EPID with Malicious Revocation

Olivier Sanders[1] and Jacques Traoré[2]

[1] Orange Labs, Applied Crypto Group, Cesson-Sévigné, France
[2] Orange Labs, Applied Crypto Group, Caen, France

Abstract. EPID systems are anonymous authentication protocols where a device can be revoked by including one of its signatures in a revocation list. Such protocols are today included in the ISO/IEC 20008-2 standard and are embedded in billions of chips, which make them a flagship of advanced cryptographic tools. Yet, their security analysis is based on a model that suffers from several important limitations, which either questions the security assurances EPID can provide in the real world or prevents such systems from achieving their full impact. The most prominent example is the one of revocation lists. Although they could be managed locally by verifiers, which would be natural in most use-cases, the security model assumes that they are managed by a trusted entity, a requirement that is not easily met in practice and that is thus tempting to ignore, as illustrated in the corresponding standard.

In this paper, we propose to revisit the security model of EPID, by removing some limitations of previous works but mostly by answering the following question: what can we achieve when revocation lists are generated by a malicious entity?

Surprisingly, even in this disadvantageous context, we show that it is possible to retain strong properties that we believe to better capture the spirit of EPID systems. Moreover, we show that we can construct very efficient schemes resisting such powerful adversaries by essentially tweaking previous approaches. In particular, our constructions do not require to perform any significant test on the revocation lists during the signature generation process. These constructions constitute the second contribution of this paper.

1 Introduction

1.1 Related Works

Direct Anonymous Attestation (DAA) was introduced by Brickell, Camenisch and Chen [10] as an anonymous authentication mechanism with some controlled linkability features. In such systems, platforms can issue anonymous signatures after being enrolled by an issuer, in a way akin to group signatures [16]. A few years later, Brickell and Li [11] proposed a variant with enhanced revocation features under the name of Enhanced Privacy ID (EPID). Indeed, EPID additionally allows to revoke a platform $\mathcal{P}$ by adding one of its signature to a so-called signature revocation list `SRL`. In such a case, $\mathcal{P}$ will not be able to produce new (valid) signatures on input SRL but we stress, to avoid confusion, that signatures issued by $\mathcal{P}$ with different revocation lists remain anonymous and unrevoked.

Both DAA and EPID systems are among the few advanced[3] cryptographic mechanisms that are widely deployed today. They are indeed embedded in billion of devices [23,30] and have been included in standards, such as ISO/IEC 20008-2 [25].

Surprisingly, the real-world popularity of these mechanisms did not extend to cryptographic literature as only a few papers have been published on these topics. This stands in sharp contrast with a sibling primitive, group signature, that has been extensively studied by the cryptographic community.

Actually, this relatedness with group signature probably explains the lack of academic interest for DAA and EPID schemes. It is indeed tempting to see a DAA or an EPID system as a simple variant of group signature, which lessens the appeal for any contribution in this area. More concretely, a DAA can be seen as a group signature where the opening feature is discarded and replaced by a linking feature that is rather easy to implement. EPID replaces the latter feature by a more intricate revocation mechanism that is nevertheless rather simple to add modularly.

From the algorithmic standpoint, constructing a DAA or an EPID system from a group signature might therefore look trivial. However, from the security model standpoint, removing the opening feature has huge consequences and makes the formalization of security properties much more difficult. In the particular case of EPID, the revocation feature creates additional problems as it is more complex to control in this context. This explains in part the somewhat chaotic history of DAA and EPID security models that we briefly recall here.

The original security model [11] for EPID was based on simulation. Back then, it was a natural choice as DAA [10] also considered simulation-based models. Unfortunately, the security model of [10] was not correct and several attempts [17–19] to fix it failed (see [5] for a discussion on these issues). This probably explains the shift to game-based models for DAA that was followed by EPID in [12]. The latter paper, enhanced with some remarks from [5], is a good starting point to study the security of EPID but unfortunately the resulting model still suffers from several problems that limit the practical assurances it provides in the real world. More recently, a new simulation-based model was proposed by Camenisch *et al* [13] for DAA and later extended to EPID in [26]. Although it implies a cleaner definition of unforgeability, it imposes in practice the same kind of restrictions to the constructions as in [12]. Moreover, [26] suffers from the same limitations regarding anonymity as previous models of EPID, which has important consequences in the real world. All these issues are discussed in details below.

Concretely this means that, as of today, the security assurances provided by mechanisms deployed in billion of devices are not well understood and sometimes rely on some assumptions that seem questionable. This is particularly true in the

[3] By "advanced" we here mean asymmetric mechanisms that go beyond standard signature, encryption and key exchange.

context of EPID as the revocation mechanism imposes very strong constraints on the whole system, which has been underestimated in previous works. In this paper, we will then focus on the case of EPID as it is the most complex one but we note that many of our remarks also apply to standard DAA schemes.

1.2 Our Contributions

The contribution of our paper is actually twofold. First, we propose a security model for EPID with new unforgeability and anonymity definitions that we believe to better capture what we expect from such systems. In particular, we are the first (to our knowledge) to consider the case where revocation lists are not generated by a trusted entity. Such a trusted entity is indeed very convenient from the theoretical standpoint but its existence is not obvious and the way it will proceed to decide in practice which signature should be added to revocation lists is far from clear. To deal with malicious revocation lists is a very challenging task but we argue it is a realistic scenario and it is thus important to understand what we can retain in this case and how to construct schemes that still resist such powerful adversaries. We extensively discuss the issues of previous models in the following paragraphs as it is necessary to understand the rationale behind our new definitions. We choose a game-based approach as we believe it is better suited for complex primitives such as EPID. In particular, we hope that separate experiments will lead to a better understanding of what an EPID system can achieve in our model. Our second contribution consists of two efficient constructions that achieve our new security properties. They share many similarities with existing constructions but also present some differences that we comment at the end of this section.

Issues with previous unforgeability notions. The problems regarding unforgeability (called traceability in [5]) concern both DAA and EPID models and we believe they all find root in the difficulty to properly define what is a valid forgery in this context. Indeed, any model must provide to the adversary $\mathcal{A}$ the ability to own some signing keys (that are called *corrupt*), which inevitably allows $\mathcal{A}$ to issue signatures. We can't therefore rely on something akin to unforgeability for standard digital signature. The idea that seems to be behind previous models is then to mimic security notions of group signatures, but this does not work well in this setting. Indeed, group signature provides an opening algorithm that allows to trace back any group signature to its issuer. Dealing with corrupt keys is then simple, as we have a way to detect if the adversary's forgery is *trivial* (that is, it has been generated using corrupt keys) or not (the signature cannot be linked to a corrupt key). In a DAA or an EPID scheme there is no counterpart of the opening algorithm but one can test if a signature has been generated by a given platform if one knows its secret key. This has led to the following two cases.

In game-based models [5, 12], the experiment assumes that the adversary provides all its signing keys, which enables "opening" of signatures issued with corrupt keys and so to rule out trivial wins by the adversary. In the context of EPID, this constraint can be partially relaxed by alternatively requiring a signature from each malicious platform that has not yet revealed its secret key (see [12]). More concretely, existing game-based EPID models consider a set $\mathcal{U}$ of malicious platforms and requires, for each $i \in \mathcal{U}$, either the signing key sk_i or a signature issued by i. If we set aside the problem of identifying in the real world the set $\mathcal{U}$ of malicious platforms without an opening procedure, it remains to explain why the adversary would ever agree to 1) reveal some of its secret keys and/or 2) return a signature generated with sk_i for every unrevealed key sk_i. The assumption 1) is clearly questionable. The plausibility of the second assumption is not much more obvious because the challenger is not able to determine if the signatures returned by the adversary fulfil the corresponding requirement (the adversary could have generated all the returned signatures using the same key) unless it knows the corresponding secret keys, which brings us back to the first problem. Put differently, this defines a success condition for the adversary that the challenger is not able to verify. To sum up, current game-based security models define an unforgeability experiment that either makes an unrealistic assumption on the adversary behaviour or whose output (success or failure) cannot be computed.

The problem 1) was actually already pointed out in [13] who addressed it by introducing a new simulation-based model taken up in [26]. The authors of these works indeed assume that their ideal functionality knows the secret keys of all corrupt platforms, which solves the problems mentioned above. This is theoretically cleaner as the model no longer expects the adversary to hand over its keys willingly. However, this implies that every construction realising their ideal functionality must provide a way to recover all platforms keys in the security proof (including corrupt ones), which in practice does not seem that different from what happens in game-based models. In particular this suggests the use of zero-knowledge proofs of the platform secret key during `Join`, which either limits the number of concurrent `Join` sessions (if one uses rewinding techniques) or requires additional features such as online-extractability [20], which negatively impacts performance.

Our unforgeability notion. At the heart of the problems we discuss in both cases, there are thus the attempts to identify the signing key that was used to generate the "forgery". In our security model, we therefore try to avoid these identification issues and favour a different reasoning that seems more pragmatic. Concretely, if an adversary owns n signing keys sk_i, then we should not focus on whether its forgeries has been generated using sk_0, sk_1, etc since it necessary leads to the issues we discuss above. What we can do is to successively revoke each of the adversary signing keys by adding every signature it generates to `SRL`, thus ensuring that it will not be able to produce $n+1$ successive signatures. In practice, it means that if a user with sk is revoked via the inclusion of one of his signatures σ in the revocation list, then he cannot produce a new signature σ' unless he gets a new signing key sk'. In the latter case, contrarily to previous models, we do not care if σ' is produced using sk or sk' because this is uncheckable without the knowledge of these keys. All we ensure is that this user will not be able to produce a third signature, regardless of the key it used to produce σ'. We believe this captures what is expected from EPID without making assumptions on the ability to recover adversary's keys.

Issues with previous anonymity notions. The last problem regarding existing models is related to anonymity and is very specific to EPID (not DAA) systems. It concerns the signature revocation list that every model (game-based [12] or simulation-based [26]) assumes to be honestly generated without discussing the plausibility and the concrete consequences of this assumption. More concretely, they all assume that these revocation lists only contain valid signatures, which is enforced in [12, 26] by appointing a trusted entity, called the *revocation manager* that will perform these verifications. We see several problems with this solution.

Firstly, we note that the existence of such trusted entity is far from insignificant as all anonymity assurances would be lost if the revocation manager did not correctly carry out its task. In some way, the revocation manager can be compared to the opening authority of a group signature as anonymity of the whole system relies on his honesty. There is indeed a link between revocation and the ability to trace users as noted in [8] although the case of EPID is more subtle.

Secondly, it implies a very centralized system where each service manager would not be able to directly revoke a platform which misbehaved based on its signature but would have to first contact the central revocation manager. As central revocation must not be treated lightly, this is likely to imply a regulated procedure and the grounds on which the revocation manager will decide the legitimacy of the demand are not clear.

Thirdly, it means that any platform must be aware of the current (and authentic) version of the revocation list $\mathtt{SRL}$ at the time of signing, which may be problematic in some use-cases. In particular, privacy can no longer be ensured if the verifier sends its own revocation list $\mathtt{SRL}$. An alternative solution could be to shift the burden of verifying the elements of $\mathtt{SRL}$ to the platform when generating the signature but this would clearly result in an inefficient signing protocol.

Finally, this requirement is somewhat inconsistent with the ISO/IEC 20008-2 [25] standard. Indeed, the specifications of the EPID system called "Mechanism 3" in [25] clearly states that signature revocation can be *local*, which, according to ISO/IEC 20008-1 [24], means that the signature revocation list may be managed by the verifier itself. However, the same specifications contain the following note:

"To preserve anonymity, it is recommended to have a trusted entity for updating the signature revocation list. If a malicious entity controls the signature revocation list, the anonymity of the signer can be reduced."

The wording is surprising for a standard as it does not sound like a requirement but it yet threatens unspecified problems regarding anonymity if one does not comply with this informal instruction. This results in a blurry situation that is indicative of an EPID paradox. On the one hand, there is a revocation mechanism that is inherently decentralized as any verifier is able to revoke a signature by placing it in its own revocation list. This is clearly the most natural and convenient solution for most use-cases. On the other hand, there are security models that all require a central and trusted entity to manage revocation lists, with the consequences discussed above. In this context, it is extremely tempting to ignore this requirement in practice and, to say the least, even the ISO/IEC recommendation above does not deter us very forcefully from doing so.

We therefore believe it would be better to remove all restrictions on $\mathtt{SRL}$ and to rather consider a model where the adversary $\mathcal{A}$ has a total control on the elements of the revocation lists and where the platform does not have to check any property of $\mathtt{SRL}$ beyond the one that it can correctly be parsed. This will offer much more flexibility to EPID systems by removing the need for a central trusted revocation entity and thus allow decentralized services managing their own revocation lists.

Our first remark is that current proofs strategies no longer work in this new setting. Indeed let us consider the quite standard approach of EPID systems (*e.g.* [11, 12]) where each signature μ contains a pair (h, h^x) where $h \in \mathbb{G}$ is random and x is the platform secret key, leading to an anonymity proof under the DDH assumption. If μ is added to $\mathtt{SRL}$, then any platform generating a signature with this revocation list will have to include a proof that they did not generate (h, h^x). In security proofs of existing models, the challenger does not know x (otherwise DDH is trivial) but it can recognize (h, h^x) from its previous signatures and so correctly simulate the proof. Now, in our model where we allow the adversary to generate $\mathtt{SRL}$, it could replace (h, h^x) by $(h^r, (h^x)^r)$ for any $r \in \mathbb{Z}_p$. In this case, the challenger will be unable to recognize if this pair was generated using x (unless it can itself solve DDH) and so will not be able to correctly answer a signature query, leading to an incorrect simulation.

This example highlights the fact that removing restrictions on $\mathtt{SRL}$ is not just a formalization issue and that it has important consequences on EPID systems. In particular, it gives to any verifier (and so to the adversary) the power of a revocation authority, which is very unusual in privacy-preserving protocols. This leads to a new anonymity experiment that we believe to better capture the security assurances we can retain in the real world.

Our anonymity notion. In our security experiment, we give the adversary a total control on the elements of this list. In return, we need to completely redefine the notion of anonymity we can really achieve without these restrictions. Indeed, we note that nothing now prevents the adversary from testing if a signature σ was issued by a given platform i. It can simply add σ (or some part of it in practice) in $\mathtt{SRL}$ and then query a signature from i with $\mathtt{SRL}$: if a valid signature is returned then σ was not produced by i; else, i will return a failure message or will abort. The main consequence is that our model cannot provide the adversary with an access to a signing oracle that takes as input an identifier i and returns a signature on behalf of this platform. This is not a restriction of our model as it only reflects the real-world situation of EPID where any verifier suspecting a signer to be the issuer of a previous signature can proceed as we have explained to identify it. This is actually the very goal of EPID.

We then need to be much more careful when defining the signing oracle and the success conditions of the adversary in our new experiment. The first novelty is the absence of a user identifier in the inputs of the signing oracle as it prevents any meaningful definition as we explained. Our oracle will instead return a signature using one of the honest keys that are not implicitly revoked by SRL. As the adversary is now oblivious of the users' identifiers we can't expect it to return the identifier of the issuer of some signature. We will instead ask it to link two signatures in a game where it will select two signatures μ_0 and μ_1 and then ask for a new signature μ^* from the issuer of μ_b. It will succeed if it can guess the value of b with probability different from $\frac{1}{2}$, that is, if it can link two signatures with non-trivial probability.

Our constructions. Our new security model has two important consequences in practice. Firstly, we no longer need to extract all platforms secret keys thanks to our new unforgeability experiment. This allows us to define simple `Join` protocols where a platform simply receives a certificate without needing to prove knowledge of its secret key or to use alternative solutions that would enable extraction of this key. Secondly, we now need to deal with malicious revocation lists SRL without performing any test on them. As we explain above, this invalidates the strategy of previous constructions as the adversary is now able to place in SRL elements that were not part of valid signatures. Surprisingly, we can deal with this problem very efficiently by defining a signature algorithm where the platform generates a proof of non-revocation with respect to the hash of (some of) the elements in SRL, instead of the elements themselves. Intuitively, the use of a hash function will prevent the re-randomization strategy of the adversary described above and leads to one of the following two situations. Either the elements of SRL were indeed included in a valid signature or they were generated by the adversary. In the former case, it is easy to know which platforms are revoked, which allows to correctly answer signing queries in the security proof. In the latter case, we show that the forged elements are unlikely to revoke an honest user under the computational Diffie-Hellman assumption. Our first construction reflects these changes and proves that our new security properties can be efficiently achieved. Our second construction is a simple variant of the first one where we further reduce the size of the EPID signatures by using a different building block, at the cost of reintroducing proofs of knowledge in the `Join` protocol.

2 Preliminaries

Bilinear groups. Our constructions require bilinear groups which are constituted of a set of three groups $\mathbb{G}_1$, $\mathbb{G}_2$, and $\mathbb{G}_T$ of order p along with a map, called pairing, $e : \mathbb{G}_1 \times \mathbb{G}_2 \to \mathbb{G}_T$ that is

1. bilinear: for any $g \in \mathbb{G}_1, \widetilde{g} \in \mathbb{G}_2$, and $a, b \in \mathbb{Z}_p$, $e(g^a, \widetilde{g}^b) = e(g, \widetilde{g})^{ab}$;
2. non-degenerate: for any $g \in \mathbb{G}_1^*$ and $\widetilde{g} \in \mathbb{G}_2^*$, $e(g, \widetilde{g}) \neq 1_{\mathbb{G}_T}$;
3. efficient: for any $g \in \mathbb{G}_1$ and $\widetilde{g} \in \mathbb{G}_2$, $e(g, \widetilde{g})$ can be efficiently computed.

As most recent cryptographic papers, we only consider bilinear groups of prime order with *type 3* pairings [22], meaning that no efficiently computable homomorphism is known between $\mathbb{G}_1$ and $\mathbb{G}_2$.

Computational assumptions. The security analysis of our protocols will make use of the following assumptions.

- DL assumption: Given $(g, g^x) \in \mathbb{G}^2$, this assumption states that it is hard to recover x.
- DDH assumption: Given $(g, g^x, g^y, g^z) \in \mathbb{G}^4$, the DDH assumption in the group $\mathbb{G}$ states that it is hard to decide whether $z = x \cdot y$ or z is random.
- CDH assumption: Given $(g, g^x, g^y, \widetilde{g}, \widetilde{g}^x, \widetilde{g}^y)$, the CDH assumption (extended to type 3 bilinear groups) states that it is hard to compute $g^{x \cdot y}$.

3 Specification of EPID

3.1 Syntax

Our EPID system considers three types of entities, an issuer $\mathcal{I}$, platforms $\mathcal{P}$ and verifiers $\mathcal{V}$. When comparing our model with the one of group signature, we will sometimes use the term *user* instead of *platform* to match the terminology of this primitive.

- $\mathtt{Setup}(1^k)$: on input a security parameter 1^k, this algorithm returns the public parameters pp of the system.
- $\mathtt{GKeygen}(pp)$: on input the public parameters pp, this algorithm generates the issuer's key pair $(\mathsf{isk}, \mathsf{ipk})$. We assume that ipk contains pp and so we remove pp from the inputs of all following algorithms.
- $\mathtt{Join}$: this is an interactive protocol between a platform $\mathcal{P}$, taking as inputs ipk, and the issuer $\mathcal{I}$ owning isk. At the end of the protocol, the platform returns either $\perp$ or a signing key sk whereas the issuer does not return anything.
- $\mathtt{KeyRevoke}(\{\mathsf{sk}_i\}_{i=1}^m)$: this algorithm takes as input a set of m platform secret keys sk_i and returns a corresponding key revocation list $\mathtt{KRL}$ containing m elements that will be denoted as $\mathtt{KRL}[i]$, for $i \in [1, m]$.
- $\mathtt{SigRevoke}(\{(\mu_i)\}_{i=1}^n)$: this algorithm takes as input a set of n EPID signatures $\{(\mu_i)\}_{i=1}^n$ and returns a corresponding signature revocation list $\mathtt{SRL}$ containing n elements that will be denoted as $\mathtt{SRL}[i]$, for $i \in [1, n]$.
- $\mathtt{Sign}(\mathsf{ipk}, \mathsf{sk}, \mathsf{m}, \mathtt{SRL})$: this algorithm takes as input the issuer's public key ipk, a platform secret key sk a message m and a signature revocation list $\mathtt{SRL}$ and returns an EPID signature μ.

- $\texttt{Identify}(\mathsf{sk}, t)$: given a platform secret key and an element t from a revocation list $\texttt{SRL}$ (*i.e.* there exists some i such that $t = \texttt{SRL}[i]$), this algorithm returns either 1 (t was generated using sk) or 0.
- $\texttt{Verify}(\mathsf{ipk}, \texttt{KRL}, \texttt{SRL}, \mu, \mathsf{m})$: given an issuer public key ipk, a key revocation list $\texttt{KRL}$, a signature revocation list $\texttt{SRL}$, a signature μ and a message m, this algorithms returns 1 (the signature is valid on m for the corresponding revocation lists) or 0.

Remark 1. We note that our definition of $\texttt{SigRevoke}$ implicitly assumes that no verification is performed on the purported signatures μ_i since this algorithm does not take as input the elements that would be necessary to run $\texttt{Verify}$ (such as the issuer's public key, the message m and the corresponding revocation list). We indeed do not see which security assurances could be provided by such verifications in our context where we do not assume the existence of a trusted entity managing the lists $\texttt{SRL}$ and therefore choose this simpler definition. In practice, it will then be up to each verifier (even a malicious one) to decide how to construct the revocation lists. This will be captured by our security model and our constructions will be secure even in this very strong model.

3.2 Security Model

As in [12], we expect an EPID system to be correct, unforgeable and anonymous. However, the comparison stops here as our definitions of unforgeability and anonymity strongly differ from the ones of previous works. We refer to Section 1.2 for a discussion on this matter and here only formalize the intuition provided in our introduction.

Correctness. Here, we essentially follow [12] and require that a signature generated with a platform signing key sk will be considered as valid by the $\texttt{Verify}$ algorithm as long as sk was not revoked, either explicitly using $\texttt{KRL}$ or implicitly using $\texttt{SRL}$. However, we note a minor problem in the correctness definition of [12] because the latter assumes that $\texttt{SRL}$ contains signatures whereas it only contains parts of signatures in their concrete constructions. To match reality, we do not assume anything regarding the elements placed in $\texttt{SRL}$ but instead use in our formal definition of correctness the $\texttt{Identify}$ algorithm that tests whether an element $\texttt{SRL}[i]$ was generated using a given signing key or not. Additional requirements on this $\texttt{Identify}$ algorithm will be specified by our other security properties. This leads to the following formal requirement: for all signing key sk_i generated using $\texttt{Join}$, $\texttt{KRL}$ generated using $\texttt{KeyRevoke}$ and $\texttt{SRL}$ generated using $\texttt{SigRevoke}$:

$$\begin{aligned}&\texttt{Verify}(\mathsf{ipk}, \texttt{KRL}, \texttt{SRL}, \texttt{Sign}(\mathsf{ipk}, \mathsf{sk}_i, m, \texttt{SRL}), m) = 1\\ &\Leftrightarrow \mathsf{sk}_i \notin \texttt{KRL} \wedge \forall j : \texttt{Identify}(\mathsf{sk}_i, \texttt{SRL}[j]) = 0\end{aligned}$$

Unforgeability. Our unforgeability experiment is defined in Figure 1, where c (resp. d) is a counter indicating the number of corrupt users created by $\mathcal{A}$ (resp. of signatures issued by the adversary $\mathcal{A}$) at the current time and where $\mathcal{A}$ may query the following oracles:

- $\mathcal{O}\mathtt{Add}(k)$ is an oracle that is used by the adversary to add a new honest platform k. A signing key sk_k is then generated for this platform using the `Join` protocol but nothing is returned to $\mathcal{A}$.
- $\mathcal{O}\mathtt{Join}_{cor}()$ is an oracle playing the issuer's side of the `Join` protocol. It is used by $\mathcal{A}$ to add a new corrupt platform. Each call to this oracle increases by 1 the current value of c ($c = c + 1$).
- $\mathcal{O}\mathtt{Cor}(k)$ is an oracle that returns the signing key sk_k of an honest platform k and also adds it to a list $\mathcal{K}$ that is initially set as empty.
- $\mathcal{O}\mathtt{Sign}(k, \mathtt{SRL}, m)$ is an oracle that is used by $\mathcal{A}$ to query a signature from the platform k on a message m with a signature revocation list $\mathtt{SRL}$. We define $\mathcal{S}$ as the set of all signatures returned by this oracle.

```
$\mathtt{Exp}^{unf}_{\mathcal{A}}(1^k)$ – Unforgeability Security Game
 1. $c, d \leftarrow 0$
 2. $\mathcal{S}_{\mathcal{A}} \leftarrow \emptyset$
 3. $pp \leftarrow \mathtt{Setup}(1^k)$
 4. $(\mathsf{isk}, \mathsf{ipk}) \leftarrow \mathtt{GKeygen}(pp)$
 5. $\mathtt{while}\ d \leq c$:
      – $\mathtt{SRL} \leftarrow \mathtt{SigRevoke}(\mathcal{S}_A)$
      – $(\mu, m) \leftarrow \mathcal{A}^{\mathcal{O}\mathtt{Add}, \mathcal{O}\mathtt{Join}_{cor}, \mathcal{O}\mathtt{Cor}, \mathcal{O}\mathtt{Sign}}(\mathtt{SRL}, \mathsf{ipk})$
      – $\mathtt{KRL} \leftarrow \mathtt{KeyRevoke}(\mathcal{K})$
      – if $1 = \mathtt{Verify}(\mathsf{ipk}, \mathtt{KRL}, \mathtt{SRL}, \mu, m) \wedge \mu \notin \mathcal{S}_{\mathcal{A}} \cup \mathcal{S}$
        then $d = d + 1 \wedge \mathcal{S}_{\mathcal{A}} \leftarrow \mathcal{S}_A \cup \{\mu\}$
 6. Return 1
```

Fig. 1. Unforgeability Game for EPID Signature

An EPID system is unforgeable if $\mathtt{Adv}^{unf}(\mathcal{A}) = \Pr[\mathtt{Exp}^{unf}_{\mathcal{A}}(1^k) = 1]$ is negligible for any $\mathcal{A}$. Concretely, an adversary owning $d - 1$ corrupt keys succeeds if it has generated d valid (and distinct) signatures despite systematic revocation of the signatures it has previously issued. For sake of simplicity, we chose to place each key corrupted through a $\mathcal{O}\mathtt{Cor}$ query on a key revocation list `KRL`. We could proceed differently by offering to the adversary the ability to decide which one should be revoked with `KRL` but we believe that it would only introduce unnecessary complexity to our model. We also implicitly assume that our `while` loop aborts after some polynomial number of iterations, in which case the experiment returns 0.

Anonymity. Our formal anonymity experiment described in Figure 2 makes use of the following oracles. An EPID system is anonymous if $\mathtt{Adv}^{an}(\mathcal{A}) = |\Pr[\mathtt{Exp}^{an-1}_{\mathcal{A}}(1^k) = 1] - \Pr[\mathtt{Exp}^{an-0}_{\mathcal{A}}(1^k) = 1]|$ is negligible for any $\mathcal{A}$.

- $\mathcal{O}\texttt{Join}_{hon}()$ is an oracle playing the user's side of the $\texttt{Join}$ protocol and is then used by $\mathcal{A}$, playing the issuer, to add a new honest platform. Each call generates a platform secret key sk that is kept secret by the challenger.
- $\mathcal{O}\texttt{Sign}^*(\texttt{SRL}, m)$ is an oracle used by $\mathcal{A}$ to query a signature on m from an honest platform that is not implicitly revoked by $\texttt{SRL}$. The challenger of the experiment randomly selects a signing key sk among those that are not revoked by $\texttt{SRL}$ (that is, the secret keys sk_i such that $\texttt{Identify}(\mathsf{sk}_i, \texttt{SRL}[j]) = 0$ for every element $\texttt{SRL}[j]$ of $\texttt{SRL}$) and then return the output of $\texttt{Sign}(\mathsf{ipk}, \mathsf{sk}, m, \texttt{SRL})$. We define $\mathcal{S}$ as the set of all signatures returned by this oracle concatenated with the key sk used to generate them.
- $\mathcal{O}\texttt{Cor}^*(\mu)$ is an oracle that returns the signing key sk_k used to generate the signature μ if there is some pair $(\mu||\mathsf{sk}_k)$ in $\mathcal{S}$. Else, it returns $\perp$. Once the adversary has returned the two challenge signatures μ_0 and μ_1 (step 3 of the anonymity game), we slightly modify the behaviour of this oracle to prevent unintentional failure of the adversary. Indeed, the adversary could inadvertently query $\mathcal{O}\texttt{Cor}^*$ on a signature μ generated using the same key as μ_0 or μ_1, making it lose the game at step 8. After step 3, this oracle therefore returns $\perp$ if queried on a signature μ generated using sk_i, for $i \in \{0, 1\}$. We note that we nevertheless still need the success condition of step 8 as the adversary could have queried $\mathcal{O}\texttt{Cor}^*$ on such a signature μ before outputting μ_0 and μ_1. However, in such a case, the adversary knows that μ_0 and μ_1 are illicit choices before returning them and its failure will then no longer be unintended.

```
$\texttt{Exp}_{\mathcal{A}}^{an-b}(1^k)$ – Anonymity Security Game
 1. $pp \leftarrow \texttt{Setup}(1^k)$
 2. $(\mathsf{isk}, \mathsf{ipk}) \leftarrow \texttt{GKeygen}(pp)$
 3. $(\mu_0, \mu_1, m, \texttt{SRL}) \leftarrow \mathcal{A}^{\mathcal{O}\texttt{Join}_{hon}, \mathcal{O}\texttt{Cor}^*, \mathcal{O}\texttt{Sign}^*}(\mathsf{isk})$
 4. if no entry $(\mu_i, \mathsf{sk}_i)$ in $\mathcal{S}$ for $i \in \{0, 1\}$, then return 0
 5. if $\exists j$ and $i \in \{0, 1\}$: $\texttt{Identify}(\mathsf{sk}_i, \texttt{SRL}[j]) = 1$, then return 0
 6. $\mu^* \leftarrow \texttt{Sign}(\mathsf{ipk}, \mathsf{sk}_b, m, \texttt{SRL})$
 7. $b' \leftarrow \mathcal{A}^{\mathcal{O}\texttt{Join}_{hon}, \mathcal{O}\texttt{Cor}^*, \mathcal{O}\texttt{Sign}^*}(\mathsf{isk}, \mu^*)$
 8. if $\exists i \in \{0, 1\}$: $\mathsf{sk}_i$ leaked during a $\mathcal{O}\texttt{Cor}^*$ query, then return 0
 9. Return $(b = b')$
```

Fig. 2. Anonymity Game for EPID Signature

4 Our First Construction

4.1 Description

Intuition. The goal of our first construction is to highlight the fact that security in our strong model can be achieved quite efficiently with a few tweaks to previous approaches. Indeed, a platform producing an EPID signature essentially does two things. Firstly, it proves that it is a legitimate platform that has been correctly enrolled by the issuer during a $\texttt{Join}$ protocol. Secondly, it proves that it has not generated any of the signatures in the revocation list $\texttt{SRL}$.

The first part is common to many privacy-preserving signatures such as group signature [3], DAA [10], EPID [11], multi-show anonymous credentials [21], etc. It essentially consists in proving knowledge of a signature issued by $\mathcal{I}$ on a secret value s generated by the platform. In bilinear groups, there are many signature schemes [6, 14, 27] that have been specifically designed for this purpose but we will not use them in our first construction for a purely technical reason. Indeed, a benefit of our new model over previous works is that we do not inherently need a way to extract all platform secrets. In particular, we do not need zero-knowledge proofs of s during $\texttt{Join}$ and thus to limit the number of concurrent $\texttt{Join}$ sessions. Using one of the signature schemes cited above would however force us to reintroduce zero-knowledge proofs as we would need, in the security proof, knowledge of the secret scalars to query the signing oracle of the corresponding EUF-CMA experiment. To avoid this problem, we will instead use a signature scheme able of signing group elements (in $\mathbb{G}_1$), and more specifically the one from [21] which is particularly well suited for anonymous constructions. We nevertheless stress that this choice is only driven by our will to highlight the differences between our model and the previous ones. In particular, the scheme from [21] can be replaced by one from [6, 14, 27] in the following construction as we will show in Section 5.

For the second part, we will here follow an approach very similar to the one from [12] but with some adjustments that are made necessary by the absence of a trusted entity to construct revocation lists $\texttt{SRL}$. As in [12], we indeed implicitly include a pair (h, h^s) to each signature generated with a secret s to enable efficient proof of non-revocation. The latter will be implemented with the protocol from [15]. However, as we explain in Section 1, we cannot just add the pair (h, h^s) to the revocation list as it will prevent us in the security proof from correctly simulating the answers from honest platforms. For the latter, we need a way to detect such elements without knowing s or $\widetilde{g}^s \in \mathbb{G}_2$. In our scheme, this will be done by constructing $h \in \mathbb{G}_1$ as a hash output $H(\texttt{str})$ for some random string $\texttt{str}$ and so by replacing (h, h^s) by $(\texttt{str}, h^s)$.This way, our security reduction faces two cases for each element $(\texttt{str}, h^s)$ in $\texttt{SRL}$. Either $\texttt{str}$ was used in a previous signing query or the reduction never used it before. In the first case, the reduction knows that the key s is revoked if and only if the corresponding entry in $\texttt{SRL}$ is exactly the pair $(\texttt{str}, h^s)$ used in a previous signature. In the second case, the reduction knows that the key s is not revoked unless the adversary managed to forge a BLS signature [7] and thus to break the CDH assumption. In all cases, this means that we can prove anonymity under DDH without performing any verification on $\texttt{SRL}$ nor making any assumption on the way $\texttt{SRL}$ is constructed.

We provide the formal description of our scheme below but first recall some elements on the signature on equivalence classes from [21].

FHS Signature. In [21], Fuchsbauer, Hanser and Slamanig introduce a signature on equivalence classes for the following equivalence relation on tuples in $\mathbb{G}_1^n$: $(M_1, \ldots, M_n)$ is in the same equivalence class as $(N_1, \ldots, N_n)$ if there exists a scalar a such that $N_i = M_i^a$ for all $i \in [1, n]$. The point of their signature is that anyone, given a signature τ on some $(M_1, \ldots, M_n)$, can derive a signature τ' on a new representative $(N_1, \ldots, N_n)$ of this class. By correctly computing the latter values, one can ensure that $(\tau, (M_1, \ldots, M_n))$ and $(\tau', (N_1, \ldots, N_n))$ are unlinkable under the DDH assumption. In this paper, we will only need the case $n = 2$.

- $\texttt{Setup}(1^\lambda)$: outputs parameters pp containing the description of type-3 bilinear groups $(\mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T, e)$, with generators $(g, \widetilde{g}) \in \mathbb{G}_1 \times \mathbb{G}_2$.
- $\texttt{Keygen}(pp)$: generates two random scalars x_1 and x_2 and sets sk as (x_1, x_2) and pk as $(\widetilde{A}_1, \widetilde{A}_2) = (\widetilde{g}^{x_1}, \widetilde{g}^{x_2})$.
- $\texttt{Sign}(\mathsf{sk}, (M_1, M_2))$: selects a random scalar t and computes the signature $(\tau_1, \tau_2, \widetilde{\tau}) \leftarrow ((M_1^{x_1} M_2^{x_2})^t, g^{1/t}, \widetilde{g}^{1/t})$ on the representative $(M_1, M_2) \in \mathbb{G}_1^2$.
- $\texttt{Verify}(\mathsf{pk}, (M_1, M_2), (\tau_1, \tau_2, \widetilde{\tau}))$: accepts $(\tau_1, \tau_2, \widetilde{\tau}) \in \mathbb{G}_1^2 \times \mathbb{G}_2$, a signature on $(M_1, M_2) \neq (1, 1)$, if $e(\tau_1, \widetilde{\tau}) = e(M_1, \widetilde{A}_1) \cdot e(M_2, \widetilde{A}_2)$ and $e(\tau_2, \widetilde{g}) = e(g, \widetilde{\tau})$ hold.

One can note that if $(\tau_1, \tau_2, \widetilde{\tau})$ is valid on (M_1, M_2), then $(\tau_1^{r \cdot t'}, \tau_2^{1/t'}, \widetilde{\tau}^{1/t'})$ is valid on (M_1^r, M_2^r) for all pairs $(r, t') \in \mathbb{Z}_p^2$.

Construction.

- $\texttt{Setup}(1^k)$: this algorithm returns the public parameters pp containing the description of a bilinear group $(e, \mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T)$ along with two generators $g \in \mathbb{G}_1$ and $\widetilde{g} \in \mathbb{G}_2$ and two hash functions $H : \{0,1\} \rightarrow \mathbb{Z}_p$ and $H' : \{0,1\} \rightarrow \mathbb{G}_1$.
- $\texttt{GKeygen}(pp)$: this algorithm generates a key pair $\mathsf{isk} \leftarrow (x_1, x_2)$ and $\mathsf{ipk} \leftarrow (\widetilde{A}_1, \widetilde{A}_2)$ for the FHS signature.
- $\texttt{Join}$: this protocol starts when a platform sends g^s to the issuer for some random secret s. $\mathcal{I}$ then generates a FHS signature $\tau \leftarrow (\tau_1, \tau_2, \widetilde{\tau})$ on the pair (g, g^s) and returns τ to the platform. The latter can then verify τ using ipk and store $\mathsf{sk} \leftarrow (s, \tau)$ (if τ is valid) or return $\perp$.
- $\texttt{KeyRevoke}(\{\mathsf{sk}_i\}_{i=1}^m)$: this algorithm takes as input a set of m platform secret keys $\mathsf{sk}_i = (s^{(i)}, \tau^{(i)})$ and returns a corresponding key revocation list $\texttt{KRL}$ with $\texttt{KRL}[i] = \mathsf{sk}_i$, for $i \in [1, m]$.
- $\texttt{SigRevoke}(\{(\mu_i)\}_{i=1}^n)$: this algorithm takes as input a set of n EPID signatures $\{(\mu_i)\}_{i=1}^n$ and parses each of them as $((\tau_1^{(i)}, \tau_2^{(i)}, \widetilde{\tau}^{(i)}), (M_1^{(i)}, M_2^{(i)}), h_2^{(i)}, \pi^{(i)})$. It then returns a signature revocation list $\texttt{SRL}$ such that $\texttt{SRL}[i] = (M_1^{(i)}, h_2^{(i)})$, for $i \in [1, n]$.
- $\texttt{Sign}(\mathsf{ipk}, \mathsf{sk}, \mathsf{m}, \texttt{SRL})$: to issue a signature on a message m with a revocation list $\texttt{SRL}$, a platform owning a secret key $(s, (\tau_1, \tau_2, \widetilde{\tau}))$ proceeds as follows:
  1. it first re-randomizes its FHS signature by selecting two random scalars (r, t) and computing $(\tau_1', \tau_2', \widetilde{\tau}') \leftarrow (\tau_1^{r \cdot t}, \tau_2^{1/t}, \widetilde{\tau}^{1/t})$ along with a new representative $(M_1, M_2) = (g^r, g^{r \cdot s})$ of (g, g^s);

2. it computes $(h_1, h_2) \leftarrow (H'(g^r), h_1^s)$;
3. for all $i \in [1, n]$, it parses $\texttt{SRL}[i]$ as $(M_1^{(i)}, h_2^{(i)})$ and computes $h_1^{(i)} \leftarrow H'(M_1^{(i)})$;
4. it generates a proof π of knowledge of s such that $M_2 = M_1^s$ and $h_2 = h_1^s$ and that $(h_1^{(i)})^s \neq h_2^{(i)}$ for all $i \in [1, n]$ using the protocol from [15]. More specifically, it selects random scalars r_i and computes $C_i = ((h_1^{(i)})^s / h_2^{(i)})^{r_i}$. If $\exists i \in [1, n]$ such that $C_i = 1$, then it returns $\perp$. Else, it selects $k, \{k_{i,1}, k_{i,2}\}_{i=1}^n \xleftarrow{\$} \mathbb{Z}_p^{2n+1}$ and computes $(K_{0,1}, K_{0,2}) \leftarrow (M_1^k, h_1^k)$ along with $(K_{i,1}, K_{i,2}) \leftarrow ((h_1^{(i)})^{k_{i,1}} \cdot (1/(h_2^{(i)})^{k_{i,2}}, h_1^{k_{i,1}} \cdot (1/h_2)^{k_{i,2}})$. It then computes

   $$c = H(\tau_1', \tau_2', \widetilde{\tau}', M_1, M_2, h_1, h_2, \{C_i\}_{i=1}^n, \{K_{i,1}, K_{i,2}\}_{i=0}^n, \mathsf{m}).$$

   along with $z = k + c \cdot s$ and $(z_{i,1}, z_{i,2}) = (k_{i,1} + c \cdot s \cdot r_i, k_{i,2} + c \cdot r_i)$. The proof π is then set as $(\{C_i\}_{i=1}^n, c, z, \{z_{i,1}, z_{i,2}\}_{i=1}^n)$;
5. it returns the signature $\mu = ((\tau_1, \tau_2, \widetilde{\tau}), (M_1, M_2), h_2, \pi)$.

- $\texttt{Identify}(\mathsf{sk}, t)$: this algorithm parses sk as $(s, (\tau_1, \tau_2, \widetilde{\tau}))$ and t as (M_1, h_2), and returns 1 if $h_2 = H'(M_1)^s$ and 0 otherwise.
- $\texttt{Verify}(\mathsf{ipk}, \texttt{KRL}, \texttt{SRL}, \mu, \mathsf{m})$: This algorithm parses μ as $((\tau_1, \tau_2, \widetilde{\tau}), (M_1, M_2), h_2, \pi)$, each $\texttt{KRL}[i]$ as $(s^{(i)}, (\tau_1^{(i)}, \tau_2^{(i)}, \widetilde{\tau}^{(i)}))$ for $i \in [1, m]$ and each $\texttt{SRL}[i]$ as $(M_1^{(i)}, h_2^{(i)})$ for $i \in [1, n]$. It then returns 1 if all the following conditions hold and 0 otherwise.
  1. $e(\tau_1, \widetilde{\tau}) = e(M_1, \widetilde{A}_1) \cdot e(M_2, \widetilde{A}_2) \wedge e(\tau_2, \widetilde{g}) = e(g, \widetilde{\tau})$;
  2. $\forall i \in [1, m]$, $\texttt{Identify}(\texttt{KRL}[i], (M_1, h_2)) = 0$;
  3. $\forall i \in [1, n]$, $C_i \neq 1$;
  4. $c = H(\tau_1, \tau_2, \widetilde{\tau}, M_1, M_2, h_1, h_2, \{C_i\}_{i=1}^n, \{K_{i,1}, K_{i,2}\}_{i=0}^n, \mathsf{m})$, where $h_1 \leftarrow H'(M_1)$, $(K_{0,1}, K_{0,2}) \leftarrow (M_1^z \cdot M_2^{-c}, h_1^z \cdot h_2^{-c})$ and $(K_{i,1}, K_{i,2}) \leftarrow (C_i^{-c} \cdot [(h_1^{(i)})^{z_{i,1}} / (h_2^{(i)})^{z_{i,2}}], h_1^{z_{i,1}} / (h_2^{z_{i,2}}))$ with $h_1^{(i)} = H'(M_1^{(i)})$.

Correctness. The first step of the verification protocol checks that $(\tau_1, \tau_2, \widetilde{\tau})$ is a valid FHS signature on the representative (M_1, M_2). The second step checks that the issuer of μ is not revoked by $\texttt{KRL}$. It is easy to verify that this step fails if μ was generated using a secret s in $\texttt{KRL}$. The third step checks that s has not been used to generate one of the elements in $\texttt{SRL}$. By construction, we would necessarily have $C_i = 1$ in this case. The last step checks the validity of π. The proof is a simple combination of the Schnorr's protocol [29] and the Camenisch's and Shoup's one [15]. For a valid signature μ, one can indeed see that

- $M_1^z \cdot M_2^{-c} = M_1^{k+c \cdot s} \cdot M_2^{-c} = M_1^k$,
- $h_1^z \cdot h_2^{-c} = h_1^{k+c \cdot s} \cdot h_2^{-c} = h_1^k$,
- $K_{i,1} = \frac{(h_1^{(i)})^{z_{i,1}}}{(h_2^{(i)})^{z_{i,2}}} \cdot C_i^{-c} = \frac{(h_1^{(i)})^{k_{i,1}+c \cdot s \cdot r_i}}{(h_2^{(i)})^{k_{i,2}+c \cdot r_i}} \cdot \frac{(h_1^{(i)})^{-c \cdot s \cdot r_i}}{(h_2^{(i)})^{-c \cdot r_i}} = \frac{(h_1^{(i)})^{k_{i,1}}}{(h_2^{(i)})^{k_{i,2}}}$,
- $K_{i,2} = \frac{h_1^{z_{i,1}}}{h_2^{z_{i,2}}} = \frac{h_1^{k_{i,1}+c \cdot s \cdot r_i}}{h_2^{k_{i,2}+c \cdot r_i}} = \frac{h_1^{k_{i,1}}}{h_2^{k_{i,2}}}$,

which ensures that the last condition is satisfied.

Remark 2. As we explain at the beginning of this section, we need to generate h_1 as some hash output. We could use any random string `str` but the latter would then have to be added to μ. As several elements of μ are already random, we arbitrarily choose to derive h_1 from one of them. We selected M_1 that is simply considered as a bitstring by the hash function H' but most other elements of μ (or combinations of them) would work.

Theorem 3. *In the random oracle model, our EPID system is*

- *unforgeable under the* DL *assumption, the* CDH *assumption and the EUF-CMA security of the FHS signature if π is a sound zero-knowledge proof system.*
- *anonymous under the* CDH *and* DDH *assumptions if π is a zero-knowledge proof system.*

4.2 Security Proofs

Unforgeability. Let $\mathcal{A}$ be an adversary succeeding against the unforgeability of our scheme with probability ϵ. We recall that $\mathcal{A}$ succeeds if it can issue $c+1$ valid signatures $\{\mu_i\}_{i=1}^{c+1}$ despite systematic revocation of its previous signatures, where c is the number of corrupted keys it has created. In this proof, an *honest* key refers to a key that was generated during a $\mathcal{O}\texttt{Add}$ query and that has never been involved in a $\mathcal{O}\texttt{Cor}$ query. We distinguish the following three types of forgeries:

- (type 1) $\mathcal{A}$ has queried $\mathcal{O}\texttt{Sign}$ with a revocation list $\texttt{SRL}$ such that $\exists i$: $\texttt{Identify}(\mathsf{sk}, \texttt{SRL}[i]) = 1$ for some honest key sk and yet none of the previous signatures returned by $\mathcal{O}\texttt{Sign}$ contains the pair in $\texttt{SRL}[i]$;
- (type 2) the previous case does not occur and $\exists i \in [1, c+1]$ such that μ_i can be parsed as $((\tau_1^{(i)}, \tau_2^{(i)}, \widetilde{\tau}^{(i)}), (M_1^{(i)}, M_2^{(i)}), h_2^{(i)}, \pi^{(i)})$ with $\texttt{Identify}(\mathsf{sk}, (M_1^{(i)}, h_2^{(i)})) = 1$ for some honest key sk;
- (type 3) none of the previous cases occur.

The first case intuitively deals with malicious verifiers that would introduce illicit elements in $\texttt{SRL}$, that is, elements that were not part of a previous EPID signature and that can yet be associated with some honest user. We show that this case implies an attack against the CDH assumption.

The second case is an attack against what would be called non-frameability in a group signature [4] paper, that is, $\mathcal{A}$ has managed to produce a signature that can be "traced back" to some honest user. We show in lemma 5 that $\mathcal{A}$ can be used against the DL assumption in such case.

Else, we will show that $\mathcal{A}$ has necessarily produced a signature on a new class of equivalences and has thus broken the security of FHS signatures.

Lemma 4. *Let $q_{H'}$ (resp. q_a) be a bound on the number of oracle queries to H' (resp. $\mathcal{O}$*`Add`*) made by the adversary $\mathcal{A}$. Then any type 1 adversary succeeding with probability ϵ can be converted into an adversary against the* CDH *assumption succeeding with probability at least $\frac{\epsilon}{q_{H'} \cdot q_a}$.*

Proof. Let $(g, g^x, g^y, \widetilde{g}, \widetilde{g}^x, \widetilde{g}^y)$ be a CDH challenge, we construct a reduction $\mathcal{R}$ using $\mathcal{A}$ to compute g^{xy}. In our proof, $\mathcal{A}$ will submit a set of $n < q_{H'}$ different strings $\{\mathtt{str}_i\}_{i=1}^n$ to the hash oracle H'. $\mathcal{R}$ then selects a random index $i^* \in [1, q_{H'}]$ and proceeds as follows to answer such queries. First it checks if $\mathtt{str}_i$ has already been queried in which case it returns the same answer as previously. Else, it selects and stores a random $r \xleftarrow{\$} \mathbb{Z}_p$ and returns g^r if $i \neq i^*$ and g^y if $i = i^*$. In the experiment, $\mathcal{R}$ makes a guess on the identifier $k^* \in [1, q_a]$ of the platform illicitly revoked by the adversary's revocation list `SRL` and generates the issuer's key pair as usual. Upon receiving a query on k to $\mathcal{O}$`Add`, it proceeds as usual except if $k = k^*$, in which case it implicitly sets the platform secret value as x. Thanks to isk, it can then handle any $\mathcal{O}$`Add` and $\mathcal{O}\mathtt{Join}_{cor}$ query. $\mathcal{R}$ can also handle any $\mathcal{O}$`Cor` query except the one on k^* in which case it aborts.

To answer a signing query with revocation list `SRL`, $\mathcal{R}$ first uses its knowledge of $\widetilde{g}^x$ to test whether `SRL` contains a pair (h_0, h_2) with $e(H'(h_0), \widetilde{g}^x) = e(h_2, \widetilde{g})$. If no such pair is found, then $\mathcal{R}$ answers a signing query for k^* as follows (the case $k \neq k^*$ is trivial). $\mathcal{R}$ re-randomizes the FHS certificate and the representative (M_1, M_2) as usual. In the very unlikely event where $M_1 = \mathtt{str}_{i^*}$, it simply chooses a different representative of (M_1, M_2). This means that it knows in all cases the scalar r such that $H'(M_1) = g^r$ and can thus compute $h_2 = (g^x)^r$. It then simulates the zero-knowledge proof π and returns the resulting EPID signature μ. Now let us assume that `SRL` contains a pair (h_0, h_2) with $e(H'(h_0), \widetilde{g}^x) = e(h_2, \widetilde{g})$. This event occurs if the guess on k^* was right as we assume type 1 adversary. Moreover, we also know that this pair has never been used by $\mathcal{R}$ to answer a $\mathcal{O}$`Sign` query. This means that h_0 has never been used by $\mathcal{R}$ as h_2 is deterministically computed from it. The value h_0 has then been queried by $\mathcal{A}$ to H' and there are two cases. Either $\mathcal{R}$ returned g^y, which means that $h_2 = g^{x \cdot y}$, or $\mathcal{R}$ returned some element g^r for a random r and it aborts.

In the former case, $\mathcal{R}$ has thus broken the CDH assumption. This occurs if both the guess on k^* and the one on i^* are correct and so with probability at least $\frac{\epsilon}{q_{H'} \cdot q_a}$.

Lemma 5. *Let q_a be a bound on the number of $\mathcal{O}$*`Add` *queries made by $\mathcal{A}$. Then, any type 2 adversary succeeding with probability ϵ can be converted into an adversary against the* DL *assumption succeeding with probability at least $\frac{\epsilon}{q_a \cdot (c+1)}$.*

Proof. Let (g, g^x) be a DL instance, we construct a reduction $\mathcal{R}$ using $\mathcal{A}$ to recover x. First, $\mathcal{R}$ makes a guess on the identifier k^* of the honest platform that will be associated with the adversary's forgery and on the index i^* of the forgery. It then generates the issuer's key pair $(\mathsf{isk}, \mathsf{ipk})$ and is thus able to answer any `Join` query. It answers any oracle query on a string `str` to the hash function H' by generating (and storing) a random scalar r and returning g^r, unless `str` has already been queried in which case it returns the original answer. Upon receiving a $\mathcal{O}\texttt{Add}$ query on k, it proceeds as usual except when $k = k^*$. In this case, it acts as if the platform secret were x. Note that this is not a problem as the issuer generates a signature τ directly on (g, g^x) instead of x. Regarding $\mathcal{O}\texttt{Cor}$ queries, it simply forwards the secret key except if $k = k^*$, in which case $\mathcal{R}$ aborts. However, the latter case does not occur if $\mathcal{R}$ guess on k^* is valid.

Let $\texttt{SRL}$ be the revocation list associated with some $\mathcal{O}\texttt{Sign}$ query on k^*. Since we here consider type 2 adversary we know that, $\forall i \in [1, n]$, either $\texttt{SRL}[i]$ is a part of a signature previously issued by $\mathcal{R}$ or that it does not revoke one of the honest platforms. In the former case, $\mathcal{R}$ checks if $\texttt{SRL}[i]$ was used to issue a signature on behalf of k^* in which case it returns $\perp$. Else, it knows that k^* can produce a signature. To generate μ, it re-randomizes the certificate τ into τ' along with the representative (g, g^x) into (M_1, M_2). It then defines $H'(M_1) = g^r$ for some random r unless in the very unlikely event where H' has already been queried on M_1, in which case $\mathcal{R}$ simply recovers the corresponding scalar r. In all cases, $\mathcal{R}$ is able to compute a valid $h_2 = (g^x)^r$. It then only remains to simulate the proof π and to return the resulting signature μ.

Now $\mathcal{R}$ extracts from the proof of knowledge contained in μ_{i^*} the secret key s^* used by the adversary to generate this signature (recall that μ_{i^*} must differ from the signatures issued by $\mathcal{R}$). If the guesses on both k^* and i^* are correct, then $s^* = x$ thanks to the soundness of the proof. $\mathcal{R}$ can thus solve the DL problem with probability at least $\frac{1}{q_a \cdot (c+1)}$.

Lemma 6. *Any type 3 adversary succeeding with probability ϵ can be converted into an adversary against the EUF-CMA security of FHS signatures succeeding with probability at least $\frac{\epsilon}{c+1}$.*

Proof. Our reduction $\mathcal{R}$ receives a FHS public key from the challenger of the EUF-CMA security and sets it as the issuer's public key ipk. $\mathcal{R}$ generates as usual the secret values for honest platforms and is thus able to handle any $\mathcal{O}\texttt{Sign}$ or $\mathcal{O}\texttt{Cor}$ query. It can also use its FHS signing oracle to address any $\mathcal{O}\texttt{Add}$ and $\mathcal{O}\texttt{Join}$ query. The simulation is then perfect and $\mathcal{A}$ eventually outputs, with probability ϵ, $c+1$ EPID signatures μ_i fulfilling the type 3 requirements. Each of them contains a FHS signature $\tau^{(i)}$ on some representative $(M_1^{(i)}, M_2^{(i)})$. The fact that all keys returned by $\mathcal{O}\texttt{Cor}$ are revoked and that none of the EPID signature satisfies the condition $\texttt{Identify}(\mathsf{sk}, (M_1^{(i)}, h_2^{(i)})) = 1$ for honest keys sk means that $(M_1^{(i)}, M_2^{(i)})$ is not in the equivalence class of (g, g^{s_j}) for all $i \in [1, c+1]$ and (g, g^{s_j}) generated during a $\mathcal{O}\texttt{Add}$ query. Moreover, as μ_i is produced while taking as input a revocation list $\texttt{SRL}$ containing $\mu_1, \ldots, \mu_{i-1}$, we know, thanks to the soundness of π, that $(M_1^{(i)}, M_2^{(i)})$ is not in the equivalence class of $(M_1^{(j)}, M_2^{(j)})$ $\forall i \neq j \in [1, c+1]$. Therefore, the $c+1$ signatures $\tau^{(i)}$ are valid on $c+1$ different equivalence classes. As the adversary only received c FHS signatures and as it did not use one associated with a honest platform (otherwise $\mathcal{A}$ would be a type 2 adversary), $\exists i^* \in [1, c+1]$ such that $\tau^{(i^*)}$ is valid on an equivalence class that was never submitted to the FHS signing oracle. $\mathcal{R}$ then makes a guess on i^*

and returns $\tau^{(i^*)}$ along with $(M_1^{(i^*)}, M_2^{(i^*)})$ to the challenger of the EUF-CMA security experiment. It then succeeds with probability at least $\frac{\epsilon}{c+1}$.

Anonymity. We here proceed through a sequence of games where Game $1, b$ is exactly the experiment $\mathtt{Exp}_{\mathcal{A}}^{an-b}$ defined in Section 3.2. For each i, we define $\mathtt{Adv}_i$ as the advantage of $\mathcal{A}$ playing Game $i, 0$ and Game $i, 1$. As Game $i, 0$ and Game $i, 1$ are virtually identical (the only difference is the parameter b), we will abuse notation and omit b in what follows. We set ϵ as the advantage of $\mathcal{A}$ playing Game 1 and define $\mathtt{Adv}_{\mathsf{CDH}}$ (resp. $\mathtt{Adv}_{\mathsf{DDH}}$) as the advantage against the CDH (resp. DDH) assumption.

Game 1. Here, the reduction generates normally all the platform secrets and is thus able to answer any oracle query by the adversary. By definition, $\mathtt{Adv}_1 = \epsilon$.

Game 2. In this second game, $\mathcal{R}$ randomly selects $k_0, k_1 \in [1, q_a]$ where q_a is a bound on the number of $\mathcal{O}\mathtt{Join}_{hon}$ queries. If the signatures μ_0 and μ_1 returned by the adversary in the experiment were not issued by the platforms k_0 and k_1, then $\mathcal{R}$ aborts. We then have $\mathtt{Adv}_2 = \frac{\epsilon}{q_a^2}$. The conditions we define in Figure 2 then ensure that no successful adversary can receive sk_{k_0} and sk_{k_1} through $\mathcal{O}\mathtt{Cor}^*$ oracle queries.

Game 3. In this third game, $\mathcal{R}$ proceeds as in Game 2 but parses each signature revocation list $\mathtt{SRL}$ used by $\mathcal{A}$ in $\mathcal{O}\mathtt{Sign}$ queries. If one of these lists contains a pair (h_0, h_2) that can be linked back to an honest secret key (that is, a key that has not leaked through a $\mathcal{O}\mathtt{Cor}^*$ query) and yet $\mathcal{R}$ never used this pair to answer a previous $\mathcal{O}\mathtt{Sign}$ query, then $\mathcal{R}$ aborts. We show below that $\mathtt{Adv}_3 \geq \mathtt{Adv}_2 - \mathtt{Adv}_{\mathsf{CDH}}$.

Game 4. In this fourth game, $\mathcal{R}$ proceeds as in Game 3 except that it now simulates the proof of knowledge π included in signatures it generates. We then have $\mathtt{Adv}_4 \geq \mathtt{Adv}_3 - \mathtt{Adv}_{ZK}$, where $\mathtt{Adv}_{ZK}$ is the advantage of an adversary against the zero-knowledge property of the proof system used to generate π.

Game (5,i). In this game, defined for $i \in [1, q_S]$ with q_S a bound on the number of $\mathcal{O}\mathtt{Sign}$ queries, $\mathcal{R}$ proceeds as in Game 4 but answers the i first $\mathcal{O}\mathtt{Sign}$ queries as follows. For $j \in [1, i]$, let $\mathtt{SRL}_j$ be the signature revocation list involved in the j-th query to $\mathcal{O}\mathtt{Sign}$ and $\mathcal{H}_j$ the set of honest keys that are not revoked by $\mathtt{SRL}_j$. Since Game 3, $\mathcal{R}$ is indeed perfectly able to identify the set $\mathcal{H}_j$ corresponding to $\mathtt{SRL}_j$. $\mathcal{R}$ selects a random key $\mathsf{sk} \in \mathcal{H}_j$ and proceeds as usual if $\mathsf{sk} \notin \{\mathsf{sk}_{k_0}, \mathsf{sk}_{k_1}\}$. Else, it replaces in the generated signature the elements (M_1, M_2) and h_2 by ones generated using a fresh random secret key and the FHS signature τ by one valid on (M_1, M_2). We show below that $\mathtt{Adv}_{(5,i)} \geq \mathtt{Adv}_{(5,i-1)} - \mathtt{Adv}_{\mathsf{DDH}}$ if we define $\mathtt{Adv}_{(5,0)} = \mathtt{Adv}_4$.

Game 6. In this sixth game, $\mathcal{R}$ replaces in μ^* the elements (M_1, M_2) and h_2 by ones generated using a fresh random secret key and the FHS signature τ by one valid on (M_1, M_2). We show below that $\mathtt{Adv}_6 \geq \mathtt{Adv}_5 - \mathtt{Adv}_{\mathsf{DDH}}$.

In the end we then get

$$\mathtt{Adv}_6 \geq \frac{\epsilon}{q_a^2} - \mathtt{Adv}_{\mathsf{CDH}} - \mathtt{Adv}_{ZK} - (q_S + 1)\mathtt{Adv}_{\mathsf{DDH}}.$$

As all the signatures in Game 6 (including μ^*) are generated with keys independent of sk_{k_0} and sk_{k_1}, the adversary can only succeed with negligible probability, which concludes our proof.

It then remains to prove the inequalities (1) $\mathtt{Adv}_3 \geq \mathtt{Adv}_2 - \mathtt{Adv}_{\mathsf{CDH}}$, (2) $\mathtt{Adv}_{(5,i)} \geq \mathtt{Adv}_{(5,i-1)} - \mathtt{Adv}_{\mathsf{DDH}}$ and (3) $\mathtt{Adv}_6 \geq \mathtt{Adv}_5 - \mathtt{Adv}_{\mathsf{DDH}}$. Regarding (1), we note that this is exactly what we prove in Lemma 4. We now focus on (2).

Let (g, g^x, g^y, g^z) be a DDH instance, we show that if $\mathcal{A}$ can distinguish Game $(5, i)$ from Game $(5, i-1)$, then it can be used to decide whether $z = x \cdot y$.

$\mathcal{R}$ implicitly sets the signing keys of k_0 and k_1 as $x \cdot u_0$ and $x \cdot u_1$, respectively, for random $u_0, u_1 \in \mathbb{Z}_p$, when it receives the corresponding $\mathcal{O}\mathtt{Join}_{hon}$ queries. Using g^x, it can indeed receive, for $d \in \{0, 1\}$, a FHS signature τ on $(g, g^{x \cdot u_d})$ that it stores for further uses.

Upon receiving the j-th $\mathcal{O}\mathtt{Sign}^*$ query, $\mathcal{R}$ proceeds as follows. If $j < i$, then it proceeds as defined in Game $(5, i-1)$. If $j > i$, it randomly selects a platform k that is not revoked by $\mathtt{SRL}$. If $k \notin \{k_0, k_1\}$, then it generates the signature as usual. Else, $k = k_d$ for some $d \in \{0, 1\}$ and $\mathcal{R}$ re-rerandomizes τ together with $(g, g^{x \cdot u_d})$ and thus gets a FHS signature τ' valid on a new representative (M_1, M_2). It then sets $h_1 = H'(M_1) = g^r$ for some random r and is then able to compute $h_2 = (g^{x \cdot u_d})^r$. As π is simulated since Game 4, it can then return a valid EPID signature μ.

If $j = i$, we distinguish two cases depending on the value of $\mathsf{sk} \in \mathcal{H}_j$, the key selected by $\mathcal{R}$ to answer this query. If $\mathsf{sk} \notin \{\mathsf{sk}_{k_0}, \mathsf{sk}_{k_1}\}$, then $\mathcal{R}$ generates normally the signature and we clearly have $\mathtt{Adv}_{(5,i)} = \mathtt{Adv}_{(5,i-1)}$. Else, let d be such that $\mathsf{sk} = \mathsf{sk}_{k_d}$. $\mathcal{R}$ selects a random $r \in \mathbb{Z}_p$, defines $M_1 = g^{y \cdot r}$ and $M_2 = g^{z \cdot r \cdot u_d}$, programs $h_1 = H'(M_1) = g^y$ (in the unlikely event where M_1 has already been queried, then $\mathcal{R}$ starts over with a new random value r) and then sets $h_2 = (g^z)^{u_d}$. Using its knowledge of isk, it can then generate a valid FHS signature on (M_1, M_2) and returns the resulting signature μ (the proof π is simulated since Game 4). In the case where $z = x \cdot y$, this is a valid signature issued by k_d and we are playing Game $(5, i-1)$. Else, this is exactly Game $(5, i)$. Any adversary able to distinguish these two games in this case can thus be used to solve the DDH problem. In all cases, we have $\mathtt{Adv}_{(5,i)} \geq \mathtt{Adv}_{(5,i-1)} - \mathtt{Adv}_{\mathsf{DDH}}$.

The proof regarding the point (3) is essentially the same, which proves anonymity of our construction.

Remark 7. A reader familiar with security proofs of group signature schemes might be surprised by our Games $5, i$ for $i \in [1, q_S]$. Indeed, with group signatures, the game 6 would be enough as it ensures that the elements constituting the challenge signature μ^* are independent of the bit b. This reasoning does not seem to apply for EPID schemes because even a fully random μ^* remains implicitly associated with sk_{k_b}, which has consequences on subsequent $\mathcal{O}\mathtt{Sign}^*$ queries. Indeed, we recall that $\mathcal{O}\mathtt{Sign}^*$ normally uses an honest key that is not implicitly revoked by $\mathtt{SRL}$ to generate a signature. An adversary querying the genuine $\mathcal{O}\mathtt{Sign}^*$ oracle with empty revocation lists $\mathtt{SRL}$ could then expect to receive from time to time signatures generated with either sk_{k_0} or sk_{k_1}. Conversely, among these two keys, only $\mathsf{sk}_{k_{\overline{b}}}$ (with $\overline{b} + b = 1$) has a chance to be used to answer $\mathcal{O}\mathtt{Sign}^*$ queries with a revocation list containing elements from the random μ^*. With standard $\mathcal{O}\mathtt{Sign}^*$ answers, it therefore seems unreasonable to claim that a random μ^* will necessarily lead to a negligible advantage for $\mathcal{A}$ as the distribution of elements returned by $\mathcal{R}$ still depends on $\mathsf{sk}_{k_{\overline{b}}}$ (and so on b). Fortunately, this problem can easily be solved by tweaking the behaviour of $\mathcal{O}\mathtt{Sign}^*$, as we do in Games $5, i$ for $i \in [1, q_S]$. Once all signatures involving k_0 or k_1 are generated with fresh random keys, we can indeed assure that our reduction in Game 6 is perfectly independent of the bit b that $\mathcal{A}$ must guess.

5 An Efficient Variant with Limited Concurrent Enrolments

5.1 Description

In the previous scheme, an EPID signature issued on an empty revocation list $\mathtt{SRL}$ contains 2 scalars, 5 elements of $\mathbb{G}_1$ and 1 of $\mathbb{G}_2$. We can do even better and construct EPID signatures with only 2 scalars and 4 elements of $\mathbb{G}_1$ (non-revocation proof excluded) using Pointcheval-Sanders (PS) signatures [27]. Using the BLS12 curve from [9] to provide a common metric, this means a reduction of 36 % of the bit size of the signature. Moreover, this avoids to implement the complex arithmetic of $\mathbb{G}_2$ (that is usually defined over a non-prime field) on the signer's side, which is particularly important when the signer is a constrained device. Apart from this, this new variant is very similar to the previous one and is mostly presented here for completeness. But first, we need to recall some elements on PS signatures.

PS Signature. In [27], Pointcheval and Sanders constructed re-randomizable signatures σ consisting of only 2 elements of $\mathbb{G}_1$, no matter the size n of the signed vector $(\mathsf{m}_1, \ldots, \mathsf{m}_n)$. Here, re-randomizability means that one can publicly derive a new signature σ' from σ by simply raising each element to the same random power. The point is that σ and σ' are unlinkable (under the DDH assumption in $\mathbb{G}_1$) for anyone that does not know the full signed vector.

We will here focus on the case where $n = 1$ as it is sufficient for our construction. The scheme described below is actually a slight variant of the original scheme that uses some folklore techniques (see *e.g.* [1, 2, 5]) to reduce the verification complexity at the cost of an additional element in the signature.

- $\mathtt{Setup}(1^\lambda)$: Outputs the parameters pp containing the description of type-3 bilinear groups $(\mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T, e)$ along with a set of generators $(g, \widetilde{g}) \in \mathbb{G}_1 \times \mathbb{G}_2$.

- $\texttt{Keygen}(pp)$: Generates two random scalars x and y and sets sk as (x, y) and pk as $(\widetilde{X} = \widetilde{g}^x, \widetilde{Y} = \widetilde{g}^y)$.
- $\texttt{Sign}(\mathsf{sk}, \mathsf{m})$: On message m, generates a signature $(\sigma_1, \sigma_2, \sigma_3) \leftarrow (g^r, g^{r(x+y\cdot\mathsf{m})}, g^{r\cdot\mathsf{m}})$ for some random scalar r.
- $\texttt{Verify}(\mathsf{pk}, m, (\sigma_1, \sigma_2, \sigma_3))$: Accepts the signature on m if $\sigma_3 = \sigma_1^{\mathsf{m}}$ and if the following equality holds: $e(\sigma_1, \widetilde{X}) \cdot e(\sigma_3, \widetilde{Y}) = e(\sigma_2, \widetilde{g})$.

With this variant, m is no longer involved in the pairing equation of $\texttt{Verify}$, which will dramatically reduce the cost of related zero-knowledge proofs. The price is the additional element σ_3, which seems reasonable in our context. If one needs instead to optimise the EPID size, then one can simply use the original PS signature and adapt the following construction. One can note that the EUF-CMA security of this variant is trivially implied by the one of the original scheme.

Construction.

- $\texttt{Setup}(1^k)$: this algorithm returns the public parameters pp containing the description of a bilinear group $(e, \mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T)$ along with two generators $g \in \mathbb{G}_1$ and $\widetilde{g} \in \mathbb{G}_2$ and two hash functions $H : \{0,1\} \to \mathbb{Z}_p$ and $H' : \{0,1\} \to \mathbb{G}_1$.
- $\texttt{GKeygen}(pp)$: this algorithm generates a key pair $(\mathsf{isk}, \mathsf{ipk})$ for the PS signature scheme by setting $\mathsf{isk} = (x, y) \overset{\$}{\leftarrow} \mathbb{Z}_p^2$ and $\mathsf{ipk} = (pp, \widetilde{X}, \widetilde{Y}) \leftarrow (\widetilde{g}^x, \widetilde{g}^y)$.
- $\texttt{Join}$: this protocol starts when a platform $\mathcal{P}$, taking as inputs ipk, contacts the issuer $\mathcal{I}$ for enrolment. It first generates a random $s \overset{\$}{\leftarrow} \mathbb{Z}_p$ and sends g^s to $\mathcal{I}$ who owns isk. $\mathcal{P}$ then engages in an interactive proof of knowledge of s with $\mathcal{I}$, using the Schnorr's protocol [29]. Once the latter is complete, $\mathcal{I}$ selects a random $r \overset{\$}{\leftarrow} \mathbb{Z}_p$ and computes a PS signature $\sigma = (\sigma_1, \sigma_2, \sigma_3) \leftarrow (g^r, g^{r\cdot x} \cdot (g^s)^{r\cdot y}, (g^s)^r)$ on s that it returns to $\mathcal{P}$. The platform then stores (s, σ) as its secret key sk.
- $\texttt{KeyRevoke}(\{\mathsf{sk}_i\}_{i=1}^m)$: this algorithm takes as input a set of m platform secret keys $\mathsf{sk}_i = (s^{(i)}, \sigma^{(i)})$ and returns a corresponding key revocation list $\texttt{KRL}$ with $\texttt{KRL}[i] = \mathsf{sk}_i$, for $i \in [1, m]$.
- $\texttt{SigRevoke}(\{(\mu_i)\}_{i=1}^n)$: this algorithm takes as input a set of n EPID signatures $\{(\mu_i)\}_{i=1}^n$ and parses each of them as $((\sigma_1^{(i)}, \sigma_2^{(i)}, \sigma_3^{(i)}), h_2^{(i)}, \pi^{(i)})$. It then returns a signature revocation list $\texttt{SRL}$ such that $\texttt{SRL}[i] = (\sigma_1^{(i)}, h_2^{(i)})$, for $i \in [1, n]$.
- $\texttt{Sign}(\mathsf{ipk}, \texttt{SRL}, \mathsf{sk}, \mathsf{m})$: To sign a message m while proving that it has not been implicitly revoked by $\texttt{SRL}$, a platform $\mathcal{P}$ owning $\mathsf{sk} = (s, (\sigma_1, \sigma_2, \sigma_3))$ generates a random $r \overset{\$}{\leftarrow} \mathbb{Z}_p^*$ and
  1. re-randomizes the PS signature $(\sigma_1', \sigma_2', \sigma_3') \leftarrow (\sigma_1^r, \sigma_2^r, \sigma_3^r)$;
  2. computes $(h_1, h_2) \leftarrow (H'(\sigma_1'), h_1^s)$;
  3. for all $i \in [1, n]$, it parses $\texttt{SRL}[i]$ as $(\sigma_1^{(i)}, h_2^{(i)})$ and computes $h_1^{(i)} \leftarrow H'(\sigma_1^{(i)})$;
  4. it generates a proof π of knowledge of s such that $\sigma_3 = \sigma_1^s$ and $h_2 = h_1^s$ and that $(h_1^{(i)})^s \neq h_2^{(i)}$ for all $i \in [1, n]$ using the protocol from [15]. More specifically, it selects random scalars r_i and computes $C_i =$

$((h_1^{(i)})^s/h_2^{(i)})^{r_i}$. If $\exists i \in [1,n]$ such that $C_i = 1$, then it returns $\perp$. Else, it selects $k, \{k_{i,1}, k_{i,2}\}_{i=1}^n \xleftarrow{\$} \mathbb{Z}_p^{2n+1}$ and computes $(K_{0,1}, K_{0,2}) \leftarrow (\sigma_1^k, h_1^k)$ along with $(K_{i,1}, K_{i,2}) \leftarrow ((h_1^{(i)})^{k_{i,1}} \cdot (1/(h_2^{(i)}))^{k_{i,2}}, h_1^{k_{i,1}} \cdot (1/h_2)^{k_{i,2}})$. It then computes

$$c = H(\sigma'_1, \sigma'_2, \sigma'_3, h_1, h_2, \{C_i\}_{i=1}^n, \{K_{i,1}, K_{i,2}\}_{i=0}^n, \mathsf{m}).$$

along with $z = k + c \cdot s$ and $(z_{i,1}, z_{i,2}) = (k_{i,1} + c \cdot s \cdot r_i, k_{i,2} + c \cdot r_i)$. The proof π is then set as $(\{C_i\}_{i=1}^n, c, z, \{z_{i,1}, z_{i,2}\}_{i=1}^n)$;

5. it returns the signature $\mu = ((\sigma'_1, \sigma'_2, \sigma'_3), h_2, \pi)$.

- `Identify`(sk, t): this algorithm parses sk as $(s, (\sigma_1, \sigma_2, \sigma_3)))$ and t as (σ_1, h_2), and returns 1 if $h_2 = H'(\sigma_1)^s$ and 0 otherwise.
- `Verify`$(\mathsf{ipk}, \mathtt{SRL}, \mathtt{KRL}, \mu, \mathsf{m})$: to verify an EPID signature μ, the verifier parses it as $((\sigma'_1, \sigma'_2, \sigma'_3), h_2, \pi)$, each $\mathtt{KRL}[i]$ as $(s^{(i)}, (\sigma_1^{(i)}, \sigma_2^{(i)}, \sigma_3^{(i)}))$ for $i \in [1,m]$ and each $\mathtt{SRL}[i]$ as $(\sigma_1^{(i)}, h_2^{(i)})$ for $i \in [1,n]$. It then returns 1 if all the following conditions hold and 0 otherwise.
  1. $\sigma_1 \neq 1_{\mathbb{G}_1} \wedge e(\sigma_1, \widetilde{X}) \cdot e(\sigma_3, \widetilde{Y}) = e(\sigma_2, \widetilde{g})$;
  2. $\forall i \in [1,m]$, `Identify`$(\mathtt{KRL}[i], (\sigma_1, h_2)) = 0$;
  3. $\forall i \in [1,n]$, $C_i \neq 1$;
  4. $c = H(\sigma_1, \sigma_2, \sigma_3, h_1, h_2, \{C_i\}_{i=1}^n, \{K_{i,1}, K_{i,2}\}_{i=0}^n, \mathsf{m})$, where $h_1 \leftarrow H'(\sigma_1)$, $(K_{0,1}, K_{0,2}) \leftarrow (\sigma_1^z \cdot \sigma_3^{-c}, h_1^z \cdot h_2^{-c})$ and $(K_{i,1}, K_{i,2}) \leftarrow ([(h_1^{(i)})^{z_{i,1}}/(h_2^{(i)})^{z_{i,2}}] \cdot C_i^{-c}, h_1^{z_{i,1}}/(h_2^{z_{i,2}}))$ with $h_1^{(i)} = H'(\sigma_1^{(i)})$.

The correctness of this variant essentially follows from the one of the previous scheme, the main difference being located in Step 1 of the `Verify` algorithm where the verification of a FHS signature is here replaced by a verification of a PS signature.

Remark 8. Several constructions from the state-of-the-art (*e.g.* [2,12]) impose a collaborative generation of the platform secret during the `Join` algorithm. That is, once the platform has sent a commitment to some secret value s, the issuer selects some scalar s' and implicitly defines the platform secret as $s + s'$ by issuing a certificate on this sum. This approach allows to generate certificates using signature schemes satisfying a weaker security notion (namely, weak chosen message security [6]) than the standard EUF-CMA. In our case, this would have no benefit on the efficiency of our signatures but only on the computational assumptions underlying PS signatures as their weak chosen message security is proven under a q-type assumption whereas their EUF-CMA security is proven under an interactive assumption (see [27,28]). We therefore choose to keep this simple, non-collaborative `Join` protocol and refer to [2] for details on the collaborative variant.

5.2 Security Proofs

The proofs of this variant are a mere adaptation of the ones from Section 4.2 so we will mainly focus on the slight differences between them.

Unforgeability. The unforgeability proof of this variant only differs from the one in Section 4.2 in the case of a type 3 forgery. Indeed, upon receiving a $\mathcal{O}$`Join` query, $\mathcal{R}$ now extracts the adversary's secret s from the interactive zero-knowledge proof and then queries its signing oracle to get a valid certificate $(\sigma_1, \sigma_2, \sigma_3)$. $\mathcal{R}$ then proceeds as for the previous construction and eventually receives $c+1$ EPID signatures, each containing a PS signature $\sigma^{(i)}$. As in Section 4.2, we therefore know that one of them is valid on some value $s^{(i^*)}$ that has never been queried to the signing oracle. $\mathcal{R}$ makes then a guess on the corresponding value i^* and then extracts $s^{(i^*)}$ from the proof of knowledge, which constitutes, along with $\sigma^{(i^*)}$, a valid forgery against PS signatures.

Anonymity. Here, the first difference occurs in Game (5,i) where $\mathcal{R}$ now answers the i first queries by generating a PS signature $(\sigma_1, \sigma_2, \sigma_3)$ on a random scalar t and by returning $h_2 = H'(\sigma_1)^t$. $\mathcal{R}$ proceeds similarly in Game 6. Here again, we prove indistinguishability between two games under DDH assumption. More precisely, if $(g, g^\alpha, g^\beta, g^\gamma)$ is a DDH instance, then $\mathcal{R}$ implicitly uses α as some platform secret key. To answer the i-th $\mathcal{O}$`Sign` query, $\mathcal{R}$ generates two random scalars r_1 and r_2 and constructs $\sigma_1 = (g^\beta)^{r_1}$, $\sigma_2 = (g^\beta)^{r_1 \cdot x}(g^\gamma)^{r_1 \cdot y}$ and $\sigma_3 = (g^\gamma)^{r_1}$. Moreover, it programs the random oracle to return $H'(\sigma_1) = g^{\beta \cdot r_2}$ and thus computes $h_2 = (g^\gamma)^{r_2}$. In the case where $\gamma = \alpha \cdot \beta$, this is a valid EPID signature for this platform and we are thus playing Game (5,$i-1$). Else, we play Game (5,i). Any adversary able to distinguish these two cases can then be used to solve the DDH problem.

Conclusion

In this paper, we have introduced a new security model for EPID, a cryptographic primitive embedded in billions of chips [23], which has important consequences in practice. Firstly, our new unforgeability property addresses the problems of previous models and in particular removes the need to extract all platforms' secret keys. This makes enrolment of new platforms simpler while allowing concurrent `Join`. Secondly, our new anonymity property allows decentralized management of revocation lists, which better captures the spirit of EPID. We have in particular showed that we can retain a strong anonymity notion even in presence of powerful adversaries with unlimited control of the revocation lists. All this leads to a better understanding of what an EPID system can truly ensure in what we believe to be the most realistic usage scenario.

Another result of our paper is that such strong properties can actually be achieved by very efficient constructions that we describe. Perhaps the most surprising feature of the latter is that they do not require to perform any significant test on the malicious revocation lists. This is particularly important as it proves that we are not simply shifting the burden of the revocation manager to each platform. The latter can indeed issue signatures with essentially the same complexity as in existing systems that require a trusted revocation manager.

Acknowledgements

The authors are grateful for the support of the ANR through project ANR-18-CE-39-0019-02 MobiS5.

References

1. Giuseppe Ateniese, Jan Camenisch, Susan Hohenberger, and Breno de Medeiros. Practical group signatures without random oracles. *IACR Cryptol. ePrint Arch.*, 2005.
2. Amira Barki, Nicolas Desmoulins, Saïd Gharout, and Jacques Traoré. Anonymous attestations made practical. ACM WISEC, 2017.
3. Mihir Bellare, Daniele Micciancio, and Bogdan Warinschi. Foundations of group signatures: Formal definitions, simplified requirements, and a construction based on general assumptions. In Eli Biham, editor, *EUROCRYPT 2003*, volume 2656 of *LNCS*, pages 614–629. Springer, Heidelberg, May 2003.
4. Mihir Bellare, Haixia Shi, and Chong Zhang. Foundations of group signatures: The case of dynamic groups. In Alfred Menezes, editor, *CT-RSA 2005*, volume 3376 of *LNCS*, pages 136–153. Springer, Heidelberg, February 2005.
5. David Bernhard, Georg Fuchsbauer, Essam Ghadafi, Nigel P. Smart, and Bogdan Warinschi. Anonymous attestation with user-controlled linkability. *Int. J. Inf. Sec.*, 2013.
6. Dan Boneh and Xavier Boyen. Short signatures without random oracles and the SDH assumption in bilinear groups. *Journal of Cryptology*, 21(2):149–177, April 2008.
7. Dan Boneh, Ben Lynn, and Hovav Shacham. Short signatures from the Weil pairing. In Colin Boyd, editor, *ASIACRYPT 2001*, volume 2248 of *LNCS*, pages 514–532. Springer, Heidelberg, December 2001.
8. Dan Boneh and Hovav Shacham. Group signatures with verifier-local revocation. In Vijayalakshmi Atluri, Birgit Pfitzmann, and Patrick McDaniel, editors, *ACM CCS 2004*, pages 168–177. ACM Press, October 2004.
9. Sean Bowe. BLS12-381: New zk-SNARK Elliptic Curve Construction. https://electriccoin.co/blog/new-snark-curve/, 2017.
10. Ernest F. Brickell, Jan Camenisch, and Liqun Chen. Direct anonymous attestation. In Vijayalakshmi Atluri, Birgit Pfitzmann, and Patrick McDaniel, editors, *ACM CCS 2004*, pages 132–145. ACM Press, October 2004.
11. Ernie Brickell and Jiangtao Li. Enhanced privacy id: a direct anonymous attestation scheme with enhanced revocation capabilities. In *WPES 2007*, 2007.
12. Ernie Brickell and Jiangtao Li. Enhanced privacy ID from bilinear pairing for hardware authentication and attestation. In Ahmed K. Elmagarmid and Divyakant Agrawal, editors, *IEEE Conference on Social Computing, SocialCom*, 2010.
13. Jan Camenisch, Manu Drijvers, and Anja Lehmann. Universally composable direct anonymous attestation. In Chen-Mou Cheng, Kai-Min Chung, Giuseppe Persiano, and Bo-Yin Yang, editors, *PKC 2016, Part II*, volume 9615 of *LNCS*, pages 234–264. Springer, Heidelberg, March 2016.
14. Jan Camenisch and Anna Lysyanskaya. Signature schemes and anonymous credentials from bilinear maps. In Matthew Franklin, editor, *CRYPTO 2004*, volume 3152 of *LNCS*, pages 56–72. Springer, Heidelberg, August 2004.

15. Jan Camenisch and Victor Shoup. Practical verifiable encryption and decryption of discrete logarithms. In Dan Boneh, editor, *CRYPTO 2003*, volume 2729 of *LNCS*, pages 126–144. Springer, Heidelberg, August 2003.
16. David Chaum and Eugène van Heyst. Group signatures. In Donald W. Davies, editor, *EUROCRYPT'91*, volume 547 of *LNCS*, pages 257–265. Springer, Heidelberg, April 1991.
17. Liqun Chen, Paul Morrissey, and Nigel P. Smart. On proofs of security for DAA schemes. In *ProvSec 2008*, 2008.
18. Liqun Chen, Paul Morrissey, and Nigel P. Smart. Pairings in trusted computing. In *Pairing 2008*, 2008.
19. Liqun Chen, Paul Morrissey, and Nigel P. Smart. Daa: Fixing the pairing based protocols. *IACR Cryptol. ePrint Arch.*, page 198, 2009.
20. Marc Fischlin. Communication-efficient non-interactive proofs of knowledge with online extractors. In Victor Shoup, editor, *CRYPTO 2005*, volume 3621 of *LNCS*, pages 152–168. Springer, Heidelberg, August 2005.
21. Georg Fuchsbauer, Christian Hanser, and Daniel Slamanig. Structure-preserving signatures on equivalence classes and constant-size anonymous credentials. *Journal of Cryptology*, 32(2):498–546, April 2019.
22. Steven D. Galbraith, Kenneth G. Paterson, and Nigel P. Smart. Pairings for cryptographers. *Discret. Appl. Math.*, 2008.
23. Intel. A cost-effective foundation for end-to-end iot security, white paper. https://www.intel.in/content/www/in/en/internet-of-things/white-papers/iot-identity-intel-epid-iot-security-white-paper.html, 2016.
24. ISO/IEC. ISO/IEC 20008-1:2013 information technology — security techniques — anonymous digital signatures — part 1: General. https://www.iso.org/standard/57018.html, 2013.
25. ISO/IEC. ISO/IEC 20008-2:2013 information technology — security techniques — anonymous digital signatures — part 2: Mechanisms using a group public key. https://www.iso.org/standard/56916.html, 2013.
26. Nada El Kassem, Luís Fiolhais, Paulo Martins, Liqun Chen, and Leonel Sousa. A lattice-based enhanced privacy ID. In *WISTP 2019*, 2019.
27. David Pointcheval and Olivier Sanders. Short randomizable signatures. In Kazue Sako, editor, *CT-RSA 2016*, volume 9610 of *LNCS*, pages 111–126. Springer, Heidelberg, February / March 2016.
28. David Pointcheval and Olivier Sanders. Reassessing security of randomizable signatures. In Nigel P. Smart, editor, *CT-RSA 2018*, volume 10808 of *LNCS*, pages 319–338. Springer, Heidelberg, April 2018.
29. Claus-Peter Schnorr. Efficient identification and signatures for smart cards. In Gilles Brassard, editor, *CRYPTO'89*, volume 435 of *LNCS*, pages 239–252. Springer, Heidelberg, August 1990.
30. TCG. https://trustedcomputinggroup.org/authentication/, 2015.