDH problem is hard. Finally, let $H_0 : \{0,1\}^* \to G$ and $H : \{0,1\}^* \to \mathbb{Z}_p$ be secure cryptographic hash functions.

**Setup**  The GM randomly generates $\gamma \in \mathbb{Z}_p$ and computes $w = h_0^\gamma$. The group secret key is $gsk = (\gamma)$ and the group public key is $gpk = (w)$.

**Registration**  Let Alice be a user registering in the group. The input for the GM is $gsk$. Below is the protocol for authentication:

1. The GM sends a random challenge $m \in \{0,1\}^\ell$ to Alice.

2. Alice sends the GM a pair $(C, \Pi_1)$. $C$ is a commitment on two randomly chosen values $z, y' \in \mathbb{Z}_p$ and $\Pi_1$ is the signature proof of knowledge

$$\text{SPK}\left\{(x, y') : C = g_1^x g_2^y \right\}(m).$$

3. If the verification of $\Pi_1$ fails, the GM aborts. Otherwise, the GM sends Alice the tuple $(A, e, y'')$, where $A = (g_0Cg_2^{y''})^{\frac{1}{\gamma}} \in G_1$ and $e, y'' \in \mathbb{Z}_p$ are randomly selected.

4. Alice computes $y = y' + y''$. If $\hat{e}(A, wh_0^\gamma) = \hat{e}(g_0g_1^yg_2^{y'}, h_0)$, Alice outputs the $\text{cred} = (A, e, x, y)$, otherwise she aborts.

**Authentication**

1. The SP sends Alice the tuple $(BL, m, d)$, where $BL$ is the blacklist, $m \in \{0,1\}^\ell$ is a random challenge, and $d$ is the threshold value.

2. Alice computes all $b_i = H_0(s_i||\text{sid})$ for $i = 1, \ldots, n$. If there are $d$ or more distinct values of $i$ for which $t_i = b_i^e$, Alice aborts. Otherwise, the protocol continues.

3. Alice sends the SP the pair $(\tau, \Pi_2)$. Here, $\tau = (s, t)$ is a ticket, where $s \in \{0,1\}^\ell$ is random, and $t = b^e$ with $b = H_0(s||\text{sid})$. $\Pi_2$ is the signature proof of knowledge

$$\text{SPK}\left\{(A, e, x, y) : A^{e+\gamma} = g_0g_1^yg_2^y \land t = b^e \land \left(\bigvee_{I \subseteq (n-d+1)[n]} \bigwedge_{i \in I} t_i \neq b_i^e \right) \right\}(m).$$
4. If the verification of $\Pi_2$ succeeds, the user successfully authenticates. Otherwise, the user fails to authenticate.

We refer to the original paper [33, Section 6] for the instantiation details of the protocols for the signature proofs of knowledge mentioned above. To conclude, we refer to variations of BLAC [35, 36, 37] that improve on the original construction and extend its functionality.

### 3.6 Discussion

In this section we make a short comparison between the presented solutions. In Table 3.1 we identify the type of linking each solution allows, as well as the entity that is able to do this linking.

<table>
<thead>
<tr>
<th>Solution</th>
<th>Type of Linking</th>
<th>Linking Entity</th>
</tr>
</thead>
<tbody>
<tr>
<td>Group Signatures [20]</td>
<td>Reveal identity of signer</td>
<td>Tracing Agent</td>
</tr>
<tr>
<td>DAA [24]</td>
<td>User controls which signatures are linkable</td>
<td>Verifier</td>
</tr>
<tr>
<td>Backward Unlinkability [30]</td>
<td>Link all transactions within epoch and all subsequent ones</td>
<td>Moderators</td>
</tr>
<tr>
<td>Vote to Link [31]</td>
<td>Link all transactions within epoch but never across epochs</td>
<td>Moderators</td>
</tr>
<tr>
<td>BLAC [33]</td>
<td>No linking whatsoever</td>
<td>—</td>
</tr>
</tbody>
</table>

We used Vote to Link [31] as the main inspiration for the cryptographic linking scheme we designed and present in Chapter 5. We take the logic of the epochs and apply it to studies. This way we can keep answers linkable within the context of a study. The linking token and auxiliary values are also used, and whenever a student answers a questionnaire, the auxiliary values are sent along with the answer. Later the linking token (this time revealed by the student at will) is what gives researchers the ability to link users’ answers.
Chapter 4

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.

4.1 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.

### 4.2 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 question-
naire at most once.

4.3 Base Scenario

Now that we have introduced the parties in our system we can introduce our basic interaction
scenario. In Figure 4.1 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.

Figure 4.1: Base interaction scenario for answering a questionnaire.

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.
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.

4.4 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 [38]. 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.

4.4.1 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 4.1 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.
4.4.2 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 4.2 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 4.4.1 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 4.2, 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 4.2, step 3). This way the QS never sees the user’s IP and thus cannot break unlinkability 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.

4.4.3 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 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.

### 4.5 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 (see Section 2.5), which satisfy these requirements. We refer to Chapter 3.1 for the description of the concrete scheme we used in this work.

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.

4.5.1 Learning Tools Interoperability

The Learning Tools Interoperability (LTI)\(^1\), designed by IMS Global\(^2\), is a standard aimed at facilitating and improving the security in the connection between Learning Management Systems (LMS) – e.g. Moodle\(^3\) or FenixEdu\(^4\) – 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 [39]. The standard describes two entities [40]: the Tool Consumer (TC), which is the LMS; and the Tool Provider (TP), representing the external resource. The specification [41] 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.

4.5.2 Issuing Credentials

Figure 4.3 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

\(^1\)https://www.imsglobal.org/activity/learning-tools-interoperability
\(^2\)https://www.imsglobal.org
\(^3\)https://moodle.org/ – accessed 30-October-2019
\(^4\)https://fenixedu.org/ – accessed 30-October-2019
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.

![Figure 4.3: Issuing credentials.](image)

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.

### 4.6 Pseudonyms

Drawing inspiration from DDA [28] and their use of pseudonyms for linkability, 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 $G_1$ be a cyclic group of prime order $p$, and $H : \{0,1\}^* \rightarrow G_1$ a cryptographic hash function that maps strings to group elements. We compute the pseudonym as

$$nym = H(qID)^x,$$
where $q\text{ID} \in \{0, 1\}^*$ 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

$$\pi = \text{SPK}\left\{ (C, x) : C(x) \land nym = H(q\text{ID})^x \right\}(answer),$$

where $C(x)$ denotes a credential over the secret key $x$. Adding $C(x)$ to $\pi$ 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 $\pi$. 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.

### 4.7 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.

We refer to Chapter 5 for a detailed description of the full scheme, which also includes the
use of pseudonyms.

4.8 Summary

We now present the full scenario. Figure 4.4 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.

![Diagram of the answering scenario](image)

**Figure 4.4: 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 unlikability**: by guaranteeing anonymity we also manage to ensure unlikability. Additionally, the introduction of attribute-based credentials, due to their unlikable nature, does not break this property.

- **User-controlled linkability**: the cryptographic linking scheme we developed (see Chapter 5) 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.

Finally, Figure 4.5 summarizes the chapter by presenting all interaction in the several phases the system goes through.
Figure 4.5: Sequence diagram showing the interaction between the system’s several parties.
Chapter 5

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 5.1 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 5.1a 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 5.1b 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).

Figure 5.1: 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.

5.1 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.

5.2 Scheme

As described in the beginning of Chapter 4, 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 Luck’s work on Vote to Link [31] (see Section 3.4 for a summary). We now explain our scheme in detail.

**Setting**  Let $(G_1, G_2)$ be a bilinear group pair of prime order $p$, $\hat{e} : G_1 \times G_2 \rightarrow G_T$ the corresponding bilinear map and $g \in G_1, h \in G_2$ generators of each group. Let $H : \{0, 1\}^* \rightarrow G_1$ 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 \( x \).

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

\[
\pi = \text{SPK}\{ (C, x) : C(x) \land nym = H(qID)^x \}\text{(answer)}.
\]

Then, for each study \( k \in S \), the student picks a random \( z_k \in \mathbb{Z}_p \) and computes \( s_k = h^{z_k} \) and \( t_k = \hat{e}(H(\text{studyID}_k), s_k)^x \). The pair \((s_k, t_k) \in G_2 \times G_T\) is the linking information. Finally, the student creates a signature proof of knowledge \( \pi_k \) 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,

\[
\pi_k = \text{SPK}\{ (C, x, z_k) : C(x) \land nym = H(qID)^x \land s_k = h^{z_k} \land t_k = \hat{e}(H(\text{studyID}_k), s_k)^x \}\text{(answer)},
\]

where \( \text{studyID}_k \in \{0, 1\}^* \) is the unique identifier of study \( k \). The student then sends the answer record \( \alpha = (\text{answer}, nym, \pi, (\pi_k, s_k, t_k)_{k \in S}) \) to the QS. Upon reception the QS verifies if \( \pi \) is valid and if the the pseudonym has not been seen before. If both are true, the QS stores \( \alpha \).

When retrieving an answer record \( \alpha \) from the QS, an ER (conducting a study \( k \)) only stores \( \alpha \) if \( \pi_k \) is valid.

**Enabling Linking** The student computes the linking token \( L = H(\text{studyID})^x \) and sends it to the ER conducting the study with identifier \( \text{studyID} \). 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.

**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.
Chapter 6

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). The repository contains three directories: ps-credentials, where our credential library can be found; linking, with the implementation of our linking scheme; and demo, with all the files that constitute our system demo.

6.1 Credential Scheme

In this section we present the implementation of the credential scheme we described in Chapter 3.1. We implemented it in TypeScript⁵ and named it ps-credentials after the authors David Pointcheval and Olivier Sanders of the original scheme [17]. 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-wasm⁶, which seemed to be the most well maintained and appropriate JavaScript library in this topic. In our code, any methods, variables, or types prefixed with “mcl.” are from the mcl-wasm library. 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.

### 6.1.1 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.

### 6.1.2 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. The following code:

```javascript
const issuer: Issuer = new Issuer(r);
const pk: PublicKey = issuer.publicKey;
```

is an example of how to instantiate an **Issuer** object, taking as input the value \( r \), which is
the number of attributes the credential carries. It also shows how to access the public key (the
secret key is a private property only accessible inside the class’s methods).

### 6.1.3 Issuing a Credential

In Figure 6.1 we show an example using our code for the issuance of a credential. The
method calls are prefixed with either “**issuer.**” or “**user.**”, which means the code should
be run respectively on the issuer’s or the user’s machine. First we define the user and issuer
determined attributes (lines 1 and 2), and then we instantiate the **User** and **Issuer** classes (lines
3 and 4) and retrieve the issuer’s public key (line 5) which we assume is available to all parties.
The user triggers the beginning of the protocol by creating a commitment on the attributes (line
7) and then sending it to the issuer. The issuer then blindly signs the commitment (line 10)
– this method throws an exception in case the commitment is not valid. The issuer sends the
blind signature to the user, who unblinds the signature (line 13) and checks if the resulting one
is valid (line 14). The resulting signature at the end of the protocol, \( \text{sig} \), is the credential.
const userAttr: string[] = ["attr1", "attr2"];
const issuerAttr: string[] = ["attr3"];
const issuer: Issuer = new Issuer(3);
const user: User = new User();
const pk: PublicKey = issuer.publicKey;
const commitment = user.createCommitment(pk, userAttr);
// User ---------------------- {commitment} ----------------------> Issuer
const blindSignature: Signature = issuer.blindSign(commitment, issuerAttr);
// Issuer ------------------ {blindSignature} ----------------------> User
const sig: Signature = user.unblindSignature(blindSignature);
verify(pk, userAttr.concat(issuerAttr), sig);

6.1.4 Showing a Credential

After obtaining the credential $\text{sig}$, 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.

To create the proof, the user runs the method proveKnowledgeOfSignature, of which we show the method signature in Figure 6.2 (line 4). Argument attributes (line 7) should contain all of the user’s attributes that were signed by the issuer, while hiddenSetIdxs (line 8) is the set of indexes of the attributes the user wishes to keep hidden while showing the credential.

Upon being shown the credential, the verifier checks its validity by calling the method in line 15, checkSignatureProof. The attributes parameter in line 18 expects a list of the attributes the user wants to disclose, while the positions on the list of the attributes that are to remain hidden should be replaced by a null value. In both lines 6 and 17, publicKey refers to the issuer’s public key.

6.1.5 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 the method in the user class to create this proof, proveKnowledgeOfSignatureAndNym, and another in the verifier class, checkSignatureProofAndNym, to assess the validity of the proof.

A pseudonym is created using a key and a domain (see Section 4.6 for more details). The following function is invoked by the user to create the pseudonym object,
public createDomainSpecificNym(domain: string, keyStr: string): Pseudonym

The domain string is mapped to an element of $G_1$ and the key string to $Z_p$, using respectively `mcl.hashAndMapToG1` and `mcl.hashToFr`, so that the pseudonym can then be correctly computed.

### 6.1.6 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. This was particularly useful in implementing our linking scheme (see Section 6.2).

### 6.2 Linking Scheme

In this section we describe our implementation of the scheme we designed (see Chapter 5) to enable cryptographic linking of answers in the context of anonymous questionnaires.
We used the credential library we implemented (see Section 6.1), \texttt{ps-credentials}, in particular the low-level module \texttt{ps-internal}. We also made use of the \texttt{mcl-wasm} library for additional pairing based computations not covered by the credential library.

The code is divided in two classes, \texttt{Student} and \texttt{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.

6.2.1 Answering a Questionnaire

The \texttt{tagAnswer} method in the \texttt{Student} class encapsulates the generation of all necessary information to correctly answer a questionnaire. Figure 6.3 shows the method’s signature. The parameter \texttt{answerObj} is the object representation of the student’s answer to the questionnaire with identifier \texttt{questionnaireID}. The array of strings \texttt{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 \texttt{Researcher} class we have method \texttt{checkAnswerCorrectness} (see signature in Figure 6.4), which checks the validity of the credential, pseudonym, and linking information.

6.2.2 Enable Linking

To generate the linking token the student calls the method \texttt{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 \texttt{Student} class.
6.2.3 Linking Answers

After collecting several answers and linking tokens, the researcher tries to establish which answers are linked to each other. We do this by grouping answers by the linking token they are associated to.

First we introduce three auxiliary data structures:

- **answers** is an array of **Answer** objects. Each of these objects contains not only the actual answer to the questionnaire, but also the questionnaire identifier, as well as the linking information;

- **tokens** is the array containing all retrieved linking tokens;

- and **links** is an associative array where each key is a string representation of a linking token, and the values are arrays of **Answer** objects. This is where we ultimately group answers by their corresponding linking token.

Once the **answers** and **tokens** arrays are populated, the researcher goes through the answers and for each answer, tests if any of the tokens is associated with the answer’s linking information. If that is the case, the answer is added to the token’s entry in **links**.

6.3 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.

6.3.1 Browser Security

In this section we present some topics on browser security. This information will help us motivate some of the decisions we made on the construction of our demo.

**Subresource Integrity (SRI)** A browser feature that allows a page to ask the browser to check the integrity of a fetched resource. This means that the browser will fetch the resource, and before it is used by the page, it checks if the cryptographic hash of the resource matches the one indicated in the page that wishes to load it. This is particularly important when using **localStorage** (see section 6.3.2), as an imported script will have access to the same **localStorage** variable as the code in the page or in any other imported script. With SRI the developer can make sure that the imported code was not tampered with [42].
Origin  The origin of a page represents the domain from which the page was loaded. The origin is determined by part of the page’s URL, which is of the form protocol://domain:port/path. To determine if two pages have the same origin, their URLs must have exactly the same protocol, domain, and port, otherwise their origins are considered distinct. For example, the URLs https://main.example.org and https://sub.example.org have different origins as their domain fields are different. On the other hand, https://sub.example.org/contact and https://sub.example.org/about/team have the same origin. Note that in the previous examples the port is implicitly 80 as it is the browser default when port is omitted [43].

Same-origin policy  A mechanism that dictates how a page or script from a certain origin can interact with resources from another origin. It is important to master the concept of origin and its correct definition, so as to guarantee the security of a page [44].

Inline Frame Element (iframe)  An iframe allows an HTML page to be embedded inside another web page. The embedded page obtains a private context, not shared with the parent page. This includes its origin, so that the parent page cannot access sensitive information, such as cookies or localStorage [45].

postMessage Interface  The postMessage interface is aimed at providing a parent page with a method to communicate with its child pages. For example, it allows a page to communicate with the iframe it is embedding, and vice versa. When the Window.postMessage method is called, the caller sends a message to the destination page. This triggers an event on the receiving page, who is free to handle it in any desired way [46].

6.3.2 Storing Credentials

The two alternatives for storing the credentials in the context of web browsers are server-side and client-side storage. As evidenced by our system architecture, we focused on client-side storage. The main reason behind this decision is to guarantee unlinkability and anonymity. The credential and linking schemes rely on the secrecy of the credential and secret key to guarantee their properties. When sharing these values with a possible adversary, the security of the scheme is easily put in jeopardy. However, we recognize that there might be solutions using server-side storage that could still satisfy our desired properties. Unfortunately their complexity would go beyond the scope of this project and thus we decided for a client-side solution which is secure under our threat model.
In terms of client-side storage we had to analyse the several options offered by web browsers. The several technologies are described below.

**JavaScript variables** Simply using JavaScript variables is in theory a safe option, however, once the user closes or refreshes a page, or navigates to a different one, all data is lost. This is clearly an unacceptable solution as we need the credentials to persist across websites. For example, we might have the credential issued on a page hosted by a credential issuer, and then we want to be able to use this credential in a page hosted by a service provider for authentication.

**HTML Cookies** Cookies are bits of information stored on the user’s browser that can be set by servers or by scripts. Cookies are sent to the server with every request, which is not an ideal scenario to help maintain the credential a secret. However, cookies provide the `httpOnly` flag, which is a mechanism that helps in the protection against cross-site scripting attacks. Cookies set with this flag are not accessible by scripts running on the user’s browser. However, this mechanism is not useful for our system, as we need the user to be able to run scripts using the credential – e.g. to compute the showing of a credential [47].

**Web Storage** An option similar to cookies but that is not automatically transmitted with every request. Web Storage also provides larger storage size than cookies. There are two types of Web Storage: `localStorage`, which is shared by all scripts in the same origin (and across tabs) and persists across sessions even after the browser is closed; and `sessionStorage` which does not persist across sessions and is shared only within the same tab. Although vulnerable to XSS attacks, `localStorage` seems to be the more adequate solution [48].

**IndexedDB and Cache API** IndexedDB offers a database-like interface, while the Cache API provides a low level cache-like interface. Both these options offer more complex operations and storage space. However, we do not need either in comparison with the simpler Web Storage solution, as both these alternatives also provide the same security properties as Web Storage [49, 50].

### 6.3.3 Architecture

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,

- **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\footnote{https://expressjs.com/ – accessed 30-October-2019} framework in JavaScript and are run using Node.js\footnote{https://nodejs.org/ – accessed 29-October-2019}. In the following sections we develop the explanation of each component, and we describe the protocol for parent-iframe communication.

### 6.3.3.1 moodle

For instantiating the eLearning platform server we used a Docker image\footnote{https://hub.docker.com/r/bitnami/moodle – accessed 29-Oct-2019} 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\footnote{https://docs.moodle.org/36/en/External_tool_settings – accessed 30-October-2019}, 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 6.3.3.4).

### 6.3.3.2 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 6.3.3.5).

Ideally, to relax the trust assumption on the cred server, we could use the equivalent of subresource integrity (SRI) – which can be used for scripts – but for the page we wish to include in the iframe. Unfortunately, this type of SRI does not exist yet\(^8\).

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.

6.3.3.3 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\(^9\), 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.

6.3.3.4 qs

Hosts questionnaires, and stores all information related to them – students’ answers, pseudonyms, 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

---

\(^8\)There exists an open issue on the W3C GitHub page for the possible implementation of integrity enforcement for iframes. https://github.com/w3c/webappsec-subresource-integrity/issues/21 – accessed 29-October-2019

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*\(^{10}\), a JavaScript library developed at EPFL's Spring Lab which provides a lightweight method of making web requests through the Tor network\(^{11}\), 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.

### 6.3.3.5 Parent-ifame 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

\(^{10}\)https://github.com/spring-epfl/lightnion – accessed 30-October-2019

signature, and then its storage; creating the “showable” credential; and signing answers, which includes creating pseudonyms, linking information pairs, and respective proofs.

6.3.4 Summary

We summarize this section by showing an example. Figure 6.5 shows the architecture of our system for the credential issuance protocol. The issuer hosts the issuance page, which is rendered on the user’s browser. This page imports, from the cred server, the helper.js script, as well as the user.html page – which is embedded in an iframe. The user.html page uses the parent_comm.js and user.js scripts, and accesses its domain localStorage. The interaction between the issuance page and the iframe is done using the postMessage interface.

![Figure 6.5: System design for credential issuance in the browser.](image)

This structure is analogous for other protocols in the system. If we swap the issuer with the qs, we essentially have the setup for answering a questionnaire.
Table 6.1: Comparison of storage technologies in the browser.

<table>
<thead>
<tr>
<th>Technology</th>
<th>Who can access?</th>
<th>Vulnerabilities</th>
<th>Limitations</th>
<th>Compatibility</th>
</tr>
</thead>
<tbody>
<tr>
<td>JavaScript</td>
<td>All scripts imported in page/ All scripts with access to document variable</td>
<td>XSS attacks</td>
<td>Non-persistent; not available across page instances</td>
<td>All browsers with JavaScript support</td>
</tr>
<tr>
<td>Cookies</td>
<td>All scripts in same domain (follows same-origin policy)</td>
<td>XSS attacks (mitigated by using httpOnly flag)</td>
<td>httpOnly flag blocks script access, prohibiting use in local computations; sent to server in every request</td>
<td>All regular browsers</td>
</tr>
<tr>
<td>localStorage</td>
<td>All scripts in same domain (follows same-origin policy)</td>
<td>XSS attacks</td>
<td>no XSS protection</td>
<td>All regular browsers</td>
</tr>
<tr>
<td>sessionStorage</td>
<td>All scripts in same domain (follows same-origin policy)</td>
<td>XSS attacks</td>
<td>no XSS protection; not persistent across sessions</td>
<td>All regular browsers</td>
</tr>
<tr>
<td>IndexedDB</td>
<td>All scripts in same domain (follows same-origin policy)</td>
<td>XSS attacks</td>
<td>same as localStorage</td>
<td>Modern browsers</td>
</tr>
<tr>
<td>Cache API</td>
<td>All scripts in same domain (follows same-origin policy)</td>
<td>XSS attacks</td>
<td>same as localStorage</td>
<td>In experimental phase and unavailable in Internet Explorer</td>
</tr>
</tbody>
</table>
Chapter 7

Evaluation

Given the nature of this thesis, the evaluation section rarely was a focus of our work. Firstly, we did a mostly theoretical exercise in designing the system and cryptographic linking scheme. Secondly, our more practical exercise of implementation focused essentially on providing functionality, with no efficiency-focused goals.

However we believe that including some performance measurements can prove useful for possible future comparisons and analysis. Thus, in this section we produce some metrics for the evaluation of our work. In Section 7.1 we show the costs associated with the cryptographic linking scheme we designed, namely the sizes of the cryptographic elements. In Section 7.2 we present the running time of some of our tests, which represent some of the most relevant operations from the implementation of the linking and credential scheme.

7.1 Size of Cryptographic Elements

Table 7.1 shows the sizes of the cryptographic elements that make up the linking scheme we designed (see Chapter 5). For the size of the credential we consider the elements $\sigma_1$ and $\sigma_2$. We get the size of the linking information pair from the two values $s$ and $t$, while the pseudonym and the linking token are both single values, so we just show their respective sizes. We implemented zero-knowledge proofs using the Fiat-Shamir heuristic (see Section 2.4), thus their sizes depend

<table>
<thead>
<tr>
<th>Element</th>
<th>Size</th>
</tr>
</thead>
<tbody>
<tr>
<td>Credential</td>
<td>$2G_1$</td>
</tr>
<tr>
<td>Pseudonym</td>
<td>$1G_1$</td>
</tr>
<tr>
<td>Linking Information Pair</td>
<td>$1G_2 + 1G_T$</td>
</tr>
<tr>
<td>Linking Token</td>
<td>$1G_1$</td>
</tr>
<tr>
<td>Zero-knowledge Proof</td>
<td>$(s + 1)Z_p$</td>
</tr>
</tbody>
</table>

Table 7.1: Sizes of the elements in our credential and linking schemes.
on the number of secrets involved in the proof. In the table, we represent the number of secrets by $s$.

In Table 7.2 we show the sizes in bytes of the cryptographic elements implemented in the mcl-wasm library. These are the sizes of the objects that represent the elements, which do not use compression. The library was setup with the BN254 elliptic curve. Additionally, we show in Table 7.3 the sizes in bytes of the values stored in the user’s browser. The users’ attributes are also stored in their respective browsers – however, we do not show their sizes as they directly depend on the attributes themselves.

### 7.2 Running Times

In order to assess the correctness of our code during development, we wrote several unit tests to test its functionality using the Mocha\(^1\) test framework, along with the Chai\(^2\) assertion library. Among the credential and linking libraries, we wrote a total of 45 tests.

We ran several experiments to assess the running times of the most relevant operations from the implemented libraries. The results can be seen in Table 7.4. For this, the mcl-wasm library was set up to use the BN254 elliptic curve – the name of the curve refers to the order of the base field which is approximately $2^{254}$. The results were produced using a single thread and on a machine running 64-bit Ubuntu 19.04 with an Intel Core i5-8265U CPU with 8 cores and running at 1.60GHz, and 8GB of RAM. The duration shown is an average of running each operation 10 times. For further details and the tests’ code, visit report-tests/ in each library’s src/directory.

\(^2\)[https://www.chaijs.com/] – accessed 31-October-2019

### Table 7.2: Sizes in bytes of the elements from the mcl-wasm library.

<table>
<thead>
<tr>
<th>Element</th>
<th>Size (bytes)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Fr</td>
<td>32</td>
</tr>
<tr>
<td>$G_1$</td>
<td>96</td>
</tr>
<tr>
<td>$G_2$</td>
<td>192</td>
</tr>
<tr>
<td>$G_T$</td>
<td>384</td>
</tr>
</tbody>
</table>

### Table 7.3: Sizes in bytes of the elements stored in the user’s browser.

<table>
<thead>
<tr>
<th>Value</th>
<th>Size (bytes)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Credential</td>
<td>128</td>
</tr>
<tr>
<td>Issuer’s Public Key</td>
<td>512</td>
</tr>
</tbody>
</table>

# Table 7.2: Sizes in bytes of the elements from the mcl-wasm library.
<table>
<thead>
<tr>
<th>Test</th>
<th>Runtime in ms</th>
</tr>
</thead>
<tbody>
<tr>
<td>show signature</td>
<td>34</td>
</tr>
<tr>
<td>show</td>
<td>35</td>
</tr>
<tr>
<td>signature + pseudonym</td>
<td></td>
</tr>
<tr>
<td>check signature</td>
<td>14</td>
</tr>
<tr>
<td>check</td>
<td>13</td>
</tr>
<tr>
<td>signature + pseudonym</td>
<td></td>
</tr>
<tr>
<td>show + check signature</td>
<td>45</td>
</tr>
<tr>
<td>show + check signature + pseudonym</td>
<td>50</td>
</tr>
</tbody>
</table>

(a) Credential library

<table>
<thead>
<tr>
<th>Test</th>
<th>Runtime in ms</th>
</tr>
</thead>
<tbody>
<tr>
<td>check linking info</td>
<td>54</td>
</tr>
<tr>
<td>correctness</td>
<td></td>
</tr>
<tr>
<td>link 1 student</td>
<td>29</td>
</tr>
<tr>
<td>among 3 in</td>
<td></td>
</tr>
<tr>
<td>3 questionnaires</td>
<td></td>
</tr>
<tr>
<td>link 2 student</td>
<td>58</td>
</tr>
<tr>
<td>among 3 in</td>
<td></td>
</tr>
<tr>
<td>3 questionnaires</td>
<td></td>
</tr>
</tbody>
</table>

(b) Linking library

Table 7.4: Running times of the libraries’ most relevant operations.
Chapter 8

Conclusions

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, who can only learn information about the class in general and not about individual students. To provide for the ability to study and analyze student performance and evolution over time, we introduce the role of the education researcher. A researcher is able to ask 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 specific researcher that made the request. We stress however, that even though we allow this linkability, students remain completely anonymous and answers are only linkable if a student explicitly allows it.

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.
8.1 Contributions

The major contributions of this work are the following:

- Design of a browser-based anonymous and unlinkable questionnaire system with the possibility for user-controlled linkability of answers.
- A novel cryptographic scheme for the selective linkability of questionnaire answers, which is controlled by the user, and allowing a two-phase authorization of the linking.
- TypeScript implementation of an attribute-based credential system.
- TypeScript implementation of the designed linking scheme.
- Proof-of-concept demo of the designed system.

8.2 Future Work

One of the major ideas for future work is the elaboration of a formal security proof for the designed cryptographic linking scheme. Additionally, and drawing inspiration from Direct Anonymous Attestation, it could be interesting to explore the delegation of credential computations to an untrusted party, such as the questionnaire hosting server, to minimize the operations run on the user’s browser.

On the experimental side, one could run an experiment with a real-life class of students to assess the usability and communication speeds in such a context. It would also be of interest to open the discussion to the education research community, to understand what other features could be useful to have in a system like ours.
Bibliography


