Protecting Privacy by Sanitizing Personal Data: a New Approach to Anonymous Credentials

Sébastien Canard
Orange Labs, Caen, France
Applied Crypto Group
sebastien.canard@orange.com

Roch Lescuyer[*]
Normandie Université, UNICAEN,
GREYC, ENSICAEN, CNRS, Caen, France
roch.lescuyer@ensicaen.fr

ABSTRACT

Anonymous credential systems allow users to obtain certified credentials from organizations and use them later without being traced. For instance, a student will be able to prove, using his student card certified by the University, that he is a student living *e.g.* in Hangzhou without revealing other information given by the student card, such as his name or studies. Besides, sanitizable signatures enable a designated person, called the sanitizer, to modify some parts of a signed message in a controlled way, such that the message can still be verified *w.r.t.* the original signer.

We propose in this paper to formalize the following new idea. A user gets from the organization a signed document certifying personal data (*e.g.* name, address, studies, etc.) and plays the role of the sanitizer. When showing his credential, he uses sanitization techniques to hide the information he does not want to reveal (*e.g.* name, studies or complete address), and shows the resulting document, which is still seen as a document certified by the organization. Unfortunately, existing sanitizable signatures can not directly be used for this purpose. We thus seek for generic conditions on them to be used as anonymous credentials. We also provide a concrete construction based on standard assumptions and secure in the random oracle model.

Categories and Subject Descriptors

E.3 [**Data Encryption**]: Public Key Cryptosystems; K.4.1 [**Computers and Society**]: Public Policy Issues—*Privacy*; K.6.5 [**Management of Computing and Information Systems**]: Security and Protection—*Authentication*

General Terms

Algorithms, Design, Security.

Keywords

Cryptographic protocols, Anonymous credentials, Privacy-enhancing systems, Sanitizable signatures

ASIA CCS'13, May 8–10, 2013, Hangzhou, China.
Copyright 2013 ACM 978-1-4503-1767-2/13/05 ...$15.00.

1. INTRODUCTION

ANONYMOUS CREDENTIALS. Digital signatures are a widely used cryptographic tool in our everyday life. We here focus on the use of digital signatures in credential systems where a user gets some certified credentials issued by some organization. These certified credentials enable the user to prove access rights, get some advantages, or anything else which requires authentication. For instance a student is given an electronic student card by the University and later uses this credential to prove that he actually is a student, if necessary. Anonymous credential systems [16] allow users to prove possession of credentials in an anonymous way. When a credential is shown, a proof of possession is done without leaking (even for the issuing organization) any knowledge about neither the owner of the credential, nor the credential itself.

Several ways exist to design anonymous credential systems and some industrial companies seem to be interested. As an evidence, IBM's Idemix system [24] is based on the use of group signatures (or some close variants), as well as several existing papers [9, 10, 2, 8, 20, 15]. Microsoft prefers the Brands' blind signature approach [4] for their UProve technology [25], as in [23]. Finally, the Orange operator seems to be interested in the use of aggregate signatures [14].

SANITIZABLE SIGNATURES. Sanitizable signatures [1] enable a designated entity, called the *sanitizer*, to modify some parts of a signed message in a controlled way defined by the initial signer at the creation of the signature.

In fact, there are multiple ways to handle such a problem and a lot of works on this subject can be found in the cryptographic literature. In a nutshell, (i) the sanitization procedure may be public [26, 5], or restricted to a designated sanitizer [1, 13, 6, 7, 11], (ii) the signed document can be a bit string [1, 13, 6, 11] or have a particular data structure [5], (iii) the sanitizer may be allowed to modify some parts of the message, or simply remove them [26, 5], and (iv) the capacity of sanitization can be given to a single sanitizer [1, 13, 6, 11], or to a group of sanitizers [7, 12].

Let us consider the case where only a designated sanitizer is able to modify a given message. The signer divides a message $m \in \{0,1\}^*$ into N blocks $m_1, \dots, m_N$, defines the set $\textsc{adm} \subseteq [1, N]$ of blocks that are said *admissible* (*i.e.* allowed to be subsequently modified) and then signs the whole message, using some key related to the sanitizer. Using this key, the sanitizer is next able to modify the admissible parts of the given message so that the resulting signature is still valid under the signer's public key.

[*] This work was done while the second author was at Orange Labs, Caen, France and ENS, Paris, France.

From sanitizable signatures to anonymous credentials. Our idea in this paper is to use sanitizable signatures to design anonymous credentials, which is, to the best of our knowledge, a new approach. The underlying idea is very simple: the organization in the anonymous credential system is a signer and the user who obtain certified credentials plays the role of a sanitizer.

For example, a student (acting as a sanitizer) obtains a signature (the certified credential) by the University (which plays the role of the signer) on the message: $m :=$ 'Name: Wang Li; City: Hangzhou; Studies: Computer Science'. This message is split by the signer into six sub-messages $m_1 :=$ 'Name:', $m_2 :=$ 'Wang Li', $m_3 :=$ '; City:', $m_4 :=$ 'Hangzhou', $m_5 :=$ '; Studies:', and $m_6 :=$ 'Computer Science', who also sets $\textsc{adm} = \{2, 4, 6\}$. Finally, the whole message m is signed to obtain σ, using some key related to the student / sanitizer, and sends the whole result (m, σ) to the student / sanitizer. The latter can next prove *e.g.* that he is a student living in Hangzhou without revealing his name, nor his studies. For this purpose, he can use the appropriate key to replace m_2 and m_6 by some predefined non relevant symbol (like "#") of adequate length, to obtain the message $m :=$ 'Name: ######; City: Hangzhou; Studies: ######' and modify the University's signature accordingly. This student is hence anonymous inside the group of students living in Hangzhou but proves such a belonging since the University's signature is valid. This simple approach seems to work but unfortunately, "the devil is on the details" and, as we will see, additional work is necessary.

Additional needs for sanitizable signatures. An anonymous credential system should be anonymous and unforgeable (see *e.g.* [14, 9]). Regarding sanitizable signature schemes, different formal security models can be found in the literature. An initial one has been designed by Brzuska *et al.* [6] and gives main procedures (for signing, sanitizing, verifying, proving accountability of a signature and judging these proofs) and basic security properties (immutability, accountability and transparency). Based on this initial model, Brzuska *et al.* [7] have next added the unlinkability property, Canard and Jambert [11] have integrated some extensions and Canard *et al.* [12] have formalized the concept for multiple signers and sanitizers. These models do not entirely match our needs for the design of a secure anonymous credential system. More precisely, the following questions need to be addressed.

Verification without sanitizing key. The verification of a sanitizable signature generally needs *the sanitizer's public key.* In our case, since we want the user / sanitizer to be anonymous, a signature should be checkable without the sanitizer's public key. We thus modify the verification algorithm accordingly.

Traceability of signatures. The above feature obviously gives the anonymity of the sanitizer. This implies, for security purpose, the necessity for an algorithm to recover the actual designated sanitizer of a given message-signature pair. However, in order to prevent organizations to trace users, this algorithm should be carried out by a separate authority. This algorithm has no equivalent in the literature of sanitizable signatures. We will call this procedure FindSan, since it is the counterpart of the FindOri algorithm in the model of [12]. The reader will notice that the above tracing procedure should not be confused with the opening procedure in anonymous primitives, like group signatures. In a group signature scheme, the opening aims at finding the actual producer of a given signature, whereas we just want here to recover the actual *designated* sanitizer of a given message/signature pair. The latter could have not modified or even seen the signature.

Dealing with the proof algorithm. The above tracing procedure is not designed to decide whether a given signature has been sanitized or not, which is the role of the Prove algorithm in [6] and the AlgOpen procedure in [12]. Such a procedure is no more useful in our case. In front of a student proving that he is a student, the verifier sees that the document has been sanitized, and some fields hidden, but this is not a matter of concern here. In fact, we will see that we do not need the accountability properties of sanitizable signatures, at least as long as there is no opening algorithm in the anonymous credential system. Such a procedure would imply a notion of non-frameability, and, consequently, an accountability property for the signer.

Restrictions on admissible values. We want the true sanitizer to be able to hide admissible parts of the message. However, we do not want him to modify as he wants these admissible parts so as to be able to prove false statements about his attributes (*e.g.* proving to be a student in Hangzhou while being a student in Shanghai). We can use for this purpose the `LimitSet` extension of sanitizable signatures given by Canard and Jambert [11], which limits the set of possible modifications on a single block of the message.

Replaying sessions. Another problem we have to deal with is that an eavesdropper could intercept sanitized documents and replay them as often as she wants. To fix this problem, we let the verifier send at each authentication a random value and the sanitizer modify a specific admissible part of the message with this value. The latter may depend on the session data, like the verifier's identity and the current time.

Comparison with related works. The solution we propose in this paper is a new way to design an anonymous credential system. As said before, previous designs make use of group [24], blind [25] or aggregate signature schemes [14]. We here propose to use sanitizable signatures. The efficiency and security of the practical scheme we propose in this paper are close to the one of related works, as we will see at the end of the paper (see Section 5). Our main aim is clearly to propose a new approach. In fact, we provide two main contributions in this paper. The first one is a generic analysis of the properties for sanitizable signatures in order to use them in a privacy-preserving context. The second one is a concrete construction to show that those ideas can be efficiently implemented. Our proposal also introduces a new use case to sanitizable signatures, which has not yet been proposed in the related literature. Furthermore, we may expect that any improvement in sanitizable signatures will improve the related anonymous credential system.

One of the most significant interest of our solution *w.r.t.* related work is that there is no need, in our solution, for an interaction between the user and the issuer during the issuing protocol, contrary to other existing systems. The issuer produces a credential only with the sanitizer's public key. In both group and blind signature based anonymous credential systems, it is necessary for the user to add a random part of the secret key which will be certified by the organization with the attributes, and which is new for each credential. This is mainly due to the unforgeability of the used signatures. In our case, this is not necessary and only the user public key is useful. Moreover, in pseudonym systems, an interactive registration is necessary in order to obtain a non-interactive anonymous credential issuing [2]. This might be preferable in practice when the issuer needs to issue credentials without being on-line with the sanitizer. The price to cost is that there is no anonymity during the registration process. Nevertheless, if one needs some anonymity during the registration process, one can add this property at the cost of an interaction.

Another advantage of our sanitizable signature based solution is that the size of a credential is lower than what exists today [24, 25, 14].

Independently to our work, [17] addresses the problematic of anonymous authentication in cloud storage, which implies some kind of "attributes hiding". However, their techniques for attributes hiding are based on group signatures techniques rather than sanitization ones (in a strict sense). More precisely, the hiding of an attribute in [17] consists in not revealing the attribute by only proving its knowledge, using for this purpose Groth-Sahai based non-interactive zero-knowledge proofs. In our case, the attribute hiding is done by using the technique of sanitizable signatures, and thus *e.g.* chameleon hash functions [1], accumulators [11] or the split into two related signatures [7]. In our construction, we use a mix of the last two ideas, and thus not a zero-knowledge proof of a hidden attribute.

One may argue that our proposal does not permit to prove some well chosen statements on non-revealed attributes, such as proving that one has to prove that he has more than 65 years, which is the case for some existing proposals. In fact, in some particular cases, such as the one in the last sentence, we can use the power of the `LimitSet` extension of sanitizable signatures by permitting the user to replace his date of birth by *e.g.* the sub message "more that 65".

Organization of the paper. First of all, anonymous credentials are formalized in Section 2. Then Section 3 formally defines sanitizable signatures according to our needs. Section 4 shows how to build anonymous credentials from this new type of sanitizable signatures. Finally, a concrete sanitizable signature scheme is presented in Section 5.

2. ANONYMOUS CREDENTIALS

2.1 Notations

All along the paper, we distinguish two types of security properties. First of all, for any adversary $\mathcal{A}$ against a property prop and any security parameter $\lambda \in \mathbb{N}$, the success probability of $\mathcal{A}$ is the probability that the related experiment outputs 1. We say that the whole scheme verifies prop if this success is negligible (as a function of λ) for any polynomial-time $\mathcal{A}$. Then, for a decisional experiment, a challenge bit $b \in \{0,1\}$ is set and for any adversary $\mathcal{A}$ against a property prop and any security parameter λ, the advantage of $\mathcal{A}$ is $\Pr\left[\mathbf{Exp}_{\mathcal{A}}^{\mathsf{prop\text{-}1}}(\lambda) = 1\right] - \Pr\left[\mathbf{Exp}_{\mathcal{A}}^{\mathsf{prop\text{-}0}}(\lambda) = 1\right]$. We next say that the whole scheme verifies prop if this advantage is negligible for any polynomial-time $\mathcal{A}$.

2.2 Definition

We base the security model below on the work done in [9, 14], for the case of one single organization (see [14] for multiple organizations). An *anonymous credential system* $\mathcal{AC}$ involves users, an organization[1] and verifiers, and is given by the following algorithms.

Setup. This algorithm takes as input a security parameter λ and outputs global parameters gpk, which are given to all algorithms as auxiliary input, and the key pair $(\mathsf{opk}, \mathsf{osk})$ for the organization.

UserKG. This algorithm, on input $j \in \mathbb{N}$, produces a key pair $(\mathsf{upk}[j], \mathsf{usk}[j])$ for the user j.

Obtain ↔ Issue. The Obtain algorithm takes a user secret key $\mathsf{usk}[j]$ and the organization public key opk; the Issue algorithm takes the organization secret key osk, a user public key $\mathsf{upk}[j]$ and a list of messages / attributes $\{m_n\}_{n=1}^{N}$. At the end of the protocol, the user U obtained a credential C.

Show ↔ Verify. The Show algorithm takes a user secret key $\mathsf{usk}[j]$, the organization public key opk, a list of messages $\{m_n\}_{n=1}^{N'}$ and a credential C; the Verify public algorithm takes the organization public key opk and a list of messages $\{m_n\}_{n=1}^{N'}$. At the end of the protocol, the Verify algorithm outputs a bit $b \in \{0,1\}$.

2.3 Security properties

We now focus on the security properties related to anonymous credential systems. Experiments are played between a challenger and an adversary $\mathcal{A}$, which may in addition call the following oracles. The tables cred and reg are global variables which maintain the issued credentials and the certified attributes.

AddU(j). It adds j to the set HU of honest users, executes UserKG and updates the j-th entry of $\mathsf{upk}[j]$ and $\mathsf{usk}[j]$. It finally returns $\mathsf{upk}[j]$.

CrptU(j, pk_U). It adds j to the set CU of corrupted users and sets $\mathsf{upk}[j] \leftarrow \mathsf{pk}_U$.

USK(j). It returns $(\mathsf{usk}[j], \mathsf{cred}[j])$ and put j in the set KU of users for whom the secrets are known[2].

SndToU$_{algo}(j, m)$ (resp. SndToO(osk, m)) , with $algo \in$ {Issue, Verify}. It plays the role of the user j (resp. the organization) which receives the message m from the corrupted organization – the Issue case – or a corrupted verifier – the Verify case – (resp. the user j). This may modify the entry $\mathsf{cred}[j]$ (resp. $\mathsf{reg}[j]$).

GetCred$(\mathsf{osk}, j, \{m_n\}_{n=1}^{N})$. It permits the honest user j to obtain a credential on $\{m_n\}_{n=1}^{N}$ from the organization. The oracle plays the role of both the user and the organization. The credentials are added to $\mathsf{cred}[j]$, the attributes to $\mathsf{reg}[j]$ and the view is returned.

[1] For sake of the exposition, we have only one signer in our model, even if the case of multiple signers can easily be studied, using [6, 11, 7].

[2] The reason for the distinction between corruption and secret key leakage concerns the anonymity property. A user must remain anonymous even if his secret keys leak.

$\textsc{LoR}(b, j_0, j_1, \{m'_{n'}\}_{n'=1}^{N'})$. It takes as input a pair (j_0, j_1) of honest users and a set $\{m'_{n'}\}_{n'=1}^{N'}$ of N' attributes certified to both j_0 and j_1. It plays the role of the user j_b with the set $\{m'_{n'}\}_{n'=1}^{N'}$ with the adversary during a verification protocol.

Correctness. First of all, such a system has to be *correct*. This means that honestly obtained credentials must be accepted by an honest verifier.

Unforgeability. Regarding the unforgeability property, the aim of the adversary is to prove that she is in possession of some credentials issued by the organization while this is not the case. The adversary may (i) interact with the organization as corrupted user with the help of the $\textsc{SndToO}$ oracle, (ii) ask for an honest user to obtain credentials on her choice with the help of the $\textsc{GetCred}$ oracle and (iii) interact with an honest user as a corrupted verifier with the help of the $\textsc{SndToU}_{\textsc{Verify}}$ oracle. A forgery may appear by different ways. For instance, a new credential has been produced. Another possibility is that several corrupted users could have colluded and shared their credentials.

In order to properly address this property, we need to introduce the notion of *identity extractor*, which formalizes the idea that there exists a well-defined identity underlying an accepted authentication protocol, even if the user is anonymous. We require that there exists a pair of algorithms $\{\textsc{ESetup}, \textsc{Extract}\}$ such that (i) parameters returned by $\textsc{ESetup}$ are indistinguishable from those returned by $\textsc{Setup}$ and (ii) the $\textsc{Extract}$ algorithm correctly identify the underlying identity. More formally, we require that:

• For all adversary $\mathcal{A}$ there exists a negligible function $\nu(\cdot)$ s.t. $\Pr\big[\mathsf{gpk}_0 \leftarrow \textsc{Setup}(1^\lambda);\ \mathsf{gpk}_1 \leftarrow \textsc{ESetup}(1^\lambda);\ b \overset{\$}{\leftarrow} \{0,1\};\ b' \leftarrow \mathcal{A}(\mathsf{gpk}_b) : b' = b\big] < \frac{1}{2} + \nu(\lambda)$

• For all $(\mathsf{gpk}, \mathsf{ek}) \leftarrow \textsc{ESetup}(1^\lambda)$, all j, all $(\mathsf{pk}_{U_j}, \mathsf{sk}_{U_j})$, all $\{m_{n'}\}_{n'=1}^{N'}$ and all C, if $\textsc{Show}(\mathsf{osk}, \mathsf{sk}_{U_j}, \{m_{n'}\}_{n'=1}^{N'}, C \leftrightarrow \textsc{Verify}(\mathsf{osk}, \{m_{n'}\}_{n'=1}^{N'})$ outputs 1, then $\textsc{Extract}(\mathsf{ek}, \mathit{view}) = j$, where *view* is the view of the protocol.

With this notion of extractor, the unforgeability experiment $\textsc{Unforgeability}(\lambda)$ is given as follows.

– $(\mathsf{gpk}, (\mathsf{opk}, \mathsf{osk}), \mathsf{ek}) \leftarrow \textsc{ESetup}(1^\lambda)$

– $\mathcal{O} \leftarrow \{\textsc{AddU}(\cdot),\textsc{USK}(\cdot),\textsc{CrptU}(\cdot,\cdot), \textsc{SndToO}(\mathsf{osk},\cdot), \textsc{SndToU}_{\textsc{Verify}}(\cdot,\cdot), \textsc{GetCred}(\mathsf{osk},\cdot,\cdot)\}$

– $(\{m'_n\}_{n'=1}^{N'}, \mathit{state}) \leftarrow \mathcal{A}^{\mathcal{O}}(\mathsf{gpk}, \mathsf{opk})$

– $\mathcal{A}(\mathit{state})$ interacts with $\textsc{Verify}(\mathsf{opk}, \{m'_n\}_{n'=1}^{N'})$. The view of the protocol is denoted *view* and the response of the $\textsc{Verify}$ algorithm is denoted d

– $j \leftarrow \textsc{Extract}(\mathsf{ek}, \mathit{view})$

– Return 1 if $d = 1$ and $\exists \mathsf{n}' \in [1, N']$ s.t. $\forall(\{m_n\}_{n=1}^{N}) \in \mathsf{reg}[j]$, $m_{\mathsf{n}'} \notin \{m_n\}_{n=1}^{N}$.

A scheme $\mathcal{AC}$ is $(t, \ell_{\max}, q_S, \epsilon)$-unforgeable if the probability for an adversary $\mathcal{A}$ to win the $\textsc{Unforgeability}(\lambda)$ game in time t, with no more than $\ell_{\max}$ messages in each list of messages, after q_S queries to the $\textsc{SndToO}$ and $\textsc{GetCred}$ oracles is at most ϵ. A scheme $\mathcal{AC}$ is said *unforgeable* if ϵ is negligible (as a function of λ) for all polynomial-time $\mathcal{A}$.

Anonymity. Regarding the anonymity, the aim of the adversary is to distinguish between two chosen honest users which one is showing his credentials. For this purpose, the adversary has access to a *left-or-right* $\textsc{LoR}$ challenge oracle. Moreover the adversary can interact with honest users as a corrupted organization in order to issue them some credentials or as a corrupted verifier with the help of the $\textsc{SndToU}$ oracles. The formal anonymity experiment $\textsc{Anonymity}_{\mathcal{A}}(\lambda)$ is given as follows.

– $(\mathsf{gpk}, (\mathsf{opk}, \mathsf{osk})) \leftarrow \textsc{Setup}(1^\lambda)$

– $b \overset{\$}{\leftarrow} \{0,1\}$

– $\mathcal{O} \leftarrow \{\textsc{AddU}(\cdot), \textsc{USK}(\cdot), \textsc{CrptU}(\cdot), \textsc{SndToU}_{\textsc{Issue}}(\cdot,\cdot), \textsc{SndToU}_{\textsc{Verify}}(\cdot,\cdot), \textsc{LoR}(b,\cdot,\cdot,\cdot)\}$

– $b' \leftarrow \mathcal{A}^{\mathcal{O}}(\mathsf{gpk}, (\mathsf{opk}, \mathsf{osk}))$

– Return 1 if $b = b'$.

A scheme $\mathcal{AC}$ is $(t, \ell_{\max}, q_C, \epsilon)$-*anonymous* if the probability for an adversary $\mathcal{A}$ to win the $\textsc{Anonymity}(\lambda)$ game in time t, with no more than $\ell_{\max}$ messages in each list of messages, after q_C queries to the $\textsc{LoR}$ oracle is at most ϵ. A scheme $\mathcal{AC}$ is said *anonymous* if ϵ is negligible (as a function of λ) for all polynomial-time $\mathcal{A}$.

3. OUR NEW PRIVACY-AWARE MODEL FOR SANITIZABLE SIGNATURES

We now formalize sanitizable signature schemes and their security according to the considerations we discussed in introduction. Our goal is to model the security requirements we need in order to use sanitizable signatures as anonymous credentials. Our work is based on [6, 11, 7, 12]. A *sanitizable signature scheme* $\mathcal{SAN}$ is given by the following seven algorithms involving one signer and several sanitizers. The verification and judge procedures only involve public data.

$\textsc{Admissible modifications}$. As in [6, 11, 7], admissible modifications are modelled by two functions $\textsc{adm}$ and $\textsc{mod}$. $\textsc{adm}$ contains a description of (i) a division of a message $m \in \{0,1\}^*$ of length t into N blocks of respective lengths $\{t_n\}_{n=1}^{N}$ such that $\sum_{n=1}^{N} t_n = t$, (ii) a subset $A \subseteq [1, N] \subseteq \mathbb{N}$ indicating the indexes of the modifiable blocks. $\textsc{adm}$ being fixed, the function $\textsc{fix}_{\textsc{adm}}$ maps a message m to the its fixed part m_{FIX}, *i.e.* the concatenation of all non-admissible blocks. By misuse of notation, A is also denoted $\textsc{adm}$.

$\textsc{mod}$ maps a message m to a modified message $m' = \textsc{mod}(m)$ and is essentially a set of pairs (n, m'_n) specifying that the block n has to be replaced by the message m'_n. $\textsc{adm}(\textsc{mod}) \in \{0,1\}$ indicates whether the modification instructions $\textsc{mod}$ matches the admissible modifications $\textsc{adm}$.

$\textsc{Setup}$. This algorithm takes as input a security parameter λ and outputs global parameters gpk, implicitly given to all algorithms as auxiliary input. A secret key tsk for tracing is produced as well. Finally, the key pair $(\mathsf{spk}, \mathsf{ssk})$ for the signer is also produced.

$\textsc{SanKG}$. This algorithm, on input $j \in \mathbb{N}$, produces the key pair (denoted $(\mathsf{pk}_{\mathsf{san}}[j], \mathsf{sk}_{\mathsf{san}}[j])$) for the j-th sanitizer.

$\textsc{Sign}$. This algorithm takes as input the secret signing key ssk, a sanitizer public key $\mathsf{pk}_{\mathsf{san}}[j]$, a message $m \in \{0,1\}^*$ and a description $\textsc{adm}$ *w.r.t.* m. It outputs a signature σ (or an error message $\perp$). We assume that $\textsc{adm}$ can always be recovered from a signature σ.

$\textsc{Sanitize}$. This algorithm takes as input a sanitizer secret key $\mathsf{sk}_{\mathsf{san}}[j]$, the signer public key spk, a message $m \in$

$\{0,1\}^*$, a signature σ and the modification instructions MOD the sanitizers wants to carry out (according to ADM). It outputs a message m' and a signature σ' (or an error message $\bot$).

Verify. This algorithm takes as input the signer public key spk, a message m and a signature σ. It outputs a bit $b \in \{0,1\}$.

FindSan. This algorithm takes as input the tracing key tsk, the signer public key spk, a message m and a signature σ. It outputs the index $j \in \mathbb{N}$ of a sanitizer and a proof τ ($j = 0$ means that the algorithm can not trace the signature).

Judge. This algorithm takes as input the signing public key spk, a sanitizer public key $\mathsf{pk_{san}}[j]$, a message m, a signature σ and a proof τ. It outputs a bit $b \in \{0,1\}$, indicating whether the tracing is correct.

Set of Admissible Values. As we need the `LimitSet` extension [11], we introduce the set $\mathcal{V} := \{V_i \subseteq \{0,1\}^{t_i} \mid i \in \textsc{adm}\}$. Each V_i defines the set for the modifications that can be done by the sanitizer for the block m_i. Following [11], the Sign and Sanitize procedures are not modified, but the definition ADM is modified in order to contain the public set $\mathcal{V}$.

Security Properties. We now focus on the security properties related to our notion of sanitizable signatures. It uses several oracles which mostly corresponds to the execution of the above procedures (we do not detail them since they exactly correspond to those given above). The sole exception is the LoR oracle which is formally described below.

$\textsc{LoR}(b, \mathsf{spk}, (j_0, m_0, \sigma_0, \textsc{mod}_0), (j_1, m_1, \sigma_1, \textsc{mod}_1))$. This oracle first stops if $\textsc{adm}_0$ and $\textsc{adm}_1$ (contained in the signatures) are different, if the signatures are not valid, or if $(m_0, \textsc{mod}_0, \textsc{adm}_0) \not\equiv (m_1, \textsc{mod}_1, \textsc{adm}_1)$ (which means that the resulting message, after modification, are different). It next sanitizes the message m_b, according to $\mathsf{sk_{san}}[j_b]$ to obtain (m'_b, σ'_b). It finally returns (m'_b, σ'_b).

Correctness. The correctness property states that honestly generated (*signing correctness*) and sanitized (*sanitizing correctness*) signatures will be accepted by the verifier, and that honestly generated proofs on valid signatures (*proof correctness*) will be accepted by the judge. The latter implies that honestly generated signatures will trace to the actual designated sanitizer.

Traceability. The traceability property states that all valid signatures should trace to a particular sanitizer. An adversary against this game aims at forging a valid signature such that the tracing procedure will not identify the designated sanitizer or will not be able to prove its charge.

```
– $(\mathsf{gpk}, \mathsf{tsk}, (\mathsf{ssk}, \mathsf{spk})) \leftarrow \textsc{Setup}(1^\lambda)$
– $(m^*, \sigma^*) \leftarrow \mathcal{A}^{\textsc{Sign}(\mathsf{ssk},\cdot,\cdot,\cdot)}(\mathsf{gpk}, \mathsf{tsk}, \mathsf{spk})$
– $(j^*, \tau^*) \leftarrow \textsc{FindSan}(\mathsf{tsk}, \mathsf{spk}, m^*, \sigma^*)$
– Return 1 if
    – $\textsc{Verify}(\mathsf{spk}, m^*, \sigma^*) = 1$ and
    – $j^* = 0$ or $\textsc{Judge}(\mathsf{spk}, \mathsf{pk_{san}}[j^*], m^*, \sigma^*), \tau^*) = 0$.
```

Immutability. The immutability property states that a sanitizer will not sanitize a part of the message he is not allowed to sanitize. In our case, this can happen either if a modification on a fixed part has been done or if a modification on a modifiable part has been done with a disallowed value.

```
– $(\mathsf{gpk}, \mathsf{tsk}, (\mathsf{ssk}, \mathsf{spk})) \leftarrow \textsc{Setup}(1^\lambda)$
– $(m^*, \sigma^*) \leftarrow \mathcal{A}^{\textsc{Sign}(\mathsf{ssk},\cdot,\cdot,\cdot)}(\mathsf{gpk}, \mathsf{tsk}, \mathsf{spk})$
– $(j^*, \tau^*) \leftarrow \textsc{FindSan}(\mathsf{tsk}, \mathsf{spk}, m^*, \sigma^*)$
– Return 1 if
    – $\textsc{Verify}(\mathsf{spk}, m^*, \sigma^*) = 1$ and
    – $\textsc{Judge}(\mathsf{spk}, \mathsf{pk_{san}}[j^*], m^*, \sigma^*, \tau^*) = 1$ and
    – $\forall(\mathsf{pk_{san}}[j], m_i, \textsc{adm}_i)$ queried to the Sign oracle s.t. $\mathsf{pk_{san}}[j^*] = \mathsf{pk_{san}}[j]$, $m^* \notin \{\textsc{mod}(m_i) | \textsc{mod} \subseteq \textsc{adm}_i\}$.
```

Unforgeability. The unforgeability property states that nobody is able to forge a valid signature without the signing secret key, nor any sanitizer's secret key. An adversary against this property is given two challenge keys, a signer public key and a sanitizer public key, together with oracles for signing and sanitizing.

```
– $(\mathsf{gpk}, \mathsf{tsk}, (\mathsf{ssk}, \mathsf{spk})) \leftarrow \textsc{Setup}(1^\lambda)$
– $(\mathsf{pk_{san}}[j^*], \mathsf{sk_{san}}[j^*]) \leftarrow \textsc{SanKG}()$
– $\mathcal{O} \leftarrow \{\textsc{Sign}(\mathsf{ssk}, \cdot, \cdot, \cdot), \textsc{Sanitize}(\mathsf{sk_{san}}[j^*], \cdot, \cdot, \cdot, \cdot)\}$
– $(m^*, \sigma^*) \leftarrow \mathcal{A}^{\mathcal{O}}(\mathsf{gpk}, \mathsf{tsk}, \mathsf{spk}, \mathsf{pk_{san}}[j^*])$
– $(j, \tau) \leftarrow \textsc{FindSan}(\mathsf{tsk}, \mathsf{spk}, m^*, \sigma^*)$
– Return 1 if
    – $j = j^*$ and
    – $\textsc{Verify}(\mathsf{spk}, m^*, \sigma^*) = 1$ and
    – $\forall(\mathsf{pk_{san}}[j], m_j)$ queried to the Sign oracle, we have $(\mathsf{pk_{san}}[j^*], m^*) \neq (\mathsf{pk_{san}}[j], m_j)$ and
    – $m^*$ has never been queried to the Sanitize oracle.
```

Remark 1. The accountability properties [6, 11, 7] state that signers and sanitizers are responsible for the signatures they produced but we do not need them for the security of our main construction. As long as we do not have an opening algorithm in the anonymous credential system, we do not need the accountability properties.

Unlinkability. The unlinkability property states that it is infeasible to distinguish sanitized signatures that have been produced from the same message and / or on the same sanitizer. We only have a trace-restricted unlinkability notion, since an adversary could easily win the game with a trace oracle on her challenge. The adversary is given a *left-or-right* oracle described above. Moreover, the adversary may add new sanitizers in the system thanks to the SanKG oracle.

```
– $(\mathsf{gpk}, \mathsf{tsk}, (\mathsf{ssk}, \mathsf{spk})) \leftarrow \textsc{Setup}(1^\lambda)$
– $b \xleftarrow{\$} \{0,1\}$
– $\mathcal{O} \leftarrow \{\textsc{SanKG}(), \textsc{Sign}(\mathsf{ssk}, \cdot, \cdot, \cdot), \textsc{Sanitize}(\cdot, \mathsf{spk}, \cdot, \cdot, \cdot), \textsc{LoR}(b, \mathsf{spk}, \cdot, \cdot), \textsc{FindSan}(\mathsf{tsk}, \mathsf{spk}, \cdot, \cdot)\}$
– $b' \leftarrow \mathcal{A}^{\mathcal{O}}(\mathsf{gpk}, \mathsf{spk})$
– Return 1 if
    – $b = b'$ and
    – $\mathcal{A}$ has not queried any $(m', \sigma')$ output by the LoR oracle to the FindSan oracle.
```

Remark 2. We could ask for a stronger notion than the existing one in the state of the art (cf. [7]), the notion of *strong unlinkability*. According to this notion, the sanitizable signatures must be unlinkable even if the adversary is given the secret signing key. Formally, the game is the same as the standard unlinkability game, except that the adversary is given the secret signing key (and, in this case, a Sign oracle is no more useful).

Remark 3. The transparency property [6, 11, 7] states that it is infeasible to distinguish a sanitized signature from a freshly generated signature. In our case, this is not a matter of concern if a verifier notices that a value has been hidden.

4. FROM SANITIZABLE SIGNATURES TO ANONYMOUS CREDENTIALS

We are now in possession of all the elements we need to build anonymous credentials from sanitizable signatures.

4.1 A Black-box Reduction

Given a sanitizable signature scheme $\mathcal{SAN}$ as described in the above section, we define the following anonymous credential system $\mathcal{AC}$. We consider in the following that the character # stands for predefined non relevant symbol that will be used by sanitizers to hide an information.

Setup and Key Generation. The $\mathcal{AC}$.Setup procedure runs the $\mathcal{SAN}$.Setup procedure. We thus set the global parameter gpk as that from the sanitizable signature scheme, the organization key pair $(\mathsf{opk}, \mathsf{osk})$ corresponds to the one of the signer (output by $\mathcal{SAN}$.Setup). Next, each user can execute the $\mathcal{AC}$.UserKG by using exactly the $\mathcal{SAN}$.SanKG of the sanitizable signature scheme, becoming a sanitizer.

Credential Issuing. On input the public key $\mathsf{upk}[j]$ of a requesting user and a list of messages / attributes $\{m_n\}_{n=1}^{N} \in \{0,1\}^*$ (m_n of length t_n) related to this user, the organization O specifies $\text{ADM} \subseteq [1, N]$ at his convenience[3]. A specific field with index n is next chosen. It will be used by the verifier to verify the freshness of the Show / Verify protocol.

Afterwards, for all admissible blocks with an index $n \neq \mathsf{n}$, the signer defines a set of admissible values $\mathcal{V}_n$ which contains 2 elements: a first element corresponding to the true sub-message m_n he is signing (*i.e.* the user attribute) and a second element corresponding to some predefined non relevant symbol (like several # characters) of adequate length (a black value for an admissible block is always assumed to be allowed). Then, O produces the signature

$$\sigma := \mathcal{SAN}.\text{Sign}(\mathsf{osk}, \mathsf{upk}[j], m, \text{ADM})$$

and sends σ to U. User U records σ as his credential C.

Credential Showing. We consider a verifier who aims at checking that user U_j has previously obtained credentials on some attributes $\mathbb{M} = \{m'_n\}_{n=1}^{N'}$. Let us assume that user U_j obtained a credential C (a sanitizable signature) on some attributes $\{m_n\}_{n=1}^{N}$ such that $\mathbb{M} \subseteq \{m_n\}_{n=1}^{N}$.

The verifier first sends a random value r to user U_j, depending of the context of the verification, like the concatenation of some verifier's data and the current time. Then the user U_j sanitizes his credential according to

$$\text{MOD} := \{(\mathsf{n}, \mathsf{r})\} \cup \left(\bigcup_{n' | m'_n \notin \mathbb{M}} \{(n', \#)\}\right),$$

and computes the pair

$$(m', \sigma') \leftarrow \mathcal{SAN}.\text{Sanitize}(\mathsf{usk}[j], \mathsf{opk}, m, \sigma, \text{MOD}).$$

In a nutshell, the user sanitizes the attributes he wants to hide (replacing them by some # characters) and set the sub-message with index n to the random challenge r. The user finally sends (m', σ') to the verifier who checks the resulting signature by testing whether $\mathcal{SAN}.\text{Verify}(\mathsf{opk}, m', \sigma') = 1$ (and verifying the use of value r).

4.2 Security Analysis

The security of the resulting construction relies on the underlying building block: we prove the security of the $\mathcal{AC}$ scheme under the assumption that $\mathcal{SAN}$ is a secure sanitizable signature scheme in the sense of Section 3.

4.2.1 From $\mathcal{SAN}$.{ Traceability, Immutability, Unforgeability } to $\mathcal{AC}$.Unforgeability

First of all, we define the extractor of the unforgeability experiment to be the execution of the tracing algorithm FindSan. In our scheme, a showing protocol corresponds to the showing of a modified document (m', σ'). Given such a pair, the FindSan algorithm is able to return the identity of the designated sanitizer, *i.e.* the identity of the user.

Let now $\mathcal{A}$ be an adversary against the unforgeability of the $\mathcal{AC}$ scheme. The winning condition for the experiment is: for j returned by ESetup, for all $\{m_n\}_{n=1}^{N} \in \mathsf{reg}[j]$, there exists $\mathsf{n}' \in [1, N']$ s.t. $m_{\mathsf{n}'} \notin \{m_n\}_{n=1}^{N}$. Informally, no user (honest or not) has obtained a certification on the whole specified list. In other words, for each delivered credential, there is at least an attribute which is not in the list.

Let us consider the following cases, in excluding order.

Type I. The (m', σ') pair is not traceable. We construct an adversary against the traceability property of the sanitizable signature scheme.

Type II. Otherwise, let assume that the tracing of σ' returned an index j^*. If $j^* \in HU$ is honest, then we construct an adversary against the unforgeability property of the sanitizable signature scheme.

Type III. Otherwise, $j^* \in CU \cup KU$ is dishonest or his keys have been compromise. We then construct an adversary against the immutability property of the sanitizable signature scheme.

Let us describe each adversary in more details. The challenger maintains the HU, KU and CU lists of users.

Type I. $\mathcal{A}$ successfully pass a showing protocol, but the signature (m', σ') is not traceable, meaning that the extractor is not able to identify a user underlying the view of the protocol. This means that the tracing algorithm is not able to trace the signature (m', σ'). We construct an adversary $\mathcal{B}$ against the traceability of $\mathcal{SAN}$. The AddU oracle is simulated by generating fresh key pairs. The SndToO oracle, together with the GetCred oracle, on list $\{m_n\}_{n=1}^{N}$ are simulated thanks to $\mathcal{B}$'s own Sign oracle on a message m constructed following the issuing protocol. The SndToU$_{\text{Verify}}$ oracle is simulated by computing the authentication ($\mathcal{B}$ owns users' keys). When $\mathcal{A}$ successfully pass the protocol, (m', σ') is not traceable, and the game is won.

Type II. $\mathcal{A}$ successfully pass the showing protocol, and the FindSan algorithm returns j^* such that $j^* \in HU$. We construct an adversary $\mathcal{B}$ against the unforgeability property of the $\mathcal{SAN}$ scheme. $\mathcal{B}$ gets two challenge keys (a signer one and a sanitizer one). $\mathcal{B}$ generates other keys and guesses that the forgery will concern his challenge key. The SndToO and GetCred oracles on lists $\{m_n\}_{n=1}^{N}$ are simulated thanks to the own Sign challenger's oracle on a message m built as in

[3] In practice, static fields are stated to non admissible while the true user attributes are stated to admissible (see our example in the introduction).

the issuing protocol. The $\textsc{SndToU}_{\textsc{Verify}}$ oracle is simulated by $\mathcal{B}$ with his own sanitization oracle (or computes the response himself if he knows the keys). Let now assume that $\mathcal{A}$ successfully pass the showing protocol. Since j^* has not got the whole specified credential, then (from the winning condition) the $\textsc{Sign}$ oracle has not been queried on the list. Since the sanitization oracle is queried only on previously generated signatures, the list did not appear in the sanitization oracle queries. We then got a valid and non-trivial forgery (m', σ') and the game is won.

Type III. $\mathcal{A}$ successfully pass the showing protocol, and the $\textsc{FindSan}$ algorithm returns j^* such that $j^* \in CU \cup KU$. We construct an adversary $\mathcal{B}$ against the immutability property of the $\mathcal{SAN}$ scheme. The $\textsc{AddU}$, $\textsc{SndToO}$, $\textsc{GetCred}$ oracles on attributes lists $\{m_n\}_{n=1}^{N}$ and the $\textsc{SndToU}_{\textsc{Verify}}$ oracle are simulated as in the two games above. When $\mathcal{A}$ successfully pass the showing protocol, the winning condition says that for all $\{m_n\}_{n=1}^{N} \in \mathsf{reg}[j^*]$, there exists $\mathsf{n}' \in [1, N']$ such that $m_{\mathsf{n}'} \notin \{m_n\}_{n=1}^{N}$. We want to show that the immutability winning condition is satisfied. The last condition says that for all $(j_k, m_k, \textsc{adm}_k)$ queried during the attack, we have $\mathsf{pk}_{\mathsf{san}}[j^*] \neq \mathsf{pk}_{\mathsf{san}}[j_k]$ or $m^* \notin \{\textsc{mod}(m_k) \mid \textsc{mod}$ s.t. $\textsc{adm}_k(\textsc{mod}) = 1\}$. This means that, when the $\mathsf{pk}_{\mathsf{san}}[j^*]$ key appeared in a query to the $\textsc{Sign}$ oracle, the message m^* did not come from the modification of one of the m_k queried to the oracle. One can easily be convinced of that, since for all obtained credential (*i.e.* call to the $\textsc{Sign}$ oracle), there exists m^* which was not involved. These attributes can not be the masking value, since the latter is always allowed. The game against the immutability is then won.

In the beginning of the game, the challenger chooses one of the three game above with uniform probability. Therefore, the security loss is $1/3|\mathcal{U}|$, which is still negligible if the underlying probabilities are negligible. □

4.2.2 From $\mathcal{SAN}$.Unlinkability to $\mathcal{AC}$.Anonymity

The anonymity property of the anonymous credential system directly relies on the unlinkability fo the sanitizable signature scheme. Let $\mathcal{A}$ be an adversary against the anonymity of the $\mathcal{AC}$ scheme. We construct an adversary $\mathcal{B}$ against the unlinkability of the $\mathcal{SAN}$ scheme. Regarding the *left-or-right* challenge, $\mathcal{A}$ asks for a showing protocol from the user j_0 or j_1 on a list $\{m'_{n'}\}_{n'=1}^{N'}$ such that both users already got a credential on them. Since both credentials have already been issued, $\mathcal{B}$ retrieves them and computes a query to his own oracle with the corresponding modification instructions $\textsc{mod}$. Finally, $\mathcal{A}$ returns a bit b that $\mathcal{B}$ returns to her own challenge. The probability that $\mathcal{B}$ wins is that of $\mathcal{A}$.

We note that, if $\mathcal{SAN}$ is a strongly unlinkable sanitizable scheme, then the credential system is anonymous against a corrupted organization. If $\mathcal{SAN}$ is an unlinkable sanitizable scheme, then the credential system is proven anonymous only with respect to an honest organization, and the adversary *is not given* the organization secret key. In other words, the anonymity level of the credential system depends on the unlinkability level of the underlying signature scheme.

5. A CONCRETE CONSTRUCTION

We now give a concrete sanitizable signature scheme that meets the features described in Section 3 and that can thus directly be used to design a secure anonymous credential system as shown just above.

5.1 Useful tools

Let us consider an asymmetric bilinear environment $(p, \mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T, e)$ where p is prime, $\mathbb{G}_1$, $\mathbb{G}_2$ and $\mathbb{G}_T$ three groups or order p, and e a map $e : \mathbb{G}_1 \times \mathbb{G}_2 \rightarrow \mathbb{G}_T$ such that (bilinearity) for all $g_1, g_2 \in \mathbb{G}_1 \times \mathbb{G}_2$ and $a, b \in \mathbb{Z}_p$, we have $e(g_1^a, g_2^b) = e(g_1, g_2)^{ab} = e(g_1^b, g_2^a)$ and (non-degenerate) for all $g_1, g_2 \in \mathbb{G}_1 \setminus \{\mathbf{1}_{\mathbb{G}_1}\} \times \mathbb{G}_2 \setminus \{\mathbf{1}_{\mathbb{G}_2}\}$, we have $e(g_1, g_2) \neq \mathbf{1}_{\mathbb{G}_T}$.

Proofs of knowledge. Our scheme makes use of zero-knowledge proofs of knowledge (ZKPK) which correspond to interactive protocols during which a prover convinces a verifier that he knows a set $(\alpha_1, \ldots, \alpha_q)$ of secret values verifying a given relation R without revealing any information about the known secrets. We denote by $\textsc{PoK}\{\langle \alpha_1, \ldots, \alpha_q \rangle : \mathsf{R}(\alpha_1, \ldots, \alpha_q)\}$ such a proof of knowledge. A signature of knowledge, denoted $\textsc{SoK}$ in the following and taking on input a message m, is a ZKPK that has been transformed into a signature scheme, using the Fiat-Shamir heuristic [19], and secure in the random oracle model[4].

Pseudo-random functions. We will also need pseudo-random functions (PRF). A PRF is given by a pair of algorithms $\mathcal{PRF} := \{\textsc{KeyGen}, \text{PRF}\}$ (*cf.* Appendix A.2).

Bilinear signatures. Finally, the signer is in the following able to produce a particular signature related to the Boneh-Boyen one [3]. Let $\mathcal{PRF} := \{\textsc{KeyGen}, \text{PRF}\}$ be a pseudo-random function and $(p, \mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T, e)$ be an asymmetric bilinear environment. During the $\textsc{KeyGen}$ procedure, the signer picks $g_1, u_1, \ldots, u_\ell, v \xleftarrow{\$} \mathbb{G}_1$, $g_2 \xleftarrow{\$} \mathbb{G}_2$ and $\gamma \xleftarrow{\$} \mathbb{Z}_p$, and next computes $X := {g_2}^{\gamma}$ and $Y := {h_2}^{\gamma}$. Next, $\mathsf{pk} := (X, Y)$, $\mathsf{sk} := \gamma$ and parameters are

$$(p, \mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T, e, g_1, u_1, \ldots, u_\ell, v, g_2, h_2)$$

During the $\textsc{Sign}$ procedure, the signer generates two scalars $r \leftarrow \text{PRF}(k, 0\|m_1\|m_2\| \ldots \|m_\ell)$ and $s \leftarrow \text{PRF}(k, 1\|m_1\|m_2 \| \ldots \|m_\ell)$ thanks to the pseudo-random function, sets $S := (g_1 {u_1}^{m_1} \cdots {u_\ell}^{m_\ell} v^s)^{\frac{1}{\gamma + r}}$ and returns $\sigma := (S, r, s)$.
The $\textsc{Verify}$ algorithm consists in checking that

$$e(S, {g_2}^r X) \stackrel{?}{=} e(g_1 {u_1}^{m_1} \cdots {u_\ell}^{m_\ell} v^s, g_2).$$

This scheme is studied in details in Appendix A.3.

5.2 Algorithms

5.2.1 Setup

Global parameters. An asymmetric bilinear group environment $(p, \mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T, e)$ is generated. Random generators g_1, u_1, u_2, $v \xleftarrow{\$} \mathbb{G}_1$ and $g_2 \xleftarrow{\$} \mathbb{G}_2$ are chosen, together with $\alpha \xleftarrow{\$} \mathbb{Z}_p$. $h_1 := {g_1}^{\alpha}$ and $h_2 := {g_2}^{\alpha}$ are computed. Let $\mathcal{H}_1 : \{0,1\}^* \longrightarrow \mathbb{Z}_p$ be a cryptographic hash function. The global parameters of our system are

$$\mathsf{gpk} := (p, \mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T, e, g_1, u_1, u_2, v, h_1, g_2, h_2, \mathcal{H}_1)$$

and the tracing key is $\mathsf{tsk} := \alpha$.

Signer's keys. The signer picks $\gamma \xleftarrow{\$} \mathbb{Z}_p$ and computes $X := {g_2}^{\gamma}$ and $Y := {h_2}^{\gamma}$ as the keys for the signature scheme. It also picks a key for the pseudo-random function $k \leftarrow \mathcal{PRF}.\textsc{KeyGen}(1^{\lambda})$. The signer's public key is $\mathsf{spk} := (X, Y)$ and the corresponding secret key is $\mathsf{ssk} := (\gamma, k)$.

[4] For example, $\textsc{SoK}\{\langle \alpha \rangle : h = g^{\alpha}\}(m)$ denotes a Schnorr proof of knowledge of a discrete logarithm transformed into a signature scheme by the Fiat-Shamir heuristic.

Sanitizers' keys. Finally, each sanitizer j picks $y_j \xleftarrow{\$} \mathbb{Z}_p$, sets $\mathsf{pk}_{\mathsf{san}}[j] := z_j = u_1^{y_j}$ and $\mathsf{sk}_{\mathsf{san}}[j] := y_j$. Each value (j, z_j) is registered in the table $\mathsf{pk}_{\mathsf{san}}$.

5.2.2 Structure of a signature

Let us now explain the idea behind our construction. Let us assume that a signer wants to sign a message $m \in \{0,1\}^*$ of length t, split into N blocks of length $\{t_n\}_{n=1}^{N}$ with admissible blocks $M \subseteq \{1, \ldots, N\}$ and the specific field $\mathsf{n} \in \mathbb{N} \setminus M$. According to our needs, we have $\textsc{adm} := \{(\mathsf{n}, \top)\} \cup \{(n, \bot) \mid n \in M\}$, which means that (i) m_{n} could be modified with any value $m'_{\mathsf{n}} \in \{0,1\}^{t_{\mathsf{n}}}$ while (ii) a value m_n for $n \in M$ should only be replaced, if needed, by a specific black block of message. The designated sanitizer with index j is represented by his value z_j. A sanitizable signature is next divided into three parts.

Handling the fixed part. The fixed part m_{FIX} of the message is handled by giving a signer's signature on both y_j (the sanitizer's secret) and h_{FIX} (a hash value of the fixed part), using the key generated by the signer during the setup phase. The resulting signature corresponds to three elements (S, r, s) where S is a group element and where the randomness (r, s) are derived via the $\mathcal{PRF}$.

However, to reach unlinkability, we should not give directly the signature and we use for this purpose encryption techniques. We thus compute an ElGamal encryption [21] $(B_{0:1}, B_{0:2})$ of the S part of the signature. During the Sign procedure, the signer picks at random $t_0 \in \mathbb{Z}_p$ and generates $c_1 \leftarrow \mathcal{H}_1(m_{\mathrm{FIX}} \| \textsc{adm})$. It next generates the randomness of the signature scheme as $r \leftarrow \mathrm{PRF}(k, 0\|0\|m_{\mathrm{FIX}}\|\textsc{adm})$ and $s \leftarrow \mathrm{PRF}(k, 0\|1\|m_{\mathrm{FIX}}\|\textsc{adm})$, and computes the signature $S := (g_1 z_j {u_2}^{c_1} v^s)^{\frac{1}{\gamma+r}}$. The ElGamal ciphertext of S is finally $(B_{0:1} := {g_1}^{-t_0}, B_{0:2} := S{h_1}^{t_0})$ and is sent to the sanitizer (while the initial signature is not). During the Sanitize procedure, as the ElGamal encryption scheme is randomizable, the sanitizer can easily randomize such a couple $(B_{0:1}, B_{0:2})$, which provides us the unlinkability property. For this purpose, the sanitizer chooses $t'_0 \in \mathbb{Z}_p$ and computes $(B'_{0:1} := B_{0:1}{g_1}^{-t'_0}, B'_{0:2} := B_{0:2}{h_1}^{t'_0})$.

Handling the admissible part. As an admissible sub-message is necessarily an accepted sub-message, the sanitizer may not modify it but the verifier should be convinced that it has been validated by the signer. For this purpose, we use the same technique as for the fixed part, but for each admissible sub-messages m_n, independently of each other. Next each admissible block is signed and next encrypted, as above (again to reach the unlinkability property). During the Sign procedure, the signer picks, for all $n \in M$, a value $t_i \xleftarrow{\$} \mathbb{Z}_p$, computes $h_n \leftarrow \mathcal{H}_1(m_{\mathrm{FIX}}\|\textsc{adm}\|n\|m_n)$ thanks to the hash function and $r_n \leftarrow \mathrm{PRF}(k, n\|0\|m_{\mathrm{FIX}}\|\textsc{adm}\|m_n)$ and $s_n \leftarrow \mathrm{PRF}(k, n\|1\|m_{\mathrm{FIX}}\|\textsc{adm}\|m_n)$ thanks to the pseudo-random function. The signature is $G_n := (g_1 z_j {u_2}^{h_n} v^{s_n})^{\frac{1}{(\gamma+r_n)}}$ and the resulting ciphertext is $(B_{n:1} := {g_1}^{-t_n}, B_{n:2} := G_n {h_1}^{t_n})$. All these steps are repeated for all admissible parts of the message. During the Sanitize procedure, the modification instructions $\textsc{mod}$ are given by the set $\{(\mathsf{n}, m'_{\mathsf{n}})\} \cup \{(n, \bot)\}_{n \in J}$, meaning that the sanitizer wants to replace m_{n} by m'_{n} and hide a subset $J \subseteq M$. There are consequently two cases for admissible parts, as explained above.

In the first case, the sanitizer is able to modify as desired an admissible part of the message. As we do not need the transparency property (contrary to existing sanitizable signature constructions), the sanitized message-signature pair can still include the ElGamal ciphertext of the signer's signature with the initially signed sub-message m_i. The new sub-message m'_n should however be signed by the authorized sanitizer, which will be done in the proof of validity of the whole signature below. Moreover, as we need unlinkability, the ElGamal ciphertext should be randomized, as for the fixed part above. Next, for all $n \in M \setminus J$, the sanitizer again randomizes each ElGamal ciphertext by choosing at random $t'_{\mathsf{n}} \in \mathbb{Z}_p$ and computing $(B'_{n:1} := B_{n:1}{g_1}^{-t'_n}, B'_{n:2} := B'_{n:2}{h_1}^{t'_n})$.

In the second case, the sanitizer can only replace the admissible sub-message by a sequence of $|m_n|$ times the '#' character. As this black value is, by convention, always allowed, we do not need to supply a proof that the value is indeed authorized (contrary to [11]). However, as the initially signed sub-message should not be retrieved, we ask the sanitizer to remove the corresponding ElGamal ciphertext. Thus, for all $n \in J$, $(B_{n:1}\|B_{n:2})$ is removed from the sanitized signature. Any other modification of such part of the message will be obviously refused by the verifier who has access to the variable $\textsc{adm}$.

Proof of validity of the whole signature. It finally remains to prove the validity of the signature. In fact, from the sanitizer's point of view, we only need to prove the knowledge of y_j in the above relations to prove that this signature comes from an authorized entity (since y_j is signed S). The verification procedure of a signature whether coming from Sign or Sanitize should be unique. However, in our case, the signer does not know the secret value y_j. Our solution consists in using a proof of the "or" statement [18, 27].

5.2.3 Signing, sanitizing, verifying and tracing

Putting all these elements together, a signature on a message m *w.r.t.* admissible modifications $\textsc{adm}$ is a signature of knowledge, on the message

$$\mathsf{M} := B_{0:1}\|B_{0:2}\|r\|s\| \bigcup_{n \in M} (B_{n:1}\|B_{n:2}\|r_n\|s_n)\|m\|\textsc{adm}\|X\|Y$$

that should be produced as:

$$U := \mathrm{SoK}\Bigg\{ \langle \rho \rangle \ : \ e(g_1, g_2)^{\rho} = e(g_1, X) \vee \\ \Bigg(e(u_1, g_2)^{\rho} = \frac{e(B_{0:2}, {g_2}^{r} X) e(B_{0:1}, {h_2}^{r} Y)}{e(g_1 {u_2}^{c_1} v^{s}, g_2)} \ \wedge \forall n \in M, \\ e(u_1, g_2)^{\rho} = \frac{e(B_{n:2}, {g_2}^{r_n} X) e(B_{n:1}, {h_2}^{r_n} Y)}{e(g_1 {u_2}^{h_n} v^{s_n}, g_2)} \Bigg) \Bigg\}(\mathsf{M})$$

Signing. During the Sign procedure, the signer, knowing γ such that $X = {g_2}^{\gamma}$ can perform the first part of the proof.

Sanitizing. During the Sanitize procedure, the sanitizer, knowing y_j can perform the second part of the "or" proof, but this time with the message $\mathsf{M}' :=$

$$B'_{0:1}\|B'_{0:2}\|r\|s\|\|\bigcup_{n \in M \setminus J} (B'_{n:1}B'_{n:2}\|r_n\|s_n)\|m'\|\textsc{adm}\|X\|Y.$$

Verification. A sanitizable signature is next composed of the ElGamal ciphertexts together with the randomness r, r_1, $\ldots$, $s, s_1, \ldots$ and signature of knowledge U. The verification procedure simply consists in verifying the signature U.

Tracing. The authority in possession of $\mathsf{tsk} := \alpha$ can decrypt the ElGamal ciphertext $(B_{0:1}, B_{0:2})$ to retrieve $S^* = B_{0:2}{B_{0:1}}^{\alpha}$, and use the table $\mathsf{pk}_{\mathsf{san}}$ to retrieve an entry (j, z_j) such that $e(S^*, {g_2}^r X) = e(g_1 z_j {u_2}^{c_1} v^s, g_2)$. The proof of validity of such a tracing is done by proving that the decryption has been correctly done, using the knowledge of α.

Security . The following theorem is proved in Appendix A.

Theorem 1 *The $\mathcal{SAN}$ scheme is a traceable, immutable, unforgeable and unlinkable signature scheme in the sense of the Section 3 if the* DDH *problem is hard in $\mathbb{G}_1$ and the* SDH *problem is hard in $(\mathbb{G}_1, \mathbb{G}_2)$, in the random oracle model.*

5.3 Comparisons

We have now introduced all the useful elements to make a quantitative comparison between related works and an anonymous credential system based on the sanitizable signature of this section. We then give in Figure 1 several elements of comparison between our solutions and [25, 24, 14]. “This1” is the scheme described above, while “This2” is a variant with some optimizations described in Appendix B.

Underlying algebraic structures may be RSA environment [24], $\mathbb{Z}_n^*$ with a subgroup of prime order q [25], or bilinear groups $(\mathbb{G}_1, \mathbb{G}_2)$ over a base field $\mathbb{F}_p$ [14]. L denotes the maximum number of attributes issued by an authority into a single credential. Bandwidth columns denote the exchanged quantity of data during the protocols runnings, whereas the other columns denote a cost either in memory (size of a public key or a certificate) or in computation (number of operations in the underlying algebraic structure). We give asymptotic complexity as well as some concrete sizes in bits.

As said before, Figure 1 explicitly shows that our solutions are direct signatures, and thus, the issuing process is for free for the user. Moreover, in our variant This2, our multiple-use credential is rather short, compared to existing schemes. UProve’s technology is based on blind signatures, so brings very efficient but single-use credentials. The other solutions bring multiple-use credentials.

6. CONCLUSION

In this paper, we proposed a new way to design anonymous credential systems. As an interesting remark, one can notice that an existing way to design anonymous credential (as stated before) is to use group signatures. The question now is the exact link between these primitives. How do articulate each other group signatures, (strongly) unlinkable sanitizable signatures and anonymous credentials ?

Acknowledgements

This work has been supported by the ANR-11-INS-0013 LYRICS Project. We are also grateful to Amandine Jambert for helpful discussions on sanitizable signatures, to Sherman Chow for his help on this final version and to anonymous referees for their valuable comments.

7. REFERENCES

[1] Giuseppe Ateniese, Daniel H. Chou, Breno de Medeiros, and Gene Tsudik. Sanitizable signatures. In *ESORICS ’05*, pages 159–177, 2005.

[2] Mira Belenkiy, Melissa Chase, Markulf Kohlweiss, and Anna Lysyanskaya. P-signatures and noninteractive anonymous credentials. In *TCC*, pages 356–374, 2008.

[3] Dan Boneh and Xavier Boyen. Short signatures without random oracles and the sdh assumption in bilinear groups. *Journal of Cryptology*, 21(2):149–177, 2008.

[4] Stefan Brands. *Rethinking PKI and digital certificates - building in privacy*. PhD thesis, Eindhoven Institute of Technology, 1999.

[5] Christina Brzuska, Heike Busch, Özgür Dagdelen, Marc Fischlin, Martin Franz, Stefan Katzenbeisser, Mark Manulis, Cristina Onete, Andreas Peter, Ber tram Poettering, and Dominique Schröder. Redactable signatures for tree-structured data: Definitions and constructions. In *ACNS ’10*, pages 87–104, 2010.

[6] Christina Brzuska, Marc Fischlin, Tobias Freudenreich, Anja Lehmann, Marcus Page, Jakob Schelbert, Dominique Schröder, and Florian Volk. Security of sanitizable signatures revisited. In *PKC ’09*, pages 317–336, 2009.

[7] Christina Brzuska, Marc Fischlin, Anja Lehmann, and Dominique Schröder. Unlinkability of sanitizable signatures. In *PKC ’10*, pages 444–461, 2010.

[8] Jan Camenisch, Markulf Kohlweiss, and Claudio Soriente. An accumulator based on bilinear maps and efficient revocation for anonymous credentials. In *PKC ’09*, pages 481–500. Springer, 2009.

[9] Jan Camenisch and Anna Lysyanskaya. An efficient system for non-transferable anonymous credentials with optional anonymity revocation. In *EUROCRYPT ’01*, pages 93–118, 2001.

[10] Jan Camenisch and Anna Lysyanskaya. Signature schemes and anonymous credentials from bilinear maps. In *CRYPTO ’04*, pages 56–72, 2004.

[11] Sébastien Canard and Amandine Jambert. On extended sanitizable signature schemes. In *CT-RSA ’10*, pages 179–194. Springer, 2010.

[12] Sébastien Canard, Amandine Jambert, and Roch Lescuyer. Sanitizable signatures with several signers and sanitizers. In *AFRICACRYPT ’12*, pages 35–52. Springer, 2012.

[13] Sébastien Canard, Fabien Laguillaumie, and Michel Milhau. Trapdoor sanitizable signatures and their application to content protection. In *ACNS ’08*, pages 258–276. Springer, 2008.

[14] Sébastien Canard and Roch Lescuyer. Anonymous credentials from (indexed) aggregate signatures. In *Digital Identity Management ’11*, pages 53–62. ACM, 2011.

[15] Sébastien Canard, Roch Lescuyer, and Jacques Traoré. Multi-show anonymous credentials with encrypted attributes in the standard model. In *CANS ’11*, pages 194–213. Springer, 2011.

[16] David Chaum and Jan-Hendrik Evertse. A secure and privacy-protecting protocol for transmitting personal information between organizations. In *CRYPTO ’86*, pages 118–167. Springer, 1986.

[17] Sherman S. M. Chow, Yi Jun He, Lucas Chi Kwong Hui, and Siu-Ming Yiu. Spice – simple privacy-preserving identity-management for cloud environment. In *ACNS ’12*, pages 526–543. Springer, 2012.

[18] Ronald Cramer, Ivan Damgard, and Berry Schoenmakers. Proofs of partial knowledge and simplified design of witness hiding protocols. In *CRYPTO ’94*, pages 174–187. Springer, 1994.

	Keys (L attributes per credential)						**Issuing Protocol**		
	Group Structures		PK Size	Credential Size			User	Issuer	Bw.
[25]	$p = 1024, q = 128$		$O(L)$	$O(1)$	$\approx 3 \cdot \lvert G_q \rvert + 128$		$O(L) : G_q$	$O(L) : G_q$	$O(1)$
[24]	$\mathbb{Z}_n$: RSA 1024		$O(L)$	$O(1)$	$\approx 2600 + 1024$		$O(L) : \mathbb{Z}_n$	$O(L) : \mathbb{Z}_n$	$O(L)$
[14]	$\mathbb{F}_p : \lvert \mathbb{G}_1 \rvert \approx 170$		$O(1)$	$O(L)$	$170 \cdot L+ \approx 510 + \lvert \mathbb{G}_2 \rvert$		$O(L) : \mathbb{G}_2$	$O(L) : \mathbb{G}_1$	$O(L)$
This1	$\mathbb{F}_p : \lvert \mathbb{G}_1 \rvert \approx 170$		$O(1)$	$O(L)$	$(2L + 2) \cdot (\lvert \mathbb{G}_1 \rvert + \lvert \mathbb{Z}_p \rvert)$		$\mathbf{O(1)}$	$O(L) : \mathbb{G}_1$	$O(L)$
This2	$\mathbb{F}_p : \lvert \mathbb{G}_1 \rvert \approx 170$		$O(L)$	$O(1)$	$\mathbf{2} \cdot \lvert \mathbb{G}_\mathbf{1} \rvert + \mathbf{3} \cdot \lvert \mathbb{Z}_\mathbf{p} \rvert$		$\mathbf{O(1)}$	$O(L) : \mathbb{G}_1{}^a$	$O(1)$
	Showing Protocol								
	User	Verifier	Bw.	User	Verifier	Bw.	User	Verifier	Bw.
	Single-use			*Single-use k-out-of-L attributes*			*K-use L-attributes*		
[25]	$O(L)$		$O(L)$	$O(k) : G_q$		$O(L) : \mathbb{Z}_q$	$O(KL) : G_q$		$O(KL)$
[24]	$O(L) : \mathbb{Z}_n$		$O(1)$	$O(L) : \mathbb{Z}_n$		$O(L - k)$	$O(L) : \mathbb{Z}_n$		$O(1)$
[14]	$O(1)$ pair.; $O(L) : \mathbb{G}_2$		$O(1)$	$O(1)$ pair.; $O(L) : \mathbb{G}_2$		$O(1)$	$O(1)$ pair.; $O(L) : \mathbb{G}_2$		$O(1)$
This1	$O(1)$	$O(L)$ pair.	$O(L)$	$O(1)$	$O(L)$ pair.	$O(k)$	$O(L) : \mathbb{G}_1$	$O(L)$ pair.	$O(L)$
This2	$O(1)$	$O(L) : \mathbb{G}_1$	$O(1)$	$O(L - k) : \mathbb{G}_T{}^b$	$O(L - k) : \mathbb{G}_T{}^b$	$O(L - k)$	$O(1)$	$O(L) : \mathbb{G}_1$	$O(1)$

[a] May be accelerated as a single multi-exponentiation (contrary to [14] and This1). [b] May be accelerated as a single multi-exponentiation.

Figure 1: Comparisons between our solution and the related works

[19] Amos Fiat and Adi Shamir. How to prove yourself: Practical solutions to identification and signature problems. In *CRYPTO'86*, pages 186–194. Springer, 1987.

[20] Georg Fuchsbauer. Commuting signatures and verifiable encryption. In *EUROCRYPT '11*, pages 224–245. Springer, 2011.

[21] Taher El Gamal. A public key cryptosystem and a signature scheme based on discrete logarithms. In *CRYPTO '84*, pages 10–18. Springer, 1984.

[22] Shafi Goldwasser, Silvio Micali, and Ronald L. Rivest. A digital signature scheme secure against adaptive chosen-message attacks. *SIAM Journal of Computing*, 17(2):281–308, 1988.

[23] Jorge Guajardo, Bart Mennink, and Berry Schoenmakers. Anonymous credential schemes with encrypted attributes. In *CANS '10*, pages 314–333. Springer, 2010.

[24] IBM. Identity mixer - Idemix, 2010.

[25] Microsoft. U-Prove community technology, 2010.

[26] Kunihiko Miyazaki, Goichiro Hanaoka, and Hideki Imai. Digitally signed document sanitizing scheme based on bilinear maps. In *ASIACCS '06*, pages 343–354. ACM, 2006.

[27] Alfredo De Santis, Giovanni Di Crescenzo, Giuseppe Persiano, and Moti Yung. On monotone formula closure of SZK. In *FOCS '94*, pages 454–465. IEEE, 1994.

APPENDIX

A. PROOF OF THEOREM 1

We now analyze the security of our concrete scheme. First of all, we state the hard problems under which the security relies. Then we give a proof of security of the underlying signature scheme and, at last, we prove the main properties.

A.1 Hard Problems

Let us fix a group $\mathbb{G}$ of prime order p and a bilinear environment $(p, \mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T, e)$. For a problem X, the X assumption in $\mathbb{G}$ says that X is hard to solve in $\mathbb{G}$.

Problem 2 (DL) *Given $(g, h) \in \mathbb{G}^2$, find α s.t. $h = g^\alpha$.*

Problem 3 (DDH) *Given $(g, g^a, g^b, g^c) \in \mathbb{G}^4$, for uniformly picked $g \overset{\$}{\leftarrow} \mathbb{G}$, and $a, b \overset{\$}{\leftarrow} \mathbb{Z}_p$, decide whether $c = ad \bmod p$ or if c was also uniformly picked in $\mathbb{Z}_p$.*

Problem 4 (XDH) *Given $\mathbb{G}_1$, $\mathbb{G}_2$ and $\mathbb{G}_T$, while the* DDH *problem could be easy in $\mathbb{G}_2$, the* XDH *assumption states that the* DDH *problem is still hard in $\mathbb{G}_1$.*

Problem 5 (q-SDH) *Given $(g_1, g_1{}^\theta, g_1{}^{\theta^2}, \ldots, g_1{}^{\theta^q}, g_2, g_2{}^\theta) \in \mathbb{G}_1{}^{q+1} \times \mathbb{G}_2^2$ for a uniformly picked $\theta \overset{\$}{\leftarrow} \mathbb{Z}_p^*$, find a pair $(c, g_1{}^{1/(\theta+c)}) \in \mathbb{Z}_p \times \mathbb{G}_1$.*

A.2 Dealing with the PRF

The security of a PRF states that for any efficient algorithm $\mathcal{B}$ the value $\left|\Pr[\mathcal{B}^{\mathrm{PRF}(k,\cdot)}(1^\lambda) = 1] - \Pr[\mathcal{B}^{\mathcal{R}}(1^\lambda) = 1]\right|$ is negligible, where the first probability is taken over $k \leftarrow \textsc{KeyGen}(1^\lambda)$ and $\mathcal{B}$'s internal coin tosses, and the second probability is over the choice of the random function $\mathcal{R} : \{0,1\}^* \to \{0,1\}^\lambda$ and $\mathcal{B}$'s randomness. We replace in the security proofs all invocation to the PRF by a truly random value in $\mathbb{Z}_p$ and maintain a list of these values in order to be consistent. Then, for any successful adversary against each property we construct a successful distinguisher against the PRF. In each case, we can decide whether an adversary has won or not. In the case of the immutability, this decidability follows from the fact that we can decide if a message m' belongs to the set of the possible resulting of a sanitization of a message m with respect to admissibility $\textsc{adm}$. Indeed, it suffices to parse the message block per block and to check whether the modification has been allowed or not.

A.3 Signing Several Messages

We study here the signature scheme, denoted $\mathcal{S}^{(\ell)}$, used in our sanitizable signature scheme and prove its unforgeability.

Theorem 6 *If the* SDH *problem is (q, t', ϵ')-hard in $(\mathbb{G}_1, \mathbb{G}_2)$, then the $\mathcal{S}^{(1)}$ signature scheme is (q_S, t, ϵ)-unforgeable as long as $t \leq t' - O(q^2)$, $q_S \leq q$ and $\epsilon \leq \epsilon'/q$.*

Proof. Let $\mathcal{A}$ be an adversary that (q_S, N, t, ϵ)-win the EUF-CMA existential unforgeability game against chosen message attacks for signature schemes (cf. [22]). We set $q = q_S$. Let $(g_1, g_1{}^\theta, g_1{}^{\theta^2}, \ldots, g_1{}^{\theta^q}, g_2, g_2{}^\theta)$ be a q-SDH challenge, for an unknown and uniformly distributed $\theta \in \mathbb{Z}_p^*$. We build an algorithm $\mathcal{B}$ which outputs $(c, {g_1}^{1/(\theta+c)})$, for a $c \in \mathbb{Z}_p \backslash \{-\theta\}$.

Parameters. Pick $k \overset{\$}{\leftarrow} \{1, \dots, q\}$ and scalars $r_1, s_1 \dots, r_q$, s_q in order to simulate the signatures. For $\{r_1, \dots, r_q\} \in \mathbb{F}_p$, define polynomials P, P_-, P_{m-} on $\mathbb{F}_p[Z]$ respectively by

$$\prod_{n=1}^{q} (Z + r_n - r_k), \prod_{\substack{n=1\\n\neq k}}^{q} (Z + r_n - r_k), \prod_{\substack{n=1\\n\neq m\\n\neq k}}^{q} (Z + r_n - r_k).$$

Expanding P on θ, we get $P(\theta) = \sum_{n=0}^{q} a_n \theta^n$ for some $\{a_n\}_{n=0}^{q}$ depending on the r_n. Since we know ${g_1}^{\theta^n}$ by the q-SDH challenge, we are able to compute ${g_1}^{P(\theta)}$, without knowledge of θ. Set $X \leftarrow {g_2}^{\theta} {g_2}^{-r_k}$ as a challenge key. Pick $s_k, \alpha, \beta, \delta, \rho \overset{\$}{\leftarrow} \mathbb{Z}_p^*$, and compute ${g_1}' \leftarrow {g_1}^{\beta(\rho P(\theta) - s_k P_-(\theta))}$, $u \leftarrow {g_1}^{\beta\delta P(\theta)}$, $v \leftarrow {g_1}^{\beta P_-(\theta)}$, $h_2 \leftarrow {g_2}^{\alpha}$, $Y \leftarrow ({g_2}^{\theta} {g_2}^{-r_k})^{\alpha}$. Thanks to $\alpha, \beta, \delta, \rho$, the public key is correctly distributed. Finally $\mathcal{A}$ is given $(p, \mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T, e, {g_1}', u, v, g_2, h_2, X, Y)$.

Simulating $\textsc{Sign}(m_n)$. We keep a counter n for the queries. If $n = k$, we set $A_k \leftarrow {g_1}^{\beta P_-(\theta)(\rho + \delta m_k)}$ and if $n \neq k$, $A_n \leftarrow {g_1}^{\beta P_{n-}(\theta)(\rho\theta + \delta\theta m_n + s_n - s_k)}$.

Forgery. $\mathcal{A}$ eventually outputs a valid and non-trivial forgery $(m_*, (A_*, r_*, s_*))$ such that

$$A_* = ({g_1}' u^{m_*} v^{s_*})^{\frac{1}{\gamma + r_*}} = {g_1}^{\beta P_-(\theta)(\rho\theta + \delta\theta m_* + s_* - s_k)\frac{1}{\theta + r_* - r_k}}$$

Solving the SDH challenge. We distinguish two cases.

I. $r_* \notin \{r_1, \dots, r_q\}$. Let us consider $\beta P_-(\theta)(\rho\theta + \delta\theta m_* + s_* - s_k)$ as a polynomial A in θ. If we carry out the euclidean division of A by $(\theta + r_* - r_k)$, we get Q and R such that $A(\theta) = (\theta + r_* - r_k)Q(\theta) + R(\theta)$. As $(\theta + r_* - r_k)$ is a first degree polynomial $X - (r_k - r_*)$, we know that $R(\theta) = A(r_k - r_*)$, so we are able to compute: $C := R(\theta) = A(r_k - r_*) = \left[\prod_{n=1, n\neq k}^{q} (r_n - r_*)\right] ((\rho + \delta m_*)(r_k - r_*) + s_* - s_k)$. As $A_* = {g_1}^{\frac{C}{\theta + r_* - r_k} + Q(\theta)}$, we are able to compute C and ${g_1}^{Q(\theta)}$ from the SDH challenge. We again have two cases.

1. $(s_* - s_k) \neq (\rho + \delta m_*)(r_* - r_k)$. In this case, $C \neq 0$. Compute ${g_1}^{\frac{1}{\theta + r_* - r_k}} = (A_* {g_1}^{-Q(\theta)})^{\frac{1}{C}}$, set $c = r_* - r_k$, and $(c, {g_1}^{1/(\theta+c)})$ as a solution to the SDH challenge.
2. $(s_* - s_k) = (\rho + \delta m_*)(r_* - r_k)$. $\mathcal{B}$ returns $\perp$.

II. $r_* \in \{r_1, \dots, r_q\}$. We distinguish two cases.

1. $r_* = r_k$. In this case, we have $({A_*}^{s_k} {A_k}^{-s_*})^{\frac{1}{s_k - s_*}} = {g_1}^{\beta(P(\theta)(\rho(s_k - s_*) + \delta(m_* s_k - m_k s_*))\frac{1}{s_k - s_*} - P_-(\theta)s_k)\frac{1}{\theta}}$. Since P vanishes in 0, but not P_-, then when we divide $\beta(P(\theta)(\rho(s_k - s_*) + \delta(m_* s_k - m_k s_*))\frac{1}{s_k - s_*} - P_-(\theta)s_k)$ by θ and we get R and Q such that $C := R(0) = -\beta s_k \left[\prod_{n=1, n\neq k}^{q} (r_n - r_*)\right]$ and $({A_*}^{s_k} {A_k}^{-s_*}))^{\frac{1}{s_k - s_*}} = {g_1}^{\frac{C}{\theta} + Q(\theta)}$ with $C \neq 0$. $\mathcal{B}$ computes

   $$ {g_1}^{1/\theta} = (({A_*}^{s_k} {A_k}^{-s_*})^{\frac{1}{s_k - s_*}} {g_1}^{-Q(\theta)})^{1/C}, $$

   sets $c = 0$ and obtains a SDH solution $(0, {g_1}^{1/\theta})$.
2. $r_* \neq r_k$. $\mathcal{B}$ returns $\perp$.

We now have to estimate the probability for $\mathcal{B}$ to fail. No information is available about k from $\mathcal{A}$'s point of view. So if $r_* \in \{r_1, \dots, r_q\}$, $r_* = r_k$ with probability $1/q$, and if $r_* \notin \{r_1, \dots, r_q\}$, $(s_* - s_k) = (\rho + \delta m_*)(r_* - r_k)$ with probability no more than $1/q$. Therefore if $\mathcal{A}$ outputs a valid forgery with probability ϵ', then the probability ϵ that $\mathcal{B}$ solve the challenge is ϵ'/q. We need $O(q)$ multiplications in order to compute ${g_1}^{P(\theta)}$, and we need to carry out an division of a polynomial of degree q by a polynomial of degree 1, which we can do in $O(q^2)$. □

Theorem 7 *If the $\mathcal{S}^{(1)}$ scheme is (q_S, t, ϵ)-EUF-CMA-secure, then the $\mathcal{S}^{(\ell)}$ scheme is $(q_S, t - O(\ell), \epsilon/2\ell)$-EUF-CMA-secure.*

Proof. Let $\mathcal{A}$ an adversary that (q_S, t, ϵ)-breaks the $\mathcal{S}^{(\ell)}$ scheme. $\mathcal{A}$ outputs valid and non-trivial $(m_{*1}, \dots, m_{*\ell})$, (S_*, r_*, s_*) with probability greater than ϵ. Let $u_1, \dots, u_\ell$ be parameters for the $\mathcal{S}^{(\ell)}$ scheme. We distinguish two types of adversary.

Type I. With probability greater than $\epsilon/2$, for all i and $(m_{i1}, \dots, m_{i\ell})$ queried during the attack, we have $\prod_{j=1}^{\ell} u_j^{m_{*j}} \neq \prod_{j=1}^{\ell} u_j^{m_{ij}}$.

Type II. With probability greater than $\epsilon/2$, $\exists i \in [1, q_S]$ s.t. $(m_{i1}, \dots, m_{i\ell})$ has been queried during the attack and we have $\prod_{j=1}^{\ell} u_j^{m_{*j}} = \prod_{j=1}^{\ell} u_j^{m_{ij}}$.

From type I adversaries we construct an EUF-CMA adversary $\mathcal{B}$ against the $\mathcal{S}^{(1)}$ scheme. From type II adversary, we construct an adversary against the DL problem.

Type I. Given $(p, \mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T, e, g_1, u, v, g_2, h_2)$, together with public key (X, Y) for the $\mathcal{S}^{(1)}$ scheme, for all $j \in \{1, \dots, \ell\}$, $\mathcal{B}$ picks $\alpha_j \overset{\$}{\leftarrow} \mathbb{Z}_p^*$ and computes $u_j := u^{\alpha_j}$. $\mathcal{A}$ is next given the parameters. When $\mathcal{A}$ asks for a signature on $(m_{i1}, \dots, m_{i\ell})$, $\mathcal{B}$ asks for a signature on $m_i = \sum_{j=1}^{\ell} \alpha_j m_{ij}$. The response (A_i, r_i, s_i) is transferred to $\mathcal{A}$. Adversary $\mathcal{A}$ eventually outputs $(m_{*1}, \dots, m_{*\ell}), (A_*, r_*, s_*)$. $\mathcal{B}$ sets $m_* = \sum_{j=1}^{\ell} \alpha_j m_{*j}$ and returns $(m_*, (A_*, r_*, s_*))$.

Type II. Given a discrete logarithm challenge $(g, h) \in \mathbb{G}_1$, we pick $k \overset{\$}{\leftarrow} [1, l]$, $v, h' \overset{\$}{\leftarrow} \mathbb{G}_1$, $g_2 \overset{\$}{\leftarrow} \mathbb{G}_2$ and $\gamma, \alpha_1, \dots, \alpha_\ell \overset{\$}{\leftarrow} \mathbb{Z}_p^*$. Set $(u_1, \dots, u_{k-1}, u_k, u_{k+1}, \dots, u_\ell) := (g^{\alpha_1}, \dots, g^{\alpha_{k-1}}, h^{\alpha_k}, g^{\alpha_{k+1}}, \dots, g^{\alpha_\ell})$. $\mathcal{A}$ is given the parameters and the public key $(X, Y) = (g^{\gamma}, {g_2}^{\gamma})$. For each request $(m_{i1}, \dots, m_{i\ell})$, a $\mathcal{S}^{(\ell)}$ signature is produced with γ. When $\mathcal{A}$ returns a forgery $(m_{*1}, \dots, m_{*\ell}, (A_*, r_*, s_*))$, with probability greater than $\epsilon/2$, $\exists i \in [1, q_S]$ s.t. $\prod_{j=1}^{\ell} u_j^{m_{*j}} = \prod_{j=1}^{\ell} u_j^{m_{ij}}$, and

$$\log_g h = \frac{\sum_{j=1, j\neq k}^{\ell} \alpha_j (m_{*j} - m_{ij})}{\alpha_k (m_{*k} - m_{ik})}$$

if $\alpha_k(m_{ik} - m_{*k}) \neq 0$. It remains to bound the probability that $\alpha_k(m_{ik} - m_{*k}) = 0$. Since $(m_{i1}, \dots, m_{i\ell}) \neq (m_{*1}, \dots, m_{*\ell})$, and since k has been uniformly picked and is independent of the adversary's view, then the probability that $m_{ik} \neq m_{*k}$ is at least $1/\ell$.

In front of an adversary $\mathcal{A}$, we pick a Type I or II adversary with probability 1/2, so the global security loss is $\epsilon/2\ell$. □

A.4 Security of the Main Scheme

Theorem 8 (Traceability) *The $\mathcal{SAN}$ scheme is traceable in the sense of definition 3.*

(Sketch of proof). Let $\mathcal{A}$ be a successful attacker against the traceability property. If the tracing is not able to conclude, then a valid signature has been produced during the attack on a new y^*, which contradict the unforgeability of the underlying signature scheme. More precisely, we extract the underlying y^* via the forking lemma and, since we know the extraction key, we extract a signature on y^* and win the unforgeability game against the $\mathcal{S}^{(2)}$ scheme. □

Theorem 9 (Immutability) *The $\mathcal{SAN}$ scheme is immutable in the sense of definition 3.*

Proof. The immutability property refers to the fact that an adversary with oracle access to the signer algorithm cannot create a valid message-signature for a sanitizer with key y^* s.t. for all previous queries i we either have $y^* \neq y_i$ or $m^* \notin \{\textsc{mod}(m) \mid \textsc{mod} \text{ with } \textsc{adm}_i(\textsc{mod}) = 1\}$. Moreover, recall that this condition takes into account the case where a non-allowed modification has been done on a modifiable block.

Let $\mathcal{A}$ be an efficient successful attacker against immutability. We construct a forger $\mathcal{B}$ against the $\mathcal{S}^{(2)}$ signature scheme. $\mathcal{B}$ receives as input a public key $\mathsf{pk}_s := (X, Y)$ of the signature scheme. $\mathcal{B}$ generates parameters for our scheme, including $h_1 := {g_1}^{\alpha}$ and $h_2 := {g_2}^{\alpha}$, knowing α. When $\mathcal{A}$ ask $\mathcal{B}$ for a signature on m for sanitizer y with respect to admissible modifications $\textsc{adm}$, $\mathcal{B}$ ask her own oracle for $|M|+1$ signatures on $(y, \mathcal{H}(m_{\text{FIX}} \| \textsc{adm}))$ and $(y, \mathcal{H}(m_{\text{FIX}} \| \textsc{adm} \| n \| m_n))$ for all $n \in M$ and generates a sanitizable signature thanks to these individual signatures. $\mathcal{A}$ eventually outputs y^*, m^* and $\sigma^* = (B^*_{0:1}, B^*_{0:2}, r^*, s^*, \{(B^*_{n:1}, B^*_{n:2}, r^*_n, s^*_n)\}_{n \in M^*}, R^*, s^*_y, \textsc{adm}^*)$. Let m^*_{FIX} be the fixed part of message m^* *w.r.t* $\textsc{adm}^*$, *i.e.* $m^*_{\text{FIX}} := \textsc{fix}_{\text{ADM}^*}(m^*)$. If $\mathcal{A}$ succeeds, then

$y^* \neq y_i$ for all query i.
Then our adversary $\mathcal{B}$ computes $S^* := B_{0:2}{B_{0:1}}^{\alpha}$ and returns (S^*, r^*, s^*) as forgery on $(y^*, m^*_{\text{FIX}} \| \textsc{adm}^*)$.

Or $(y^*, m^*_{\text{FIX}} \| \textsc{adm}^*)$ has not been submitted by $\mathcal{B}$ to the signature oracle before.
We then have $\textsc{fix}_{\text{ADM}^*}(m^*) \neq \textsc{fix}_{\text{ADM}_i}(m_i)$, *i.e.* $m^*_{\text{FIX}} \neq m_{\text{FIX},i}$. Then $\mathcal{B}$ computes $S^* := B_{0:2}{B_{0:1}}^{\alpha}$ and returns (S^*, r^*, s^*) as forgery on $(y^*, m^*_{\text{FIX}} \| \textsc{adm}^*)$.

Or $\textsc{fix}_{\text{ADM}^*}(m^*) = \textsc{fix}_{\text{ADM}_i}(m_i)$, but since $m^* \notin \{\textsc{mod}(m) \mid \textsc{mod} \text{ s.t. } \textsc{adm}_i(\textsc{mod}) = 1\}$, there exists $n \in M^*$ such that m_n is not an admissible value.
Set $G^* := B_{0:2}{B_{0:1}}^{\alpha}$. (G^*, r^*_n, s^*_n) is a valid and non trivial forgery on $(y^*, m^*_{\text{FIX}} \| \textsc{adm}^* \| n \| m_n)$.

In either case we have a forgery on $\mathcal{S}^{(2)}$, which concludes the proof. □

Theorem 10 (Unforgeability) *The $\mathcal{SAN}$ scheme is unforgeable in the sense of definition 3.*

(Sketch of proof). Our scheme is unforgeable since both (1) the signature scheme $\mathcal{S}^{(2)}$ for the fixed part and (2) the signature of the modifiable part are unforgeable (which is an adaptation of the proof of the Schnorr signature). □

Theorem 11 (Unlinkability) *The $\mathcal{SAN}$ scheme is (trace-restricted-)unlinkable in the sense of definition 3.*

Proof. Given access to a signing oracle, a sanitizing oracle and a trace oracle, an adversary cannot essentially distinguish left or right outputs of a *left-or-right* type LoR oracle better than a flipping coin. In addition, the adversary has access to an oracle for generating sanitizers' keys. The LoR oracle is initialized with a random bit b, takes two pairs $(j_0, m_0, \textsc{mod}_0, \sigma_0, \textsc{adm}_0)$ and $(j_1, m_1, \textsc{mod}_1, \sigma_1, \textsc{adm}_1)$ where $\textsc{adm}_0 = \textsc{adm}_1$, both signatures are valid, the modified message $\textsc{mod}_0(m_0) = \textsc{mod}_1(m_1)$ coincide, and both sanitizers j_0 and j_1 are able to sanitize the respective messages. It outputs $\textsc{Sanitize}(\mathsf{sk}_{\mathsf{san}, j_b}, m_b, \sigma_b, \textsc{mod}_b)$.

First note that for each query to LoR, we have $\textsc{adm} := \textsc{adm}_0 = \textsc{adm}_1$ and since admissible modifications do not change the fixed part of messages and the modified messages of a query coincide, it also holds $m_{\text{FIX},0} = m_{\text{FIX},1}$. So the r, s signature component for the fixed parts must be the same for both messages. If an adversary submit distinct r, s, both signatures $S_{\text{FIX},0} \neq S_{\text{FIX},1}$ contains in the $B_{0:1}, B_{0:2}$ part are valid, and it contradicts the strong unforgeability of $\mathcal{S}$: since an honest signer produces those signatures deterministically, it has output at most one signature.

Hence, such a query with identical messages and distinct signatures for the fixed part must thus contain a forgery, and the signature part r, s for the sanitized messages must be identical. By a similar argument, the r_n, s_n part for each restricted modification $n \in M$ must be identical.

Since a new signature for the same message $m'_0 = m'_1$ is computed from scratch, both signatures are identically distributed in both cases. It follows that the probability of predicting b is $\frac{1}{2}$, unless the adversary creates a forgery against $\mathcal{S}^{(2)}$. We notice that the $B_{n:1}, B_{n:2}$ parts, including the case $n = 0$, are randomized. They are then identically distributed for each sanitization. For a successful distinguisher against the unlinkability game, we are able to construct a distinguisher against the semantic security of those ElGamal encryptions, namely the DDH assumption in $\mathbb{G}_1$. □

Remark 4. Unfortunately, our scheme is not *strongly* unlinkable. The reason is that the signer is supposed to be honest in the generation of the randomness. If not, he could use different randomness to link several sanitizing processes.

B. A MORE EFFICIENT SCHEME

We now give a slightly more efficient variant of our scheme. Instead of producing a signature per attributes, the issuer certifies them together in a single signature. We use the $\mathcal{S}^{(\ell)}$ signature scheme, for $\ell = |M|+2$. For $(S, r, s) \in \mathbb{G}_1 \times \mathbb{Z}_p \times \mathbb{Z}_p$, we have :

$$S := \left(g_1 {u_0}^{h_0} {u_1}^{y_j} \prod_{n=2}^{|M|+1} {u_n}^{h_{\phi(n-1)}} v^s \right)^{\frac{1}{\gamma + r}}$$

where ϕ maps the modifiable blocs to their position in the message. A signature σ on a message m is

$$\sigma := \Big((B_1, B_2, r, s), (\vec{R}, \vec{c}, \vec{s_\rho}), \textsc{adm} \Big)$$

The signature of knowledge becomes :

$$(\vec{R}, \vec{c}, \vec{s_\rho}) := \text{SoK}\left\{ \Big\langle \rho, \{h_n\}_{\substack{n \in M \\ n \notin K}} \Big\rangle \ : \ e(g_1, g_2)^{\rho} = e(g_1, X) \vee \right.$$

$$\left. e(u_1, g_2)^{\rho} = \frac{e(B_2, {g_2}^r X) e(B_1, {h_2}^r Y)}{e(g_1 {u_0}^{h_{\text{FIX}}} \prod_{n=2}^{|M|+1} {u_n}^{h_{\phi(n-1)}} v^s, g_2)} \right\} (\mathsf{M})$$

on the message $\mathsf{M} := B_1 \| B_2 \| r \| s \| m \| \textsc{adm} \| X \| Y$.