# D3.2 Guidance on Selection and Complementarity of ABC Protocols

*Gert Læssøe Mikkelsen*
*Michael Østergaard Pedersen*
*Fatbardh Veseli*

| | |
|---|---|
| *Editors:* | *Gert Læssøe Mikkelsen (Alexandra Institute AS)* |
| | *Michael Østergaard Pedersen (Miracle A/S)* |
| *Reviewers:* | *Stamatiou Iwannis (*Computer Technology Institute & Press – "DIOPHANTUS"*)* |
| | *Gregory Neven (IBM Research – Zurich)* |
| *Identifier:* | *D3.2* |
| *Type:* | *Deliverable* |
| *Version:* | *1.0* |
| *Date:* | *12/08/2014* |
| *Status:* | *Final* |
| *Class:* | *Public* |

## Abstract

In this report we present comparison results on different Privacy-Enhancing Attribute Based Credential (Privacy-ABC) Schemes and give some guidance on choosing between different Privacy-ABC schemes. The target audience for this report is application developers and IT architects, that need easily accessible background information, comparison results, and guidance on choosing a Privacy-ABC scheme.

*The research leading to these results has received funding from the European Community's Seventh Framework Programme (FP7/2007-2013) under grant agreement n° 257782 for the project Attribute-based Credentials for Trust (ABC4Trust) as part of the "ICT Trust and Security Research" theme.*

# Members of the ABC4TRUST consortium

| | | | |
|---|---|---|---|
| 1. | Alexandra Institute AS | ALX | Denmark |
| 2. | CryptoExperts SAS | CRX | France |
| 3. | Eurodocs AB | EDOC | Sweden |
| 4. | IBM Research – Zurich | IBM | Switzerland |
| 5. | Johann Wolfgang Goethe – Universität Frankfurt | GUF | Germany |
| 6. | Microsoft Belgium NV | MS | Belgium |
| 7. | Miracle A/S | MCL | Denmark |
| 8. | Nokia Solutions and Networks Management International (NSN) | NSN | Germany |
| 9. | Computer Technology Institute & Press – "DIOPHANTUS" (CTI) | CTI | Greece |
| 10. | Söderhamn Kommun | SK | Sweden |
| 11. | Technische Universität Darmstadt | TUD | Germany |
| 12. | Unabhängiges Landeszentrum für Datenschutz | ULD | Germany |

# List of Contributors

| **Chapter** | **Author(s)** |
|---|---|
| Executive Summary | Gert Læssøe Mikkelsen (Alexandra Institute AS)<br>Michael Østergaard Pedersen (Miracle A/S) |
| Introduction | Gert Læssøe Mikkelsen (Alexandra Institute AS)<br>Michael Østergaard Pedersen (Miracle A/S) |
| Background | Gert Læssøe Mikkelsen (Alexandra Institute AS)<br>Michael Østergaard Pedersen (Miracle A/S)<br>Fatbardh Veseli (Johann Wolfgang Goethe – Universität Frankfurt) |
| Privacy-ABC Building Blocks | Gert Læssøe Mikkelsen (Alexandra Institute AS)<br>Michael Østergaard Pedersen (Miracle A/S) |
| Privacy-ABC Schemes | Gert Læssøe Mikkelsen (Alexandra Institute AS)<br>Michael Østergaard Pedersen (Miracle A/S)<br>Fatbardh Veseli (Johann Wolfgang Goethe – Universität Frankfurt) |
| Recommendations and Guidelines | Gert Læssøe Mikkelsen (Alexandra Institute AS)<br>Michael Østergaard Pedersen (Miracle A/S)<br>Fatbardh Veseli (Johann Wolfgang Goethe – Universität Frankfurt) |

# Executive Summary

The need for authentication on the Internet is growing. However, at the same time users' concern about data misuse is also increasing. Privacy-Enhancing Attribute Based Credentials offer a solution to this by providing, at the same time, support for strong authentication and strong privacy properties for the user, thereby stronger security for the user against data misuse.

In this report we present comparison results on different Privacy-ABC schemes, and give guidance on choosing between different Privacy-ABC schemes based on the functionality, security and performance they provide. Most results in this report are taken from earlier work in the ABC4Trust project, but new content has been added where previous results were focused on a different target audience. The results are presented in a compact and accessible way for application developers and IT architects that need easy accessible background information and performance results for choosing a Privacy-ABC scheme suitable for their application.

# Contents

# 1 Introduction

The need for authentication on the Internet is growing. However, at the same time the users' concern about data misuse is also increasing. Privacy-Enhancing Attribute Based Credentials (Privacy-ABCs) offer a solution to this by providing, at the same time, support for strong authentication and strong privacy properties for the user, thereby stronger security for the user against data misuse. Say that a user has to be over a certain age to access an online gambling website, then by using Privacy-ABCs the user will be able to prove that she is old enough without revealing *any* other information about herself such as her birthday, name or home address. Furthermore, other service providers are not able to tell that she visited the website, even if they cooperate with the owner of the website.

In this report we present comparison results on different Privacy-ABC schemes, and give some guidance on choosing between Privacy-ABC schemes based on the functionality, security and performance they provide. The target audience for this report is application developers and IT architects that need easy accessible background information and performance results for choosing a Privacy-ABC scheme suitable for their needs. For the comparison results to be useful, some background information is needed both to be able to understand the security properties, but also for the comparison of functionality and performance of different schemes.

This report mostly presents results from other reports of the ABC4Trust project, mainly *D3.1 Scientific Comparison of ABC Protocols* [Cam+14], however, where that deliverable is very formal in its definitions and results, the purpose of this report is to present these results in a shorter and more accessible way with more focus on practical use, especially for the security comparisons.

## 1.1 Privacy-Enhancing Attribute Based Credentials

The central entity in a Privacy-ABC system is the User, which wants to be able to access resources or services at one or more Service Providers. For gaining access to the service, the User has to prove that some attributes of her credentials fulfill the policy for access set forth by the Service Provider, which she can do by generating a *presentation token* based on her credentials. In a Privacy-ABC setup, the Service Provider is known as the Verifier, as this entity verifies the presentation token presented by the User. The Verifier trusts the proof based on the underlying cryptographic mechanisms and because the attributes have been certified by a trusted Issuer. Besides the User, the Verifier and the Issuer there can be two other optional entities: The Inspector and the Revocation Authority.

In some scenarios we want to offer anonymity for the honest users, but we still want to be able to somehow prevent or catch the dishonest users. This is where the Inspector comes into play. If the presentation token contains inspectable attributes, these attributes can be revealed by the Inspector, however, only if the inspection grounds are fulfilled. The inspection grounds are the description of which conditions have to be fulfilled before the Inspector will reveal the content of the inspectable attributes to the Verifier, and the User has to accept the inspection grounds before producing an inspectable presentation token. The inspection grounds are cryptographically linked to the inspectable attributes, such that the Inspector can be sure that if he chooses to inspect an attribute and send the content back to the Verifier, then this is done according to the inspection grounds accepted by the User.

The other optional entity is the Revocation Authority, which can revoke issued credentials if these credentials are marked as revocable.

One example of a use case for inspection and revocation is a hotel booking scenario. If the User has a credit card credential containing the credit card number as an attribute, then she can make a presentation at booking time proving that she has a valid non-revoked credit card, and reveal the credit card number as an inspectable attribute with a trusted third party as the Inspector. If she doesn't cancel the booking in time and does not show up, then the hotel can go to the Inspector, stating that the inspection grounds have been met (she did not show up and did not cancel her booking in time), get the credit card number, and charge her for the booking. In this scenario the honest users gets additional security, as they do not have to hand over their credit card number to the hotel at booking time. If the hotel offers reduced price or additional service to members of some club, students or other groups of people, the User can combine the credit card credential and the membership credential to get these additional services, and the hotel is ensured that the two credentials belongs to the same User.

Another feature of Privacy-ABC schemes are pseudonyms, which are linkable values the User can choose to add to the presentation of the credentials. If a User has a user account at a Service Provider, she can choose to create a pseudonym there, and then she will be able to access her user account by proving possession of her pseudonym. In this case Privacy-ABCs offer similar functionality to single sign-on services. Pseudonyms are often established during a presentation of one or more credentials. For example, when the User presents credentials showing that she is entitled to create a user account, she can at the same time create a pseudonym for the account to be used for future logins. Pseudonyms can also be *scope exclusive* meaning that one User can only create one pseudonym for a given service.

In the list below we summarize the different entities of a Privacy-ABC scheme:

- The User collects credentials from one or more Issuers, and uses these to create presentation tokens that reveal some information from one or more of her credentials. The tokens are sent to Verifiers that control access to services that the User wishes to access.
- An Issuer issues credentials to the User and vouches for the correctness of the attributes in the credential. The role of the Issuer is similar to a certificate authority in classical PKI solutions, and that of the Identity Provider in other IdM systems.
- A Verifier controls access to a service that the User wishes to access. Depending on the service requested, a Verifier imposes certain requirements on a User's credentials and what Issuer issued them, and checks if the presentation token sent by the User satisfies these requirements. If this is the case the User is allowed access.
- If a presentation contains one or more inspectable attributes, then these can be inspected by the Inspector if the certain conditions are fulfilled. Inspection means that the Inspector in this case learns the values of the inspectable attributes.
- A Revocation Authority can revoke issued credentials, if for any reason the credential should no longer be considered valid.

A Privacy-ABC scheme should support at least some of the following features:

- *Selective Disclosure*: A presentation token does not have to reveal all attributes of the used credentials.
- *Predicate Proofs*: In a presentation token, it should be possible to reveal only a predicate over some of the attributes in a credential.
- *Multi-Credential Proofs*: A presentation token can be based on attributes from more than one credential.

- *Key-Binding*: During issuance it is possible to bind a credential to a specific secret key held by the user, such that knowledge of that key is required during presentations of that credential. This can be used make sure that two credentials are issued to the same user i.e., they are bound to the same secret key; or if the key is stored on hardware device, it can prevent the use of a credential without using the hardware device at the same time.
- *Carry-Over Issuance*: It is possible to issue credentials containing attributes taken from another credential without the Issuer learning the value of the attributes.
- *Issuance on Hidden Attributes*: It is possible to issue credentials containing attributes chosen by the User without the Issuer learning the value of the attributes.
- *Pseudonyms*: A User can deliberately choose to break the *unlinkability* property by establishing a pseudonym with the Verifier, that she can prove possession of in later presentations.
- *Revocation*: If a credential is marked as *revocable* then a designated Revocation Authority can choose to mark the credential as invalid, and from that point on presentation tokens derived from it will no longer be accepted by a Verifier.
- *Inspection*: During presentation, a Verifier can request that certain attributes become *inspectable* meaning that a designated Inspector can later recover the value of those attributes if the conditions for inspection (the *inspection grounds* are fulfilled.

A Privacy-ABC scheme should fulfil the following security properties:

- *Unforgeability*: It should not be possible for a User to produce a presentation token based on attributes she has not received a certificate on. Also, it should not be possible for a group of colluding Users to combine their certified attributes.
- *Untraceability*: The Issuer cannot trace the use of a credential after issuance. Not even of the Verifier and the Issuer colludes and the Verifier supplies the Issuer with the presentation tokens derived from the credential.
- *Unlinkability*: A Verifier cannot link a presentation token to a previously seen presentation tokens unless the User deliberately makes this possible, e.g. though the use of pseudonyms.

For a more thorough, yet still easily accessible, description of these properties we refer the reader to Chapter 2 of D2.1 [Bic+14a]. A formal definition of these security properties can be found in D3.1 [Cam+14].

## 1.2 ABC4Trust Reference Implementation

In the ABC4Trust project a reference implementation of a Privacy-ABC scheme has been implemented which implements the functionality described in D2.1 and H2.2 [Bic+14a; Bic+13a]. The reference implementation consists of the Cryptographic Engine implementing the cryptographic functionality needed to support a Privacy-ABC scheme, and the ABC-Engine which parses policies, manages the flow of the protocols and handles credentials. The architecture of the Cryptographic Engine is described in H2.3 [Bic+13b] and the rest of the reference implementation is described in D4.2 [Bic+14b]. The main goal of the reference implementation has been to support the functionality and further performance enhancements might be possible. It is implemented as Java applications, but a feasibility study on the use of smart phones has been conducted in D4.4 [Jen14]. This study also has some performance tests which we do not include here, but instead we refer to D4.4 for these results.

## 1.3 Structure of this report

The structure of this report follows to some degree the structure of the ENISA report *Algorithms, Key Sizes and Parameters Report* [Eur13] as the purpose of the report at hand is similar to that of the ENISA report, although the focus of this report is strictly on Privacy-ABC schemes and the scope of the ENISA report is on more widely used cryptographic schemes.

In Section 2 we give some background information for the later comparisons. This background information is important for understanding the results of the comparisons, especially the security comparisons. Section 3 provides a theoretical security and functional comparison of various cryptographic building blocks used to build a Privacy-ABC scheme. The focus is on the security aspect, as most building blocks of the same type are rather similar when it comes to functionality. We do not compare the performance of the different building blocks, but only of the complete schemes since this is what the users will experience.

In section 4 we compare some examples of Privacy-ABC schemes based on a selection of building blocks. This is done both on functionality, security, and performance. We compare three different examples of Privacy-ABC schemes in the theoretical comparison, two based on the reference implementation and one outside the reference implementation. For the practical results we concentrate on two different instantiations of building blocks in the reference implementation.

Section 5 presents some recommendations and guidelines based on the comparisons of the previous sections. The purpose of this section is to guide developers, IT architects and to some degree decision makers into making the right choices when they have to choose a Privacy-ABC scheme for a given project.

# 2 Background

In this section we provide background information for the comparison of functionality, security and performance. Of these three, security is by far the most complex to compare and understand to a level which makes one capable of making a selection between schemes, which is why the security part is longer than functionality and performance.

## 2.1 Functional Comparison

The functionality provided by a Privacy-ABC scheme, depends a lot on the choice of underlying building blocks, but in this report we focus on the difference in functionality of Privacy-ABC schemes instead of the functionality of comparable building blocks, as this is the functionality that application developers have to work with.

Some features, can be treated both as functional and security features. One example is unlinkability, the property that is impossible to tell if two presentations are made using the same credential or two different credentials.

## 2.2 Security Comparison

In order to compare the security provided by different schemes we first need a definition of what security for a Privacy-ABC scheme means. A security definition will usually consist of a formal description of the assumed capabilities of the adversary combined with some properties of the scheme that should hold against such an adversary.

Different entities of a system might have different requirements on the security. Consider this simplified example in the Privacy-ABC setting: A scheme is considered secure for the Issuer and the Verifier if a dishonest User is unable to forge a credential, but the same scheme might leak information about a User's hidden attributes to the Verifier during presentation, and hence be considered insecure from the User's point of view. In D3.1 [Cam+14] we formally defined the security properties of a Privacy-ABC scheme to ensure that it provides security for all parties.

Breaking these security properties will have different implications and in some cases an adversary might have more time to break one security property over another. For example, forging a credential is only interesting for an adversary as long as the credential has not expired and the system being able to verify it is still running, whereas violating the User's privacy might have value to an attacker even years after the system has been shut down. Additionally it might be possible to upgrade a system to invalidate forgeries, but if private information has been leaked it cannot be unleaked.

We compare security from both a theoretical and a practical point of view. The theoretical comparison involves assumptions and security proofs, whereas the practical view is focused on key sizes needed to obtain a given level of security.

### 2.2.1 Security Properties and Assumptions

When it comes to the capabilities of the adversary, we consider both the computational resources available to the adversary, and other capabilities he might have, such as the ability to corrupt or colluding with a set of entities or Users. Some security properties are said to hold unconditionally which means that no adversary, no matter how many computational resources he has available, will ever be able to break that security property. Unfortunately only a few schemes are unconditionally secure for all their security properties, and this is especially true for the settings where we want to use Privacy-ABCs. In fact for many types of cryptographic primitives it can be proven that an unconditionally secure scheme cannot exist. We can, however, base the security on computational assumptions instead. Such assumptions are that some computational problems are infeasible to compute for sufficiently large numbers. The more well studied such an assumption is, the more we are inclined to believe that it is actually true.

To relate a security property of a scheme to a computationally assumption, we rely on a security proof, sometimes also called a security reduction. A security proof of a cryptographic scheme is a reduction showing that if there exists an adversary that can break the defined security property of the scheme, then there exists an algorithm turning this attack on the scheme into an efficient computable solution of the assumed infeasible computational problem. This means that if in fact the assumption that the underlying computational problem is hard holds, then the scheme is secure. One could argue that this does not actually prove the security of the scheme as it only moves the trust from the scheme to the assumption. This is of course true, however, many assumptions have been very well studied for many years, and are still believed to hold. Moreover, it is a lot easier to analyze one instance of a simple assumption than it is to analyse the security of a complex scheme without a proof. There are also schemes used in practice where the only assumption is basically the assumption that the scheme is secure. Despite this lack of a security proof, some of these schemes are used in practice either because they are more efficient or due to other desired properties they have.

These schemes that are not unconditionally secure, but are based on a computationally assumption are referred to as computationally secure as they are only secure against a computationally bounded adversary. By changing the key size, we can affect the security level, but also the performance of the scheme. The security level can vary from an adversary

Table 1: Security levels (symmetric equivalent) from ECRYPT II [Sma12]

| **Symmetric Security (bits)** | **Protection** | **Comment** |
|---|---|---|
| 32 | Attacks in real-time by individuals. | Only acceptable for auth. tag. |
| 64 | Very short-term protection against small organizations. | Should not be used for confidentiality in new systems. |
| 72 | Short-term protection against medium organizations, medium-term protection against small organizations. | |
| 80 | Very short-term protection against agencies, long term protection against small organizations. | Smallest general-purpose level, $\leq$ 4 years protection. |
| 96 | Legacy standard level. | Approx. 10 years protection. |
| 112 | Medium-term protection. | Approx. 20 years protection. |
| 128 | Long-term protection. | Good, generic application independent recommendation (approx. 30 years protection). |
| 256 | Foreseeable future | Good protection against quantum computers unless Shor's algorithm applies. |

being able to break the scheme in hours with a standard computer to taking millions of years with current technology. One additional note regarding key sizes is the tightness of a security proof. For some proofs we can only conclude that if an adversary can break the scheme then we can turn this into a solution of the underlying computational problem for much smaller numbers than the actual key size of the scheme. In that case one needs (in theory) to choose a larger key size so that the reduction results in a key size of the underlying computational problem that is still secure. When a reduction shows that the key size for a scheme and the size of an instance of the corresponding computational problem are of about the same size, we say that the reduction is tight. Reductions that are not tight are called loose. In practice, however, tightness of the reduction is often ignored in the analysis of key size which we also do in this report.

### 2.2.2 Key Sizes in Practice

Increasing the key size of a cryptographic scheme makes the scheme more secure, but this additional security comes at the cost of lower performance. Therefore when comparing cryptographic schemes it is important that they are compared at the same level of security. Since different cryptographic schemes may require different key sizes to provide the same level of security, one can use the concept of security levels corresponding to the security level against brute force attacks of an ideal symmetric cipher with the given key size.

Choosing the key size for a given security level for a concrete scheme requires taking many factors into account such as the underlying computational problem, the specific group, best known algorithms for solving the computational problem in that group and tightness of the security reduction. The ECRYPT II project provides a comprehensive report on which key sizes to use for cryptographic schemes in various groups [Sma12]. In Table 1 we briefly summarize the the various security levels they use.

Table 2 shows the relation between security level and actual key size for computational problems in various groups. The table uses groups instead of assumptions since for most well-known assumptions they are equally hard for a given key size in a given group. In the

Table 2: Key sizes for given security levels from ECRYPT II

| Symmetric | RSA Based | Subgroup | Logarithm Group | Elliptic Curve | Hash |
|---|---|---|---|---|---|
| 64 | 816 | 128 | 816 | 128 | 128 |
| 72 | 1008 | 144 | 1008 | 144 | 144 |
| 80 | 1248 | 160 | 1248 | 160 | 160 |
| 96 | 1776 | 192 | 1776 | 192 | 192 |
| 112 | 2432 | 224 | 2432 | 224 | 224 |
| 128 | 3248 | 256 | 3248 | 256 | 256 |
| 256 | 15424 | 512 | 15424 | 512 | 512 |

ECRYPT II report they provide more details about how the size of parameters in a given group can be affected by the computational assumption.

In Table 2 the columns refer to the following: *Symmetric* refers to the desired security level from Table 1; *RSA Based* is the size of the RSA modulus in schemes relying on RSA-like assumptions. *Subgroup* and *Logarithm Group* refer to the size of the subgroup and the size of the group for the schemes based on discrete logarithm based assumptions, when these are instantiated as a subgroup of a group defined by multiplication modulo a prime. *Elliptic Curve* refers to the size of the group for schemes based on discrete logarithm assumptions when these are instantiated as groups over elliptic curves. Note that this does not necessarily cover schemes relying on bilinear maps. *Hash* refers to the size of the output of hash algorithms.

### 2.3 Performance Comparison

Performance comparison is the comparison of various performance related properties of the selected Privacy-ABC schemes, and all results are based on concrete benchmarks from D3.1 [Cam+14]. Results in this section should only be taken as a rough estimate, as they only state what you would get on this very specific system, and furthermore the Reference Implementation has not been fully optimized for speed. It is also impossible to benchmark all possible use cases for Privacy-ABCs so the results might not cover the use case one has in mind. Therefore any User who wants to deploy Privacy-ABCs in a production environment should perform their own benchmarks on their exact scenarios before deciding on the final selections of building blocks for their system.

Nevertheless we believe that the results in this report can still provide some guidance and an understanding of what performance figures one can expect.

## 3 Privacy-ABC Building Blocks

The building blocks used in the reference implementation is compared in this section. As mentioned earlier the level of formalism in this report is much lower than that of D3.1 as here we concentrate on doing a presentation for a broader audience. Before we compare the building blocks, we first introduce the different computational assumptions the building blocks rely on, then we discuss the necessary security properties we need from the building blocks, and finally we present the building blocks, their properties and their security guarantee.

## 3.1 Computational Assumptions

There are many different assumptions in use in the cryptographic literature. Some of them are fairly new and only used to reason about security of a very small number of protocols, while others are widely used and have been so for many years. Clearly the latter have received much more scrutiny and are more widely believed to hold. All assumptions listed below belong to this group of well studied assumptions.

### 3.1.1 Unconditionally Security - no Computational Assumptions

A scheme where the security is not related to the computational resources of the adversary is called unconditionally secure. There exists a type of security between unconditional security and computational security, namely statistical security. However for simplicity we let unconditional security cover statistical security in this report.

### 3.1.2 Factoring Related Assumptions

The following assumptions all rely on factoring being hard, although it is unknown in general whether hardness of factoring is enough. These assumptions are all based on some properties when computing in a multiplicative group modulo a large composite number which is constructed as the product of two large prime numbers.

**Strong RSA (SRSA)** As with the RSA assumption the strong RSA (SRSA) assumption is related to the hardness of factoring. The SRSA assumption is a strengthening of the RSA assumption, where the adversary is given the ability to choose the public exponent of the instance he wants to attack. See D3.1 [Cam+14] for a formal description.

**Quadratic Residuosity (QR)** This is also an assumption related factoring, basically this assumption says that given an element $x$ in an RSA group, an adversary cannot tell whether or not there exists an element $y$ in the same group s.t. $x = y^2 \mod n$, where $n$ is the RSA modulus. See [HK09] for a more detailed description.

**Decisional Composite Residuosity (DCR)** This is also an assumption related factoring, basically this assumption says that given an element $x$ and RSA modulus $n$, as adversary cannot tell whether or not there exists an element $y$ s.t. $x = y^N \mod n^2$. See [Pai99] for a formal description.

### 3.1.3 Discrete Logarithm related Assuptions

**Discrete Logarithm (DL)** This assumption states that given an element $y$ and $x = y^a \mod p$, where $p$ is a large prime number, an adversary cannot compute $a$.

**Decisional Diffie-Hellman (DDH)** This assumption is related to the assumption above as solving DL implies the ability to solve this assumption, by not vice versa. This assumption states that it is hard, given three elements of a group, to distinguish whether the three elements are $(g^a, g^b, g^{ab})$ or $(g^a, g^b, g^c)$. See [Bon98] for a formal description.

When looking at computational assumptions one always has to keep in mind in which group the assumption is believed to hold. For example there are groups in which the DDH problem is easy to solve, yet computing discrete logarithms is still believed to be hard. The ability to compute discrete logarithms in a group would imply that it would be easy to solve the DDH problem in that group as well, but this is not the case in the opposite direction.

We therefore say that DDH is a stronger assumption than DL since requiring that the DDH assumption holds in a group is a more restrictive requirement. As we have just shown some assumptions are related, e.g. DL and DDH, while others are not.

### 3.1.4 Other assumptions

We introduce two other assumptions that are somewhat different, as they are not based on a simple computational problem. Nevertheless they are important as the Cryptographic Engine rely on these to hold for the security of the Privacy-ABC schemes.

**Collision Resistance** Collision resistance is the assumption that for a hash function $\mathsf{H}$, it is difficult to find two different inputs $x$ and $y$ such that $\mathsf{H}(x) = \mathsf{H}(y)$. Hash functions are defined in Section 3.2.1

**Fiat-Shamir Heuristic** The Fiat-Shamir Heuristic is the assumption that the Fiat-Shamir construction of non-interactive zero knowledge proofs is secure. Non-interactive Zero-Knowledge proofs are defined in Section 3.2.2.

The Fiat-Shamir Heuristic can be proven secure in the random oracle model, but unfortunately security in the random oracle model does not necessarily mean that the scheme is secure when used in practise. Still, practical uses of the Fiat-Shamir construction remain unbroken.

## 3.2 Building Blocks

We now describe some of the different building blocks that can be used to compose a Privacy-ABC scheme, with the the Privacy-Enhancing Attribute Based Signature (PABS) being the central building block. This list of building blocks is based on the building blocks used in the schemes described in Section 4, and most of the building blocks and the composition of these are formally described in D3.1. The security properties of the building blocks are:

**Security for Issuer/Verifier** Security for the Issuer and Verifier against cheating Users, trying to impersonate other Users, forge credentials or forge presentation proofs.

**User Privacy** Security for the User against any kind of tracking (untraceability and unlinkability).

These informal security properties refer to different formal properties for the different building blocks.

### 3.2.1 Cryptographic Hash Functions

Cryptographic Hash functions (simply called hash functions from now on) are a basic building block used as an internal part of some of the other building blocks. A hash function is a function that takes an input of any size and outputs a fixed size deterministic value depending on the input, but at the same time *collision resistant* which means that it is computationally infeasible to find two different inputs giving the same output. Further discussion on hash functions is out of scope of this work, but we refer to ENISA [Eur13] and ECRYPT II [Sma12] for more discussion and concrete instantiations.

As mentioned above, hash functions are a central component in the Cryptographic Engine as they are the basis for encoding of some types of attributes, as well as in Non-Interactive Zero Knowledge Proofs (see below). Since many security properties of a Privacy-ABC scheme rely on security of the hash function, we will not explicitly mention security of the hash function when we describe the underlying assumptions of the other building blocks.

### 3.2.2 Non-Interactive Zero Knowledge (NIZK) Proofs

Non-Interactive Zero Knowledge (NIZK) proofs are very basic building blocks in Privacy-ABC schemes, both on their own, but also as subprotocols of other building block. A NIZK proof is a two-party protocol, where the one party, the Prover, can prove properties about a secret message to the other party, the Verifier.

One of the simplest NIZK proofs is the proof of knowledge of a committed value. If one party have committed himself to a value using one of the commitment schemes mentioned above, he can send the commitment to another party and then prove that he indeed knows the committed value without revealing it. This might at first seem unimportant, but is actually very important in many cases. There exists NIZK proofs for proving a wide variety of properties about secret values contained in commitments, signed attributes, encryptions, etc.

NIZK proofs are one step protocols, where only one message is sent from the Prover to the Verifier, whereas in interactive proofs several messages are sent back and forth. Some interactive proofs, known as sigma protocols, can be converted into NIZK proofs by applying the Fiat-Shamir construction [FS87]. This construction introduces the Fiat Shamir heuristic 3.1.4 as an additional assumption.

NIZK Proofs must be *zero knowledge*, meaning that the Verifier learns nothing about the hidden value except it fulfils the proven property and *sound*, that if the proof verifies then indeed the proven property is fulfilled.

NIZK Proofs based on the Fiat-Shamir construction are a central component in the Cryptographic Engine as they are the basis for basically all functionality, and they are not easily replaced by another instantiation. Therefore, as with hash functions, we will not explicitly mention security of Fiat-Shamir construction when we describe the underlying assumptions of the other building blocks.

### 3.2.3 Commitment Schemes

A commitment scheme is a two phase protocol, where in the first stage one party (often the User in the context of Privacy-ABCs) can commit to a hidden value towards another party. This is done in such a way that the first party cannot change the hidden value, when it is to be revealed in the second phase. A commitment schemes has to be hiding, meaning the second party cannot extract any information about the hidden value from the commitment, and binding meaning the first party cannot reveal another value than the one committed to. Commitment schemes are often used to glue other protocols together, and is therefore a very basic building block. We show two commitment schemes and Table 3 gives an overview of the security properties of these two schemes.

**Pedersen Commitments** Was introduced by Pedersen in [Ped91]. The input that can be committed to is integers in a certain interval specified by the key size and other initialization parameters. The reader is referred to D3.1 [Cam+14] for a description of this scheme.

**Pedersen/Damgård-Fujisaki Commitments** The Pedersen/Damgård-Fujisaki Commitment scheme is an extension of the Pedersen commitments scheme making it possible to commit to arbitrarily big integers. The scheme was originally proposed by Fujisaki and Okamoto [FO97] and Damgård and Fujisaki [DF02]. The reader is again referred to D3.1 [Cam+14] for further description.

Table 3: Overview of commitment schemes and their corresponding assumptions

| Instantiation | Security Property | Formal Property | Assumption |
|---|---|---|---|
| Pedersen Commitments | Security for Issuer/Verifier | Binding | DDH |
| | User Privacy | Hiding | Unconditional |
| Pedersen/Damgård-Fujisaki | Security for Issuer/Verifier | Binding | Strong RSA assumption |
| | User Privacy | Hiding | Unconditional |

### 3.2.4 Privacy-Enhancing Attribute Based Signature (PABS) Schemes

The central building block of the Privacy-ABC schemes presented in this document is the Privacy-Enhancing Attribute Based Signature Scheme. During an interactive protocol the Issuer signs the attributes of the credential in a way that preserves the privacy of the User. The User can then use proofs over the signed attributes when doing a presentation. The security and functional properties of PABS schemes are shown below. For a full description, we refer to D3.1:

**Security for Issuer/Verifier** Security for the Issuer and Verifier against cheating Users, trying to impersonate other Users, forge credentials or forge presentation proofs.

**User Privacy** For Privacy-Enhanced Attribute Based Signature schemes this covers the following properties:

- *Untraceability* means that the Issuer is not capable of tracing the use of an issued credential, even when the Issuer and the Verifier colludes, except in trivial cases such as when a unique attribute known by the Verifier is revealed during presentation.
- *Attribute Hiding* means that the Verifier does not get any information about undisclosed attributes, even if the Issuer and the Verifier colludes.
- *Unlinkability* means that a Verifier cannot tell if two presentations are done using the same signature or two different signatures even from different Users. This holds even in the case where the Issuer and the Verifier colludes.

**Weak User Privacy** As the above except only *untraceability* and *attribute hiding* are provided.

**CL-Signatures** This scheme is due to Camenisch and Lysyanskaya [CL02a]. It is based on the SRSA assumption and functionality and security wise it fulfils all the security properties we would like a PABS scheme to have.

**Brands Signatures** This signature scheme is due to Brands [Bra93] and a variant of it is used in Microsoft U-Prove. This is also the variant we refer to when we talk about Brands signatures. The scheme is related to the DL assumption in the sense that if the DL assumption does not hold in the group that the scheme has been instantiated in then it is not secure. However, there does not exist a security proof stating that the scheme is secure if the DL assumption holds. In fact, According to Baldimtsi and Lysyanskaya [BL13] it is impossible using known random oracle rewinding techniques, the standard way to prove the security of these kinds of schemes, to prove that any of the assumptions listed in this section (as well as many other assumptions) are sufficient.

The scheme does not achieve unlinkability. This might, depending on the use case, not be relevant, and can in other cases be circumvented by making several signatures and then only use each of them once. On the positive side, Brands signature scheme is more flexible than CL when it comes to implementation, as it does not require computations modulo an RSA modulus, but can be instantiated over a subgroup of a group defined by multiplication modulo a prime or over elliptic curves. Therefore, in some use cases, Brands signatures might be more efficient than CL-signatures. U-Prove, as it is available directly from Microsoft [Paq13], can be implemented both over standard subgroups and elliptic curves whereas the Cryptographic Engine only implements the former.

Table 4: Overview of PABS schemes and their corresponding assumptions

| Instantiation | Security Property | Formal Property | Assumption |
|---|---|---|---|
| CL-Signatures | Security for Issuer/Verifier | Unforgeability | Strong RSA |
| | Privacy for the User (Untraceability, Attribute Hiding and Unlinkability) | User Private | Unconditional |
| | Weak privacy for the User (Untraceability and Attribute Hiding) | Weak User Private | Unconditional |
| Brands Signatures | Security for Issuer/Verifier | Unforgeability | No security proof exists |
| | Privacy for the User (Untraceability, Attribute Hiding and Unlinkability) | User Private | Does not fulfill this property |
| | Weak privacy for the User (Untraceability and Attribute Hiding) | Weak User Private | Unconditional |

### 3.2.5 Verifiable Encryption

Verifiable encryption is basically an encryption scheme with the special property that the party encrypting a value can prove properties of the encrypted value towards another party. The second party can then verify these claims without decrypting the ciphertext and without knowledge of the secret key. A verifiable encryption scheme should also have the additional property that besides the secret key, you need a label to decrypt, and this label has to be the exact same label that was used during encryption. Verifiable encryption is used to make inspection possible, such that the User can encrypt an attribute of her credential and make a proof that the encrypted value is actually the exact same value from the credential. In case of inspection the Verifier sends the ciphertext to the Inspector together with the inspection grounds. To ensure the correct inspection grounds are send to the Inspector they are used as the label.

**Camenisch-Shoup Verifiable Encryption Scheme** We only list one scheme here, namely the Camenisch-Shoup Verifiable Encryption Scheme [CS03] which fulfills the needed security properties, and even supports verifiable decryption. Verifiable decryption makes it possible for the inspector to prove that what he claims to have decrypted is in fact identical to what was in the ciphertext. This can be used to avoid a User to be framed by the Inspector.

Table 5: Overview of verifiable encryption schemes and their corresponding assumptions

| Instantiation | Security Property | Formal Property$^\alpha$ | Assumption |
|---|---|---|---|
| Camenisch-Shoup Verifiable Encryption | User Privacy | Encryption Security (IND-CCA2) | DCR |
| | Security for Verifier | Verifiability | Strong RSA |

$^\alpha$ As verifiable encryption is not treated in D3.1 [Cam+14] these formal properties cannot be found there. We refer to [CS03] for formal treatment of verifiable encryption.

### 3.2.6 Pseudonym Schemes

To implement the different kinds of pseudonyms used in a Privacy-ABC scheme, a pseudonym scheme is used.

The security properties of such a scheme are the following: The Issuer and Verifier are assured that the User actually knows the secret key used to make the pseudonym. This property is known as *key extractability.* It should be noted that despite the name this does not make it possible for neither the Issuer nor the Verifier to extract the User's private key. The scheme must also be collision resistant, meaning that an adversary cannot pretend to be a valid User by presenting the same pseudonym. For privacy the pseudonym scheme must be unlinkable, meaning that it is not possible to link two different pseudonyms even if they are generated from the same private key.

**Scope Exclusive Pseudonym Scheme** We only describe one pseudonym scheme here, for further description see D3.1 [Cam+14] and H2.3 [Bic+13b].

Table 6: Overview of pseudonym schemes and their corresponding assumptions

| Instantiation | Security Property | Formal Property | Assumption |
|---|---|---|---|
| Scope Exclusive Pseudonym Scheme | Security for Issuer/Verifier | Key Extractability | Unconditional |
| | | Collision Resistant | Security of hash function |
| | User Privacy | Pseudonym Unlinkability | DDH |

### 3.2.7 Revocation Schemes

To allow issued credentials to be revoked at a later stage, revocation schemes are used. These schemes work by embedding a unique revocation handle as a never disclosed attribute in the credential during issuance. When doing a presentation proof the User does a NIZK proof that the revocation handle is not on the list of revoked revocation handles (black-list revocation) or that it is on the list of valid revocation handles (white-list revocation).

One of the required security properties is *revocation soundness* meaning that the User must know the revocation handle when doing the proof, that only the Revocation Authority can come up with a new valid revocation information and that if the proof is accepted, then the revocation handle has not been revoked. The other security property is *Revocation privacy* which requires that no adversary can tell which of two unrevoked revocation handles are used in a revocation token, ensuring the User that she does not reveal anything besides the fact that her credential has not been revoked.

**Nakanishi et al. revocation scheme** This is a black-list revocation scheme proposed by Nakanishi et al. [Nak+09]. It is based on a signature scheme, and hence some security properties rely on the security of this signature scheme.

**CL revocation scheme** This is a white-list revocation scheme proposed by Camenisch and Lysyanskaya [CL02b]. It works by accumulating valid revocation handles in a single value (the accumulator) and giving the User a witness she can use to do a NIZK proof that her revocation handle is indeed stored inside the accumulator.

Table 7: Overview of revocation schemes and their corresponding assumption

| Instantiation | Security Property | Formal Property | Assumption |
|---|---|---|---|
| Nakanishi et al. revocation | Security for Issuer/Verifier | Revocation Soundness | Unforgeability of underlying signature scheme |
| | User Privacy | Privacy | Unconditional |
| CL revocation | Security for Issuer/Verifier | Revocation Soundness | SRSA |
| | User Privacy | Privacy | Unconditional |

## 4 Privacy-ABC Schemes

The Cryptographic Architecture is very flexible in the sense that it allows different instantiations of various cryptograhic building blocks, as long as these instantiations implement a specific interface. While this provides great flexibility, it also makes comparisons somewhat more complicated as there are many different Privacy-ABC schemes available, depending on the specific instantiations of the cryptographic building blocks.

However, in the reference implementation of the Cryptographic Engine most of these building blocks have only one implementation. In fact the only exception is the signature building block that have two different implementations (either CL signatures, or Brands signatures). We therefore define two Privacy-ABC schemes as the basis for the comparison depending on the implementation of the signature building block. We also include a third scheme, namely MS U-Prove which is the version of U-Prove that is available directly from Microsoft [Paq13]. While MS U-Prove is more limited in supported features, we do provide a converter that convert MS U-Prove presentation tokens so they can be used with the Reference Implementation. Table 8 lists the implementations of the building blocks for the Privacy-ABC schemes we compare in this section.

We do want to point out that there do exist building blocks with different properties than the ones listed here. These building blocks could be implemented in the Cryptographic Engine if they have some desired properties not found in the currently implemented building blocks.

Table 8: The three Privacy-ABC schemes compared

| Scheme | Building Blocks |
|---|---|
| PABC-CL | Pedersen/Damgård-Fujisaki Commitments |
| | Camenisch-Lysyanskaya Signatures |
| | Camenisch-Shoup Encryption Scheme |
| | Scope-Exclusive Pseudonym Scheme |
| | Camenisch-Lysyanskaya Accumulators |
| PABC-Brands | Pedersen/Damgård-Fujisaki Commitments |
| | Brands Signatures |
| | Camenisch-Shoup Encryption Scheme |
| | Scope-Exclusive Pseudonym Scheme |
| | Camenisch-Lysyanskaya Accumulators |
| MS U-Prove [α] | Pedersen Commitments |
| | Brands Signatures |

[α] MS U-Prove can be implemented over a subgroup of a group defined by multiplication modulo a prime or over elliptic curves. The supported features are identical for both, but performance and key sizes differ between them.

## 4.1 Functional Comparison

In this section we summarize the results of the functional comparison in D3.1 [Cam+14]. The features used in the comparison are not every feature that is possible to achieve with Privacy-ABCs, but rather a list of the features currently supported by the ABC4Trust Policy Language [Cam+13]. For a detailed description of each feature we refer to D2.1 [Bic+14a] and H2.2 [Bic+13a].

In the tables below we denote each feature with one of the values from Table 9.

Table 9: Possible comparison values

| Value | Meaning |
|---|---|
| Yes | The feature is supported |
| No | The feature is not supported |
| Not Applicable | The feature in question is irrelevant, e.g. it depends on an unsupported feature |

In Table 10 we show the supported functionality for each Privacy-ABC scheme related to issuance. It comes as no surprise that the two schemes based on the ABC4Trust Cryptographic Engine support the same features, whereas MS U-Prove supports a more basic set of features.

Table 11 shows the supported functionality related to presentation. Again, MS U-Prove lacks the more advanced features that were implemented in ABC4Trust. However, we do note that it is possible to convert MS U-Prove presentation tokens into PABC-Brands presentation tokens if those features are needed.

For the Privacy-ABC schemes that do support predicate functions over attributes, we show which predicates are supported in Table 12.

While we compare functionality and not security in this section, some security properties might not be needed in all use cases and hence they can be viewed as functional properties. In Table 13 we list those properties.

If Privacy-ABCs are unlinkable there is no way to control how many times a User can use a credential for a presentation unless the scheme supports some form of limited spending functionality. In Table 14 we show the level of support for limited spending.

Table 10: Issuance features of Privacy-ABC schemes

| Feature | PABC-CL | PABC-Brands | MS U-Prove |
|---|---|---|---|
| Issuance from scratch | Yes | Yes | Yes |
| Issuance of key-bound credentials | Yes | Yes | Yes$^{\alpha}$ |
| Issuance with carry-over attributes | Yes | Yes | No |
| Issuance on hidden attributes | Yes | Yes | No |
| Issuance on jointly random attributes | No$^{\beta}$ | No$^{\beta}$ | No$^{\gamma}$ |
| Credential update | No | No | No |

$^{\alpha}$ All U-Prove tokens are bound to a token key.
$^{\beta}$ Jointly random attributes are supported by the policy language, but not by the current implementation.
$^{\gamma}$ U-Prove specifies that the User and Issuer starts with a shared list of all attribute values. Some of those could possibly be generated jointly random, but this would happen outside the scope of the U-Prove protocols.

Table 11: Presentation features of Privacy-ABC schemes

| Feature | PABC-CL | PABC-Brands | MS U-Prove |
|---|---|---|---|
| Selective disclosure | Yes | Yes | Yes |
| Predicate functions over the attributes | Yes | Yes | No |
| Multi-credential presentations | Yes | Yes | No |

Table 12: Predicate functions over attributes of Privacy-ABC schemes

| Feature | PABC-CL | PABC-Brands | MS U-Prove |
|---|---|---|---|
| Equality of strings | Yes | Yes | Not Applicable |
| Equality of integers | Yes | Yes | Not Applicable |
| Equality of booleans | Yes | Yes | Not Applicable |
| Equality of times | Yes | Yes | Not Applicable |
| Equality of dates | Yes | Yes | Not Applicable |
| Inequality of strings | Yes | Yes | Not Applicable |
| Inequality of integers | Yes | Yes | Not Applicable |
| Inequality of booleans | Yes | Yes | Not Applicable |
| Inequality of times | Yes | Yes | Not Applicable |
| Inequality of dates | Yes | Yes | Not Applicable |

Table 13: Security features of Privacy-ABC schemes

| Feature | PABC-CL | PABC-Brands | MS U-Prove |
|---|---|---|---|
| Untraceability | Yes | Yes | Yes |
| Unlinkability | Yes | No | No |

Table 14: Limited spending of Privacy-ABC schemes

| Feature | PABC-CL | PABC-Brands | MS U-Prove |
|---|---|---|---|
| Limited Spending | Yes$^{\alpha}$ | Yes$^{\alpha\beta}$ | Yes$^{\beta}$ |

$^{\alpha}$ Some form of limited spending can be implemented using for example scope-exclusive pseudonyms.
$^{\beta}$ Since Brands signatures are linkable, they provide a form of limited spending towards the same Verifier.

Pseudonyms allow a User to establish a linkable identity with a Verifier. In Table 15 we show the level of support for different types of pseudonyms in Privacy-ABC schemes.

Table 15: Types of pseudonyms of Privacy-ABC schemes

| **Feature** | **PABC-CL** | **PABC-Brands** | **MS U-Prove** |
|---|---|---|---|
| Verifiable pseudonyms | Yes | Yes | Yes |
| Certified pseudonyms | Yes | Yes | Yes |
| Scope-exclusive pseudonyms | Yes | Yes | No |

Inspection allow a Verifier to request, during presentation, that some attributes can later be revealed by a designated Inspector. Table 16 lists which Privacy-ABC schemes that support inspection.

Table 16: Inspection support of Privacy-ABC schemes

| **Feature** | **PABC-CL** | **PABC-Brands** | **MS U-Prove** |
|---|---|---|---|
| Support for inspection | Yes | Yes | No |

Revocation in Privacy-ABC schemes work by having an entity called the Revocation Authority being responsible for maintaining some public revocation information, which all Users and Verifiers need when proving and verifying that a credential is not revoked. In some cases Users also need some User specific non-revocation information to prove that their credential is not revoked, and Issuers might need some information from the Revocation Authority when issuing credentials. This creates a somewhat complicated revocation picture where many parameters comes into play. In Table 17 we summarize the features of the currently implemented revocation scheme. We note that the revocation scheme described in D3.1 is different from the CL revocation implemented in PABC-CL and PABC-Brands and actually does not require the User to keep user specific non-revocation information.

Table 17: Revocation features of Privacy-ABC schemes

| Feature | PABC-CL | PABC-Brands | MS U-Prove |
|---|---|---|---|
| Issuer driven revocation | Yes | Yes | No |
| Verifier driven revocation | No$^{\alpha}$ | No$^{\alpha}$ | No |
| User connectivity with the Revocation Authority during presentation | Application specific | Application specific | Not Applicable |
| Verifier connectivity with the Revocation Authority | Application specific | Application specific | Not Applicable |
| Issuer connectivity with the Revocation Authority | Yes | Yes | Not Applicable |
| Offline usage | No | No | Not Applicable |
| Support for immediate revocation | Yes$^{\beta}$ | Yes$^{\beta}$ | Not Applicable |
| Scheme distribution | No | No | Not Applicable |
| Backward-unlinkability after revocation | Yes | Yes | Not Applicable |
| Anonymous update of non-revocation evidence | Yes | Yes | Not Applicable |
| User unlinkability during update of non-revocation evidence | Application specific$^{\gamma}$ | Application specific$^{\gamma}$ | Not Applicable |
| Frequency of User contact with Revocation Authority | After revocation$^{\delta}$ | After revocation$^{\delta}$ | Not Applicable |
| Frequency of Verifier contact with Revocation Authority | Application specific | Application specific | Not Applicable |

$^{\alpha}$ This feature is partly implemented in the Reference Implementation, however it is not yet fully functional.
$^{\beta}$ Only if the Verifier always keeps a up to date copy of the latest revocation information.
$^{\gamma}$ The non-revocation information is public and can be downloaded anonymously. However, some attacks might still leak information about the User (timing attacks, IP address, etc.).
$^{\delta}$ The user needs some way to learn that a revocation has taken place, either by asking the Revocation Authority or by having revocations take place at fixed intervals.

## 4.2 Security Comparison

Security comparison of Privacy-ABCs was done in D3.1. In Part 1 we formally defined security properties of Privacy-ABCs, formally analysed the building block based construction which is the basis for the design of the Cryptographic Architecture and provided instantiations of the building blocks to demonstrate feasibility of this approach. Part 2 used key sizes of the underlying building blocks to base the practical comparisons on instantiations with comparable security levels.

In this section we summarize these results. We compare the security properties of the implemented Privacy-ABCs from both a theoretical and a practical perspective. The theoretical security comparison looks at security proofs and assumptions while the practical security comparison takes into account which key sizes are needed for a given level of security.

### 4.2.1 Theoretical Security Comparison

In Section 3 each building block is described together with the assumptions upon which they are based. It is obvious that a scheme relying on a combination of e.g. two building blocks require at least the assumptions of both building blocks to hold. However, the other way, that the combination is secure given that the assumptions of both building blocks hold is not necessarily true. The building blocks are only proven secure as standalone schemes, and combining them might not preserve their security properties. Nevertheless, using building blocks with security proofs to build Privacy-ABC schemes is still a big step towards a secure scheme.

In D3.1 we formally defined security properties of a Privacy-ABC scheme, namely *correctness*, *pseudonym collision-resistance*, *unforgeability* and *privacy* (or *weak privacy*). However, we ignore the completeness property in this comparison, as it is not strictly a security requirement and doesn't depend on any assumptions. In Table 18 we show which entities are affected by the different security properties of Privacy-ABC schemes.

Table 18: Security properties and affected entities for Privacy-ABC schemes

| **Pseudonym Collision-Resistance** | **Unforgeability** | **Privacy** | **Weak-Privacy** |
|---|---|---|---|
| User, Issuer, Verifier | User[α], Issuer, Verifier | User | User |

[α] While unforgeability is primarily security for the Issuer/Verifier it also guarantees the User that an adversary cannot prove possession of an attribute that was intended to be unique for the User.

In Table 19 we summarize which assumptions must necessarily hold for the given security properties of our three candidate Privacy-ABC schemes.

Table 19: Underlying assumptions for properties of Privacy-ABC schemes

| **Scheme** | **Pseudonym Collision-Resistance** | **Unforgeability** | **Privacy** | **Weak-Privacy** |
|---|---|---|---|---|
| PABC-CL | Security of hash function[α] | SRSA | Unconditional, DCR[δ] | Unconditional, DCR[δ] |
| PABC-Brands | Security of hash function[α] | DL, Unknown[β], SRSA[γ] | No | Unconditional, DCR[δ] |
| MS U-Prove | Not Applicable[ϵ] | DL, Unknown[β] | No | Unconditional |

[α] Only applicable for scope exclusive pseudonyms. Non-scope exclusive pseudonyms are unconditionally secure.
[β] There does not exist a security proof for Brands signatures, but the best currently known attack is to solve the DL problem.
[γ] This assumption is only relevant if the revocation feature is used.
[δ] This assumption is only relevant if the inspection feature is used.
[ϵ] Pseudonyms in MS U-Prove rely on the unforgeability property for security.

The different computational assumptions are widely believed to be of the same difficulty, and hence all security properties that depends on a computational assumptions should provide the same level of security. However, only the PABC-CL Privacy-ABC scheme is based solely on provably secure building blocks. Both PABC-Brands and MS U-Prove relies on the security of the Brands signature scheme for which no security reduction is currently known. Still there are no known attacks against the scheme either.

### 4.2.2 Practical Security Comparison

Assuming that the underlying assumptions in the previous section all hold, does it mean that we can be sure that there is no practical way to break the schemes in question? The answer to that question is unfortunately no. First of all, as we have already mentioned, there is no proof that composition of the building blocks as done in the Cryptographic Architecture results in a secure Privacy-ABC scheme. Second, in practice we need appropriate key sizes for the building blocks to provide an adequate level of security. Finally there could be implementation errors that could compromise security in some way such as poor random number generation, buffer overflows, side-channel leakage of secrets, etc.

In D3.1 we have a description of a Privacy-ABC scheme that is provably secure if the building blocks are secure. However, this scheme is a proof of concept, and for efficiency reasons the Cryptographic Architecture is designed in another way, so in this security comparison we have to assume that the composition of the building blocks as done in the Cryptographic Architecture is secure. We also assume that there are no implementation errors that affect security of the implementation.

What is now left is to look at the key sizes for the building blocks. The design of the Cryptographic Architecture allows for adding different implementations of the building blocks, so in general it is not possible to talk about a single key size for the entire Privacy-ABC scheme. However, the building blocks that are currently implemented in the Cryptographic Engine all use key sizes from the *RSA Based* and *Logarithm Group* columns of Table 2. Since they are identical, the current version of the Cryptographic Engine only takes a single key size as input, which is a key size from the *RSA Based* column. If one were to implement building blocks with key sizes from e.g. the *Elliptic Curve* column, one would need to do this differently, e.g. by supplying a key size for each building block, or by specifying a security level as input and letting each building block generate keys of the correct length for that level of security.

By taking the underlying assumptions for the building blocks in Section 3 into account and combining them with the ECRYPT II recommended key sizes in Table 2, we show the actual key sizes for the different building blocks that provide a comparable level of security. These key sizes are show in Table 20.

Table 20: Actual key sizes for given security levels

| **Sec. Level** | **Pedersen/ Damgård-Fujisaki Commitments** | **CL Signatures** | **Brands Signatures (Subgroup)** | **Brands Signatures (Elliptic Curve)** | **Camenisch-Shoup Encryption Scheme** | **Scope-Exclusive Pseudonym** | **CL Accumulators** |
|---|---|---|---|---|---|---|---|
| 64 | 816 | 816 | 816 | 128 | 816 | 816 | 816 |
| 72 | 1008 | 1008 | 1008 | 144 | 1008 | 1008 | 1008 |
| 80 | 1248 | 1248 | 1248 | 160 | 1248 | 1248 | 1248 |
| 96 | 1776 | 1776 | 1776 | 192 | 1776 | 1776 | 1776 |
| 112 | 2432 | 2432 | 2432 | 224 | 2432 | 2432 | 2432 |
| 128 | 3248 | 3248 | 3248 | 256 | 3248 | 3248 | 3248 |
| 256 | 15424 | 15424 | 15424 | 512 | 15424 | 15424 | 15424 |

For simplicity the output size of the hash function does not depend on the chosen key size, but is fixed at 256 bits using SHA256, except in some cases where SHA1 is used for compatiblity with MS U-Prove. As hashing is very efficient compared to the operations of the other building blocks, there is no reason to use a lower security level for the hash function.

The key size for Brands signatures is based on the assumption that solving the discrete logarithm problem is the most efficient attack, but there is no proof that this is the case. If other attacks exist this might influence the actual key size, or render the scheme insecure for any key size.

## 4.3 Performance Comparison

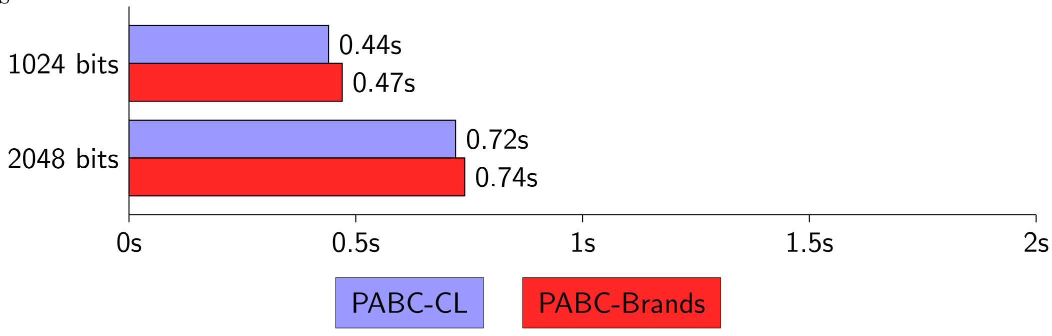
In this section we summarise the practical benchmarks from D3.1 and the conclusion drawn from them. There are no comparable practical benchmarks for MS U-Prove since it does not fit directly with the policies used in ABC4Trust and hence we couldn't run the same benchmarks we did for PABC-CL and PABC-Brands. However we do in some cases reason about how we would expect MS U-Prove to perform in comparison to the other two schemes.

All benchmarks were performed on a computer with dual core 1.8 GHz Intel Core i7 processor with 4 GB RAM running OS X 10.8.5, except for the hardware smart card benchmarks, which were performed on a MultOS ML3 smart card connected with a card reader to a Windows 7 32-bit PC with an Intel Core Duo 2.2 GHz CPU and 2 GB RAM.

### 4.3.1 Computational Efficiency

Computational efficiency measures the time to perform different types of operations of issuance and presentation. In Figure 1 we look at how different types of issuance of credentials compare in terms of computational efficiency for two different key sizes, namely 1024 and 2048 bits. In Figure 2 we compare simple issuance, issuance when showing a pseudonym, issuance with same-key binding as the pseudonym, and issuance with carry-over of attributes from another credential.

Figure 1: Performance for simple issuance of credentials with six attributes and various key sizes



For simple issuance, we see that PABC-CL and PABC-Brands are close with a negligible advantage to PABC-CL. As mentioned earlier, PABC-Brands does not provide unlinkability, but one way to achieve it anyway is to issue multiple credentials and then only use each of them once. The Cryptographic Engine supports this, and our experiments have shown that issuing 20 PABC-Brands credentials at a time only takes about twice as much time as issuing a single credential. This is due to optimizations in the issuance process where some values only need to be computed once.

Using some of the advanced issuance features, such as key-binding and carry-over of attributes, results in lower performance as they typically involve an additional NIZK proof compared to simple issuance. For the key size of 2048 bits that was used in these advanced issuance benchmarks, PABC-CL is slightly more efficient than PABC-Brands. In general, there is an overhead for both schemes when using advanced issuance.

Presentation of credentials with various features are shown in Figure 3. First we look at three benchmarks involving only one credential: Just proving possession of a credential (Credential Only), proving possession of a credential and a pseudonym (Cred + Nym) and proving possession of a key-bound credential and a pseudonym (Cred + Nym + Key). Then we have three benchmarks involving more than one credential: Proving possession of two credentials where one attribute in one of them is equal to one attribute in the other (Equality Attribute), and proving possession of 2 and 3 credentials respectively. Finally we show benchmarks for a proof of possession of one credential with one inspectable attribute and possession of one revokable credential.

In most scenarios, presentation efficiency is the most important performance factor, as users will normally use it much more often than issuance. A simple presentation proving

Figure 2: Performance for advanced issuance of credentials with six attributes and a key size of 2048 bits

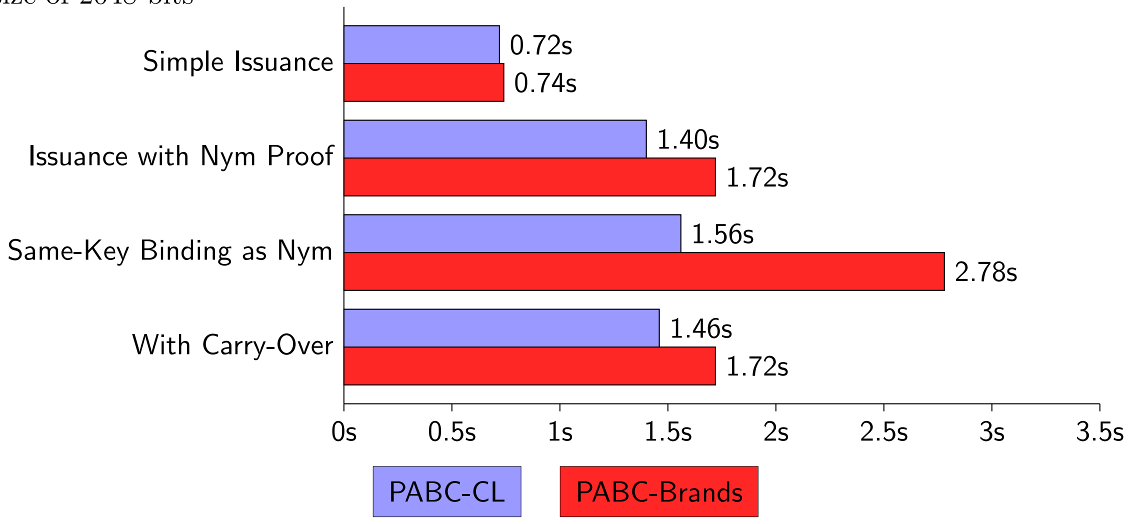
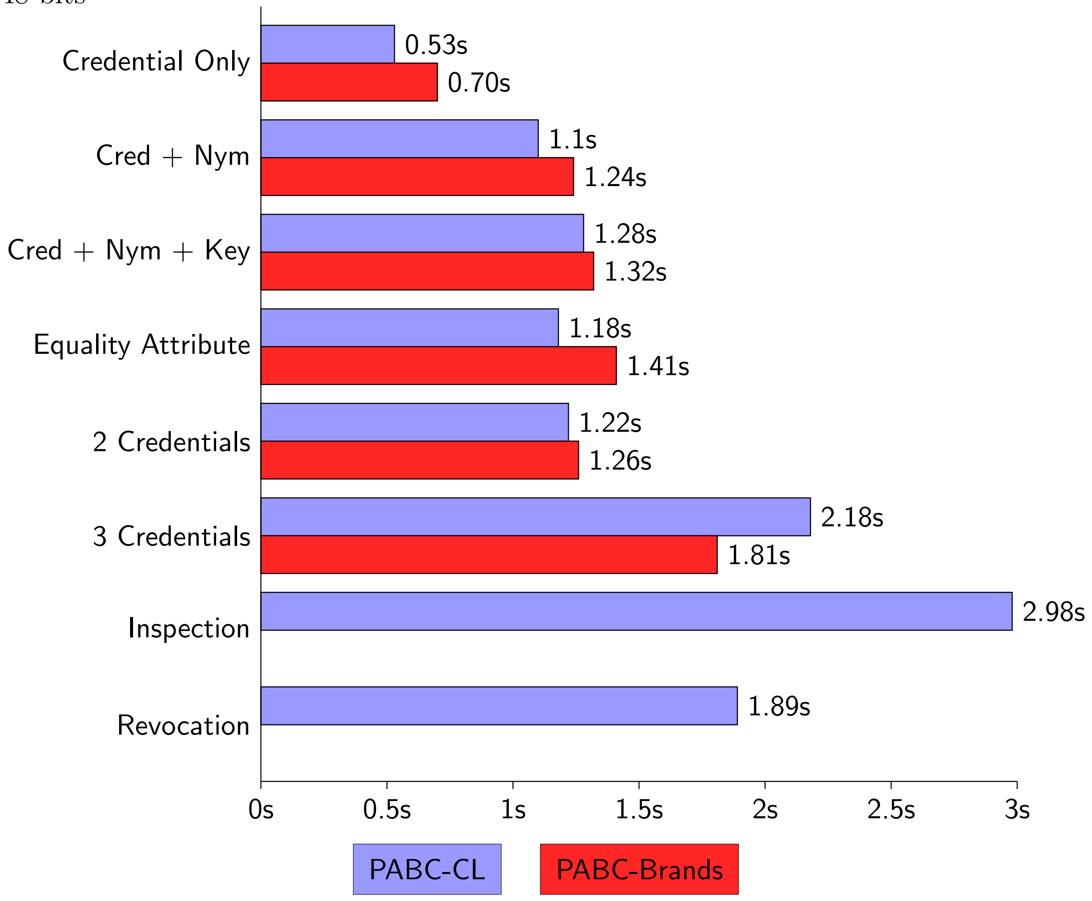


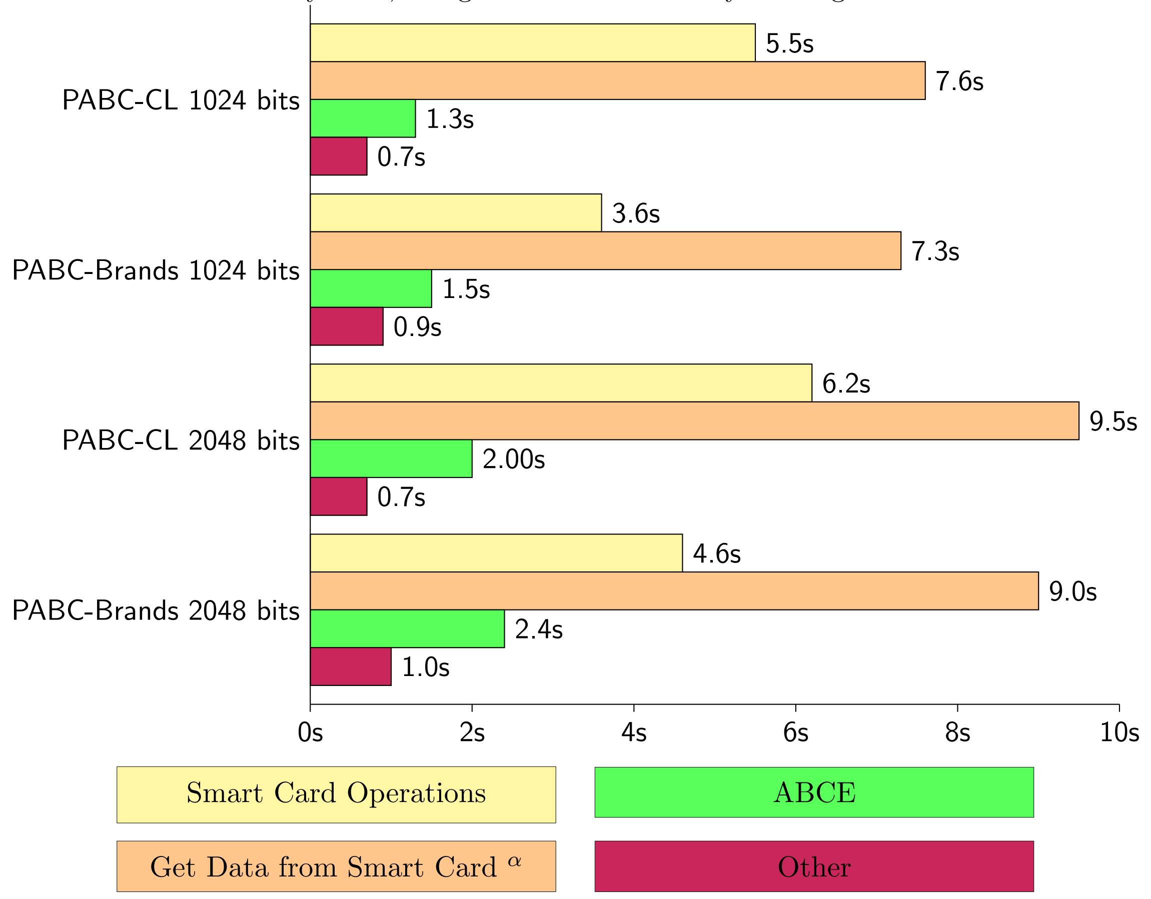
Figure 3: Performance for presentation of credentials with six attributes and a key size of 2048 bits



possession of a credential (without revealing anything else) is fastest as one would probably expect. Use of additional presentation features, such as key-binding, pseudonyms, equality proofs, and even presenting two credentials at the same time increases the presentation time, but have about the same performance. This is not surprising as each of these features introduces one additional NIZK proof. We also see that presenting three credentials instead of two adds about 33% to the running time of the presentation. The performance between PABC-CL and PABC-Brands is not significantly different for any of the tests.

All benchmarks so far are performed in software only, but in Figure 4 we take a look at the computational efficiency of presentation of a single PABC-CL or PABC-Brands key-bound credential stored on a smart card at various security levels. The total time used during presentation is the sum of the four parts.

Figure 4: Time spent in different phases during presentation of a single credential with one attribute and various key sizes, using a smart card for key-binding



$\alpha$ This data can be cached after the first use, so this is only needed the first time the data is needed.

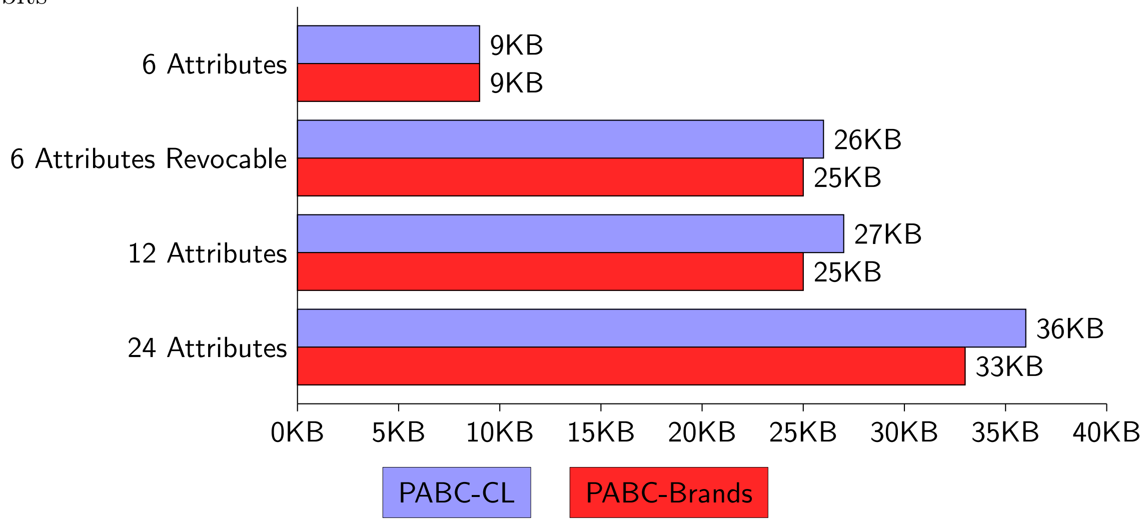
From these results we can see that most of the time is spent on getting data from, and doing computations on, the smart card. Since the ABCE actually performs more cryptographic operations than the smart card does, we can conclude that doing the cryptographic operations on the smart card is significantly slower than on the PC. It is worth noting that some low level benchmarks of the smart cards [Pai14] have shown better performance than in Figure 4 so it is possible that there are some performance issues in the layer between the Java platform and the smart card.

### 4.3.2 Communication Efficiency

In this section we summarize the communication efficiency of our two schemes, meaning that we measure the amount of data sent between the User and the Issuer/Verifier. The results are show in Figure 5 for various credentials.

We see that the size of the data is roughly the same for PABC-CL and PABC-Brands, with a small advantage to PABC-Brands.

Figure 5: Communication size for issuance of various credentials with a key size of 2048 bits



In Figure 6 we show the size of the presentation token during presentation of various credentials. Some of the tests are for the same credentials as in Figure 3, but we have added tests for proving possession of credentials with 12 and 24 attributes respectively. In all tests PABC-CL outperforms PABC-Brands with up to a factor of 2.

#### 4.3.3 Storage Efficiency

This benchmark is concerned with the size of the credential which the User has to store. In some cases this is not an important measurement, whereas in other cases it is critical e.g. when the credential has to be stored on devices with very little available storage space such as a smart card.

Figure 7 shows the credential size for credentials with 6, 12 and 24 attributes and also for a revocable credential with 6 attributes.

There is no noticeable difference in size for PABC-CL and PABC-Brands, however one has to keep in mind that PABC-Brands does not provide unlinkability. This can in some cases be mitigated by issuing multiple credentials, but that would increase the storage requirements for PABC-Brands linearly in the number of issued credentials.

Figure 6: Presentation token size for various credentials with six attributes (unless otherwise stated) and a key size of 2048 bits

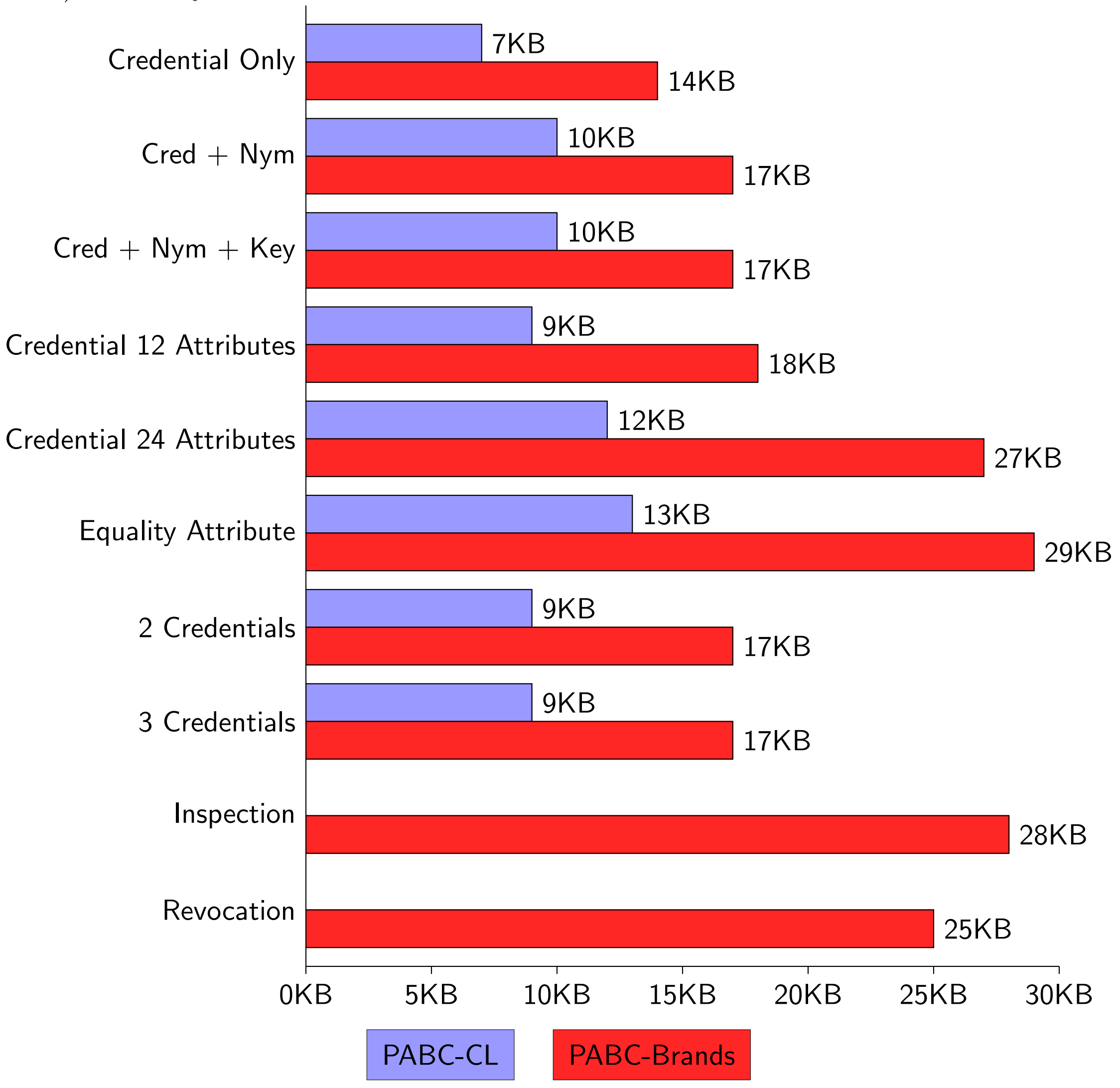
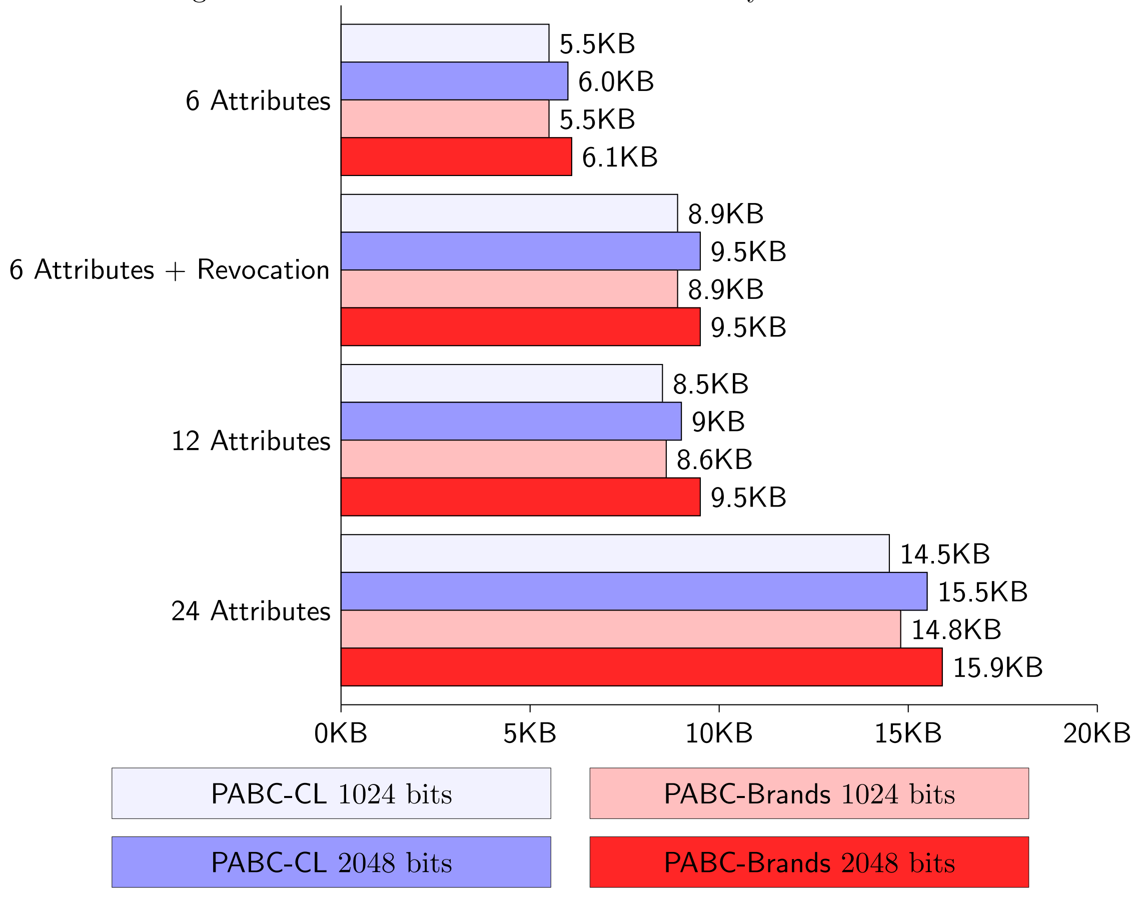


Figure 7: Size of various credentials with a key size of 2048 bits



# 5 Recommendations and Guidelines

When it comes to choosing a Privacy-ABC scheme for a given scenario there are many factors to consider as each scheme has its strengths and weaknesses. In the previous chapters we have summarized the differences between them, and in this chapter we give some recommendations on which one to choose for a given scenario. First we give recommendations based on specific features, and then we give more general recommendations and discuss a number of factors that influence all Privacy-ABC schemes.

The conclusions in this section do not tell the whole story, and should only serve as a quick overview. For more detailed analysis, we refer to the rest of this document as well as the previous documents, especially D2.1 [Bic+14a], H2.2 [Bic+13a] and D3.1 [Cam+14]. Also note that performance figures and supported features are based on the current version of the ABC4Trust Reference Implementation as well as the current version of MS U-Prove and could easily change in the future as more features are added or the implementations are optimized.

## 5.1 Recommendations Based on Specific Features

Below we describe some features, and which schemes are recommended if those features are needed. Clearly one can have scenarios where multiple features are needed, but where the recommendations are not consistent. In such case one has to make a trade-off.

**Simple Setup** Scenarios where none of the advanced features of Privacy-ABCs are needed. Users can get credentials and do selective disclosure towards relying parties. Security guarantees to the User is that issuance and presentation cannot be linked. There is no support for presentation policies.

*Recommendation:* MS U-Prove unless unlinkability is needed. While all Privacy-ABC schemes support this scenario, MS U-Prove is simpler and is expected to perform better, partially due to being simpler.

**Policy Language** Scenarios where more advanced features or where a well-defined policy language for specifying presentation policies are needed.

*Recommendation:* PABC-CL or PABC-Brands. Since they are based on the ABC4Trust Architecture, both of these schemes support this scenario equally well.

**Unlinkability** Scenarios where presentation tokens are required to be unlinkable.

*Recommendation:* PABC-CL. Only PABC-CL provides unlinkability as a core feature, but in some cases PABC-Brands or MS U-Prove might be applicable if it is possible to issue multiple credentials and only use each of them once. However, this required additional bookkeeping and might leak usage information depending on the scenario.

**Inspection or Revocation** Scenarios that need the advanced features inspection or revocation.

*Recommendation:* PABC-CL or PABC-Brands. In the current Reference Implementation only one revocation scheme and one inspection scheme is implemented, which both PABC-CL or PABC-Brands can use. However if other properties of revocation and inspection schemes are needed, one could look into implementing alternative schemes.

**Predicate Proofs over Attributes** Scenarios where predicate proofs over the attributes of a credential are needed.

*Recommendation:* PABC-CL or PABC-Brands.

**Integration with U-Prove** Scenarios where integration with other services that use U-Prove is a requirement.

*Recommendation:* MS U-Prove or PABC-Brands. MS U-Prove can of course support these scenarios, but so can PABC-Brands to some extent if one uses the converter implemented in ABC4Trust.

**Provable Security** Scenarios where one wants the security guarantee that the underlying schemes have a proof of security.

*Recommendation:* PABC-CL. Only PABC-CL has security proofs for all building blocks.

**Unconditional Security** Scenarios where one wants unconditional security for the User when it comes to *untraceability* and *unlinkability.*

*Recommendation:* PABC-CL. PABC-Brands and MS U-Prove support unconditional security for untraceability, but does not offer unlinkability.

Furthermore one should avoid *scope-exclusive pseudonyms* or *inspection* as these features will only give computational security for the User.

**Fast Issuance** Scenarios where issuance needs to be fast, e.g. because credentials are issued often and Users have to wait while they are being issued.

*Recommendation:* PABC-CL or PABC-Brands (maybe MS U-Prove). Both schemes are comparable in performance with a very small advantage to PABC-CL. Unfortunately we don't have performance figures for MS U-Prove, but due to the fact that it can be implemented over elliptic curves it at least has the potential to be significantly faster than PABC-CL.

**Fast Presentation** Scenarios where presentation needs to be fast.

*Recommendation:* PABC-CL or PABC-Brands (maybe MS U-Prove). PABC-CL and PABC-Brands are close in almost every test. Since MS U-Prove can be implemented over elliptic curves it at least has the potential to be significantly faster than the other two schemes.

**Low Communication for Issuance** Scenarios where the amount of data sent between the Issuer and the User needs to be small.

*Recommendation:* PABC-CL (or PABC-Brands if unlinkability is not needed). PABC-Brands does a little better than PABC-CL in these tests, but again the figures are pretty close, so other requirements will likely dictate which scheme to use.

**Low Communication for Presentation** Scenarios where the size of the presentation token needs to be small.

*Recommendation:* PABC-CL. In this case PABC-CL outperforms PABC-Brands by a factor of two in most tests.

**Storage Efficient** Scenarios where the size of the credentials needs to be small. This is especially important in use cases where users need to store credentials on devices with very limited storage space, such as smart cards.

*Recommendation:* PABC-CL or PABC-Brands. PABC-CL credentials are a bit smaller, but not by much, so other requirements will likely dictate which scheme to use.

In Table 22 we present these recommendations in table form. Table 21 lists the possible values for each scheme.

Table 21: Possible recommendation values

| Value | Meaning |
|---|---|
| Yes | This scheme is recommended for this requirement |
| No | This scheme is not recommended for this requirement |
| Maybe | This scheme could be recommended for this requirement and we refer to the description for more information |

Table 22: Summary of recommendations

| Requirement | PABC-CL | PABC-Brands | MS U-Prove |
|---|---|---|---|
| Simple Setup | No | No | Yes |
| Policy Language | Yes | Yes | No |
| Unlinkability | Yes | Maybe | Maybe |
| Inspection or Revocation | Yes | Yes | No |
| Predicate Proofs over Attributes | Yes | Yes | No |
| Integration with U-Prove | No | Maybe | Yes |
| Provable Security | Yes | No | No |
| Unconditional Security | Yes | Maybe | Maybe |
| Fast Issuance | Yes | Yes | Maybe[a] |
| Fast Presentation | Yes | Yes | Maybe[a] |
| Low Communication for Issuance | Yes | Yes | Maybe[a] |
| Low Communication for Presentation | Yes | No | Maybe[a] |
| Storage Efficient | Yes | Yes | Maybe[a] |

[a] No practical benchmark has been performed.

## 5.2 General Recommendations

In this section we give some general recommendations that have a practical impact on the efficiency of operations of both PABC-CL and PABC-Brands. These recommendations are lessons learned from the process of doing the efficiency benchmarks presented earlier. We discuss factors that could influence the running time of issuance and especially presentation, and also give some recommendations for smart card use.

**Efficiency of Issuance** For both PABC-CL and PABC-Brands, the time needed for issuing credentials is impacted by a number of factors, such as the *type of features*, *hardware* and *key size*. The use of advanced issuance features such as *key binding*, *issuance with carry-over attributes*, and the use of *pseudonyms*, all require additional NIZK proofs during issuance. This is clearly an overhead, both in terms of running time, but also in terms of the communication size, and both can be minimised if such features are not used.

**Efficiency of Presentation** Similarly to the factors influencing the efficiency of issuance, the efficiency of presentation can be improved if less features are used during the presentation. Every additional feature besides the proof of possession of a credential will have a negative impact on all types of efficiency such as *running time*, *efficiency of communication*, and the *amount of data that needs to be transmitted*. In this regard, all features such as *proving possession of more than one credential or pseudonym*, *use of key-binding* or *predicate proofs over attributes* will increase the time needed to do a presentation. This effect on presentation is especially important, as this will cause a bad user experience when users will have to wait during presentation, which is expected to happen quite often in most scenarios.

Similarly, the larger the *key size* used for the cryptographic operations are, the longer time it will take to do the cryptographic operations. On the other hand, a larger key size gives a higher level of security, so a careful assessment of the desired level of security and performance needs to be done.

In addition to these factors, the *computational power of the User's device* will impact the time needed to do a presentation. Presentation on a fast computer will take less time than when smart cards are used, e.g. for key binding. Therefore avoiding the use of smart card will result in faster presentation, but of course at the cost of security unless some other secure means of storing the secret keys are implemented.

Finally, the use of *inspection* and *revocation* will result in additional time needed to do the presentation. In particular, for inspection the efficiency of presentation is in direct (negative) relation to the number of *inspectable attributes*, so in case inspection is really needed, one should avoid having more inspectable attributes than absolutely necessary. This is also true from a privacy point of view.

**Storage Efficiency and Smart Cards** Storage efficiency is more relevant for the User than for the Issuer and Verifier, as this may be a deciding factor on the choice of storage medium for User's credentials and related data, such as pseudonyms, non-revocation information, and so on. In many scenarios it may be desired to use smart cards for storing the credentials, as they are considered to provide a higher level of security than computers or smart phones. Nevertheless, smart cards have low computational power and storage capacity, which may limit the number of credentials that can be stored on the card.

Specifically for smart cards one has to carefully consider a number of factors such as the *number of credentials*, the *number of attributes in them*, as well as the additional overhead that comes from storing *revocation-related data* for each revocable credential. An interesting result from our benchmarks is that the key size has a negligible impact on the size of the credentials, which certainly is welcome in this case. However, the important lesson here is that *the choice to use smart card requires consideration of the above factors as well as the requirements of the scenario.*

## References

[Cam+14] J. Camenisch, S. Krenn, A. Lehmann, J. Luna, G. L. Mikkelsen, G. Neven, M. Ø. Pedersen, A. Sabouri, T. Vateva-Gurova and F. Veseli. *D3.1 Scientific Comparison of ABC Protocols. ABC4Trust* - Deliverable to the European Commision. 2014.

[Bic+14a] P. Bichsel, J. Camenisch, M. Dubovitskaya, R. R. Enderlein, S. Krenn, I. Krontiris, A. Lehmann, G. Neven, J. D. Nielsen, C. Paquin, F.-S. Preiss, K. Rannenberg, A. Sabouri and M. Stausholm. *D2.2 Architecture for Attribute-based Credential Technologies - Final Version. ABC4Trust* - Deliverable to the European Commision. 2014.

[Bic+13a] P. Bichsel, J. Camenisch, M. Dubovitskaya, R. R. Enderlein, I. Krontiris, A. Lehmann, G. Neven, J. D. Nielsen, C. Paquin, F.-S. Preiss, K. Rannenberg, M. Stausholm and H. Zwingelberg. *H2.2 ABC4Trust Architecture for Developers. ABC4Trust* - Deliverable to the European Commision. 2013.

[Bic+13b] P. Bichsel, J. Camenisch, M. Dubovitskaya, R. R. Enderlein, A. Lehmann, G. Neven and D. Sommer. *H2.3 ABC4Trust Crypto Architecture. ABC4Trust* - Deliverable to the European Commision. 2013.

[Bic+14b] P. Bichsel, R. R. Enderlein, H. Knudsen, K. Damgård, J. Jensen, J. Luna, J. Nielsen and M. Stausholm. *D4.2 Final Reference Implementation. ABC4Trust* - Deliverable to the European Commision. 2014.

[Jen14] J. L. Jensen. *D4.4 Smartphone Feasibility Analysis. ABC4Trust* - Deliverable to the European Commision. 2014.

[Eur13] European Union Agency for Network and Information Security Agency. *Algorithms, Key Sizes and Parameters Report, version 1.0.* `http://www.enisa.europa.eu/activities/identity-and-trust/library/deliverables/algorithms-key-sizes-and-parameters-report`. 2013.

[Sma12] N. Smart. *ECRYPT II Yearly Report on Algorithms and Keysizes (2011-2012).* `www.ecrypt.eu.org/documents/D.SPA.20.pdf`. 2012.

[HK09] D. Hofheinz and E. Kiltz. 'The Group of Signed Quadratic Residues and Applications'. In: *CRYPTO 09.* Ed. by S. Halevi. Vol. 5677. LNCS. Springer, 2009, pp. 637–653.

[Pai99] P. Paillier. 'Public-Key Cryptosystems Based on Composite Degree Residuosity Classes'. In: *EUROCRYPT 99.* Ed. by J. Stern. Vol. 1592. LNCS. Springer, 1999, pp. 223–238.

[Bon98] D. Boneh. 'The Decision Diffie-Hellman Problem'. In: *Proceedings of the Third International Symposium on Algorithmic Number Theory.* ANTS-III. Springer, 1998, pp. 48–63.

[FS87] A. Fiat and A. Shamir. 'How to Prove Yourself: Practical Solutions to Identification and Signature Problems'. In: *CRYPTO 86.* Ed. by A. M. Odlyzko. Vol. 263. LNCS. Springer, 1987, pp. 186–194.

[Ped91] T. P. Pedersen. 'Non-Interactive and Information-Theoretic Secure Verifiable Secret Sharing'. In: *CRYPTO 91.* Ed. by J. Feigenbaum. Vol. 576. LNCS. Springer, 1991, pp. 129–140.

[FO97] E. Fujisaki and T. Okamoto. 'Statistical Zero Knowledge Protocols to Prove Modular Polynomial Relations'. In: *CRYPTO 97.* Ed. by B. S. K. Jr. Vol. 1294. LNCS. Springer, 1997, pp. 16–30.

[DF02] I. Damgård and E. Fujisaki. 'A Statistically-Hiding Integer Commitment Scheme Based on Groups with Hidden Order'. In: *ASIACRYPT 02.* Ed. by Y. Zheng. Vol. 2501. LNCS. Springer, 2002, pp. 125–142.

[CL02a] J. Camenisch and A. Lysyanskaya. 'A Signature Scheme with Efficient Protocols'. In: *SCN 02.* Ed. by S. Cimato, C. Galdi and G. Persiano. Vol. 2576. LNCS. Springer, 2002, pp. 268–289.

[Bra93] S. Brands. *An Efficient Off-line Electronic Cash System Based On The Representation Problem.* Tech. rep. 1993.

[BL13] F. Baldimtsi and A. Lysyanskaya. 'Anonymous Credentials Light'. In: *ACM CCS 13.* ACM, 2013, pp. 1087–1098.

[Paq13] C. Paquin. *U-Prove Cryptographic Specification V1.1 (Revision 3.* 2013.

[CS03] J. Camenisch and V. Shoup. ' Practical verifiable encryption and decryption of discrete logarithms'. In: *CRYPTO 03.* Ed. by D. Boneh. Vol. 2729. LNCS. Springer, 2003, pp. 126–144.

[Nak+09] T. Nakanishi, H. Fujii, Y. Hira and N. Funabiki. 'Revocable Group Signature Schemes with Constant Costs for Signing and Verifying'. In: *PKC 09*. Ed. by S. Jarecki and G. Tsudik. Vol. 5443. LNCS. Springer, 2009, pp. 463–480.

[CL02b] J. Camenisch and A. Lysyanskaya. 'Dynamic Accumulators and Application to Efficient Revocation of Anonymous Credentials'. In: *CRYPTO 02*. Ed. by M. Yung. Vol. 2442. LNCS. Springer, 2002, pp. 61–76.

[Cam+13] J. Camenisch, M. Dubovitskaya, A. Lehmann, G. Neven, C. Paquin and F.-S. Preiss. 'Concepts and Languages for Privacy-Preserving Attribute-Based Authentication'. In: *IDMAN 13*. Ed. by S. Fischer-Hübner, E. de Leeuw and C. Mitchell. Vol. 396. IFIP Advances in Information and Communication Technology. Springer, 2013, pp. 34–52.

[Pai14] P. Paillier. *Benchmarks of the ABC4Trust Lite Smart Card Implementation by CryptoExperts*. Internal Communication. 2014.