Comment on "Attribute-Based Signatures for Supporting Anonymous Certification" by N. Kaaniche and M. Laurent (ESORICS 2016)

Damien Vergnaud

École normale supérieure
DI/ENS (ENS, CNRS, INRIA, PSL)
45 rue d'Ulm – 75230 Paris Cedex 05 France

Abstract

Anonymous credential systems enable users to authenticate themselves in a privacy-preserving manner. At the conference ESORICS 2016, Kaaniche and Laurent presented an anonymous certification scheme based on a new attribute based signature. In this note, we provide several attacks on their scheme.

1 Introduction

Anonymous certification is a kind of cryptographic protocols in which users obtain certificates from organizations and subsequently demonstrate their possession in such a way that transactions carried out by the same user cannot be linked. They were introduced by Chaum [Cha85] and extensively studied in the last three decades (*e.g.* [BCC+09, BCKL08, CL01, CL03, CL04, CG08, ILV11]).

Recently, Kaaniche and Laurent [KL16a] presented a novel anonymous certification primitive based on attribute based signatures. The objective of their primitive (called $\mathcal{HABS}$, for Homomorphic Attribute Based Signatures) is to "enable a user to anonymously authenticate with a verifier, while providing only required information for the service provider, with respect to its presentation policy". The authors presented a concrete realization of this primitive and claimed that its security relies on standard assumptions in the random oracle model. In this note, we prove that their scheme does not achieve the claimed security properties.

2 Description of Kaaniche-Laurent Model and Proposal

2.1 Syntactic Definition

The privacy-preserving attribute based signature $\mathcal{HABS}$ introduced by Kaaniche and Laurent [KL16a] is defined by the following four algorithms, one two-party protocol and a "procedure":

$$\mathcal{HABS} = (\textsc{Setup}, \textsc{KeyGen}, \textsc{Obtain}, \textsc{Issue}, \textsc{Show} \leftrightarrow \textsc{Verify}, \textsc{Inspec})$$

- Setup: this (probabilistic polynomial time) algorithm takes as input a security parameter $\xi \in \mathbb{N}$ and outputs the public parameters denoted params and a public/private key pair for the tracing authority (or *inspector*) denoted $(\mathsf{pk}_{\mathsf{ins}}, \mathsf{sk}_{\mathsf{ins}})$.
- KeyGen: this (probabilistic polynomial time) algorithm takes as input the public parameters and a bit b. It outputs a public/private key pair for a (regular) user (if $b = 0$) denoted $(\mathsf{pk}_u, \mathsf{sk}_u)$ or a public/private key pair for the issuer (if $b = 1$) denoted $(\mathsf{pk}_{\mathsf{is}}, \mathsf{sk}_{\mathsf{is}})$.
- Issue: this (probabilistic polynomial time) algorithm[1] params, the public key of a user pk_u, a set of attributes $\mathcal{S}$, the public key of the tracing authority $\mathsf{pk}_{\mathsf{ins}}$ and the private key of the issuer $\mathsf{sk}_{\mathsf{is}}$. It outputs a *credential* C over the set of attributes $\mathcal{S}$ for pk_u.

[1] In [KL16a], the "issuance procedure" of a credential is alluded as a two-party protocol Obtain $\leftrightarrow$ Issue but in their formal model as well as in their concrete proposal, the two algorithms are actually independent and the protocol is non-interactive.

- $\textsc{Obtain}$: this (polynomial time) algorithm takes as input the public parameters params, the private key of a user sk_u, the public key of the tracing authority $\mathsf{pk}_{\mathsf{ins}}$ and the public key of the issuer $\mathsf{pk}_{\mathsf{is}}$ and a putative credential C. It outputs a bit $b \in \{0, 1\}$.
- $\textsc{Show} \leftrightarrow \textsc{Verify}$: this (probabilistic polynomial time) two-party protocol is defined by two (probabilistic polynomial time) algorithms:
  - $\textsc{Show}$: this algorithm takes as input the public parameters params, the secret key of a user sk_u, a set of attributes $\mathcal{S}$, the public key of the tracing authority $\mathsf{pk}_{\mathsf{ins}}$, the public key of the issuer $\mathsf{pk}_{\mathsf{is}}$, a *credential* C over the set of attributes $\mathcal{S}$ for pk_u and a *signing predicate* Υ.
  - $\textsc{Verify}$: this algorithm takes as input the public parameters params, the public key of the tracing authority $\mathsf{pk}_{\mathsf{ins}}$, the public key of the issuer $\mathsf{pk}_{\mathsf{is}}$ and the signing predicate Υ.

  At the end of the execution, $\textsc{Verify}$ outputs a bit $b \in \{0, 1\}$ (for success of failure of the verification).
- $\textsc{Inspec}$: this *inspection procedure* is performed by the tracing authority using the private key $\mathsf{sk}_{\mathsf{ins}}$. It actually consists in two (polynomial-time) algorithms `trace` and `judge`. This procedure is not relevant for our attacks and is therefore not described.

These algorithms should satisfy the following *correctness* properties[2]:

- **correctness of the issuance procedure:** for all params and $(\mathsf{pk}_{\mathsf{ins}}, \mathsf{sk}_{\mathsf{ins}})$ output by $\textsc{Setup}$, for all $(\mathsf{pk}_u, \mathsf{sk}_u)$ output by $\textsc{KeyGen}(\mathsf{params}, 0)$, for all $(\mathsf{pk}_{\mathsf{is}}, \mathsf{sk}_{\mathsf{is}})$ output by $\textsc{KeyGen}(\mathsf{params}, 1)$, for all set of attributes $\mathcal{S}$, for all C output by $\textsc{Issue}(\mathsf{params}, \mathsf{pk}_u, \mathcal{S}, \mathsf{pk}_{\mathsf{ins}}, \mathsf{sk}_{\mathsf{is}})$,

$$\textsc{Obtain}(\mathsf{params}, \mathsf{sk}_u, \mathsf{pk}_{\mathsf{ins}}, \mathsf{pk}_{\mathsf{is}}, C) = 1.$$

- **correctness of the presentation procedure:** for all params and $(\mathsf{pk}_{\mathsf{ins}}, \mathsf{sk}_{\mathsf{ins}})$ output by $\textsc{Setup}$, for all $(\mathsf{pk}_u, \mathsf{sk}_u)$ output by $\textsc{KeyGen}(\mathsf{params}, 0)$, for all $(\mathsf{pk}_{\mathsf{is}}, \mathsf{sk}_{\mathsf{is}})$ output by $\textsc{KeyGen}(\mathsf{params}, 1)$, for all set of attributes $\mathcal{S}$, for all C output by $\textsc{Issue}(\mathsf{params}, \mathsf{pk}_u, \mathcal{S}, \mathsf{pk}_{\mathsf{ins}}, \mathsf{sk}_{\mathsf{is}})$, for all signing predicate Υ satisfied by $\mathcal{S}$,

$$\textsc{Show}(\mathsf{params}, \mathsf{sk}_u, \mathcal{S}, \mathsf{pk}_{\mathsf{ins}}, \mathsf{pk}_{\mathsf{is}}, C, \Upsilon) \leftrightarrow \textsc{Verify}(\mathsf{params}, \mathsf{pk}_u, \mathsf{pk}_{\mathsf{ins}}, \mathsf{pk}_{\mathsf{is}}, \Upsilon) = (\perp, 1).$$

  (*i.e.* at the end of the interactive protocol, with an honestly generated presentation token, the algorithm $\textsc{Verify}$ outputs 1).

Remark 1 *It is worth mentioning that in [KL16a], the definition of the* $\textsc{Show} \leftrightarrow \textsc{Verify}$ *protocol is more specific. In their framework, the user always identify himself to the verifier by signing a (random) message of the verifier's choosing. This approach is indeed the one used in the construction proposed by Kaaniche and Laurent but we choose to state a more generic definition for the* $\mathcal{HABS}$ *primitive.*

2.2 Security Properties.

Kaaniche and Laurent [KL16a] formalized the security properties of their primitive using seven security properties divided into three categories:

- **unforgeability:** a user not owning an appropriate legitimate credential is not able to generate a valid presentation token. This requirement is captured using three different security games (see [KL16a] for details):
  - $\mathsf{MC\text{-}Game}$: the adversary is allowed to makes (polynomially many) queries to the $\textsc{Issue}$ algorithm for different set of attributes for a fixed user public key and issuer (unknown) private key and should not be able to output a credential C^* for a *new* set of attributes which is accepted by the $\textsc{Obtain}$ algorithm.
  - $\mathsf{MU\text{-}Game}$: the adversary is given as input a user public key[3], a set of attributes $\mathcal{S}$ and a credential C over $\mathcal{S}$ for pk_u. It can run (polynomially many) presentation procedure as a verifier for any signing predicate satisfied by (a subset of) $\mathcal{S}$ but should not be able to be accepted later by an honest verifier in a presentation procedure for this credential.

[2]In addition, to these properties, Kaaniche and Laurent [KL16a] added naturally the correctness of the inspection procedure.

[3]In [KL16a], it is written that it is also given the corresponding private key, but this makes the security property impossible to achieve

- Col-Game: the adversary is given two public/private key pairs for users $(\mathsf{pk}_{u_1}, \mathsf{sk}_{u_1})$ and $(\mathsf{pk}_{u_2}, \mathsf{sk}_{u_2})$, two set of attributes $\mathcal{S}_1$ and $\mathcal{S}_2$ disjoint and non-empty, two credentials C_1 over $\mathcal{S}_1$ for pk_{u_1} and C_2 over $\mathcal{S}_2$ for pk_{u_2}. It should not be able to be accepted by an honest verifier in a presentation procedure for this signing predicate (for a key pair $(\mathsf{pk}_{u_b}, \mathsf{sk}_{u_b})$ for $b \in \{1,2\}$ and some signing predicate Υ such that $\mathcal{S}_b$ does not satisfy Υ).

- **anonymity:** the user must remain anonymous among a set of users during the presentation procedure to the verifier. This requirement is captured using three different security games (again, see [KL16a] for details). In all indistinguishability games, the adversary is given two public/private key pairs for users $(\mathsf{pk}_{u_1}, \mathsf{sk}_{u_1})$ and $(\mathsf{pk}_{u_2}, \mathsf{sk}_{u_2})$, a set of attributes $\mathcal{S}$. It can run (polynomially many) presentation procedure as a verifier for any signing predicate satisfied by (a subset of) $\mathcal{S}$ for two fixed credentials C_1 over $\mathcal{S}$ for pk_{u_1} and C_2 over $\mathcal{S}$ for pk_{u_2}. The adversary should not be able to guess with probability significantly better than one half:
  - PP-Game: which key pair $(\mathsf{pk}_{u_b}, \mathsf{sk}_{u_b})$ for $b \in \{1,2\}$, was used in a presentation procedure in which it acts as a verifier (for a chosen subset of attributes and a fixed signing predicate)
  - MS-Game: whether the same key pair $(\mathsf{pk}_{u_b}, \mathsf{sk}_{u_b})$ was used in two presentation procedures in which it acts as a verifier (for a subset of attributes and a signing predicate of its choice)
  - IS-Game: which key pair $(\mathsf{pk}_{u_b}, \mathsf{sk}_{u_b})$ and credential C_b for $b \in \{1,2\}$, was used in a presentation procedure in which it acts as a verifier (for a fixed subset of attributes and a fixed signing predicate). In this game, the adversary is given C_1 and C_2 at the end of the presentation procedure.

  The PP-Game and IS-Game formalized notions of anonymity while the MS-Game formalized a notion of unlinkability.

- **anonymity removal:** the user should not be able to be accepted in a presentation procedure without being traceable by the tracing authority. This requirement is captured by a single security game IA-Game in which a dishonest user tries to make the transcript of presentation procedure trace to another honest user. This property does not hold trivially if the scheme does not achieve the unforgeability property as captured by the games MU-Game or Col-Game.

3 Description of Kaaniche-Laurent Scheme.

In this section, we describe the concrete proposal given by Kaaniche and Laurent in [KL16a]. It relies on linear secret sharing schemes and monotone span programs on one hand and bilinear maps on the other hand. We refer to [KL16a] (and references therein) for details on these primitives.

- Setup: this algorithm generates an asymmetric bilinear group environment $(p, \mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T, e)$ where $\mathbb{G}_1$, $\mathbb{G}_2$ and $\mathbb{G}_T$ are groups of prime order p and $e : \mathbb{G}_1 \times \mathbb{G}_2 \leftarrow \mathbb{G}_T$ is an asymmetric bilinear map (or Type-3 pairing [GPS08]). The group laws in the three groups are denoted multiplicatively and the bilinear map e is non-degenerate (*i.e.*, given g_1 and g_2 generators of $\mathbb{G}_1$ and $\mathbb{G}_2$, $e(g_1, g_2)$ is a generator of $\mathbb{G}_T$).
  The algorithm selects uniformly at random $g_1 \in \mathbb{G}_1$, $\{u_i\}_{i\in[1,U]} \in \mathbb{G}_1^U$ (where U denotes the maximum number of attributes supported by the scheme) and $g_2 \in \mathbb{G}_2$. It picks uniformly at random $\alpha \in \mathbb{Z}_p$ and sets $h_1 = g_1^{\alpha} \in \mathbb{G}_1$ and $h_2 = g_2^{-\alpha} \in \mathbb{G}_2$. It outputs the description of $\mathcal{H}$ a *cryptographic hash function*[4]. The global parameters of the scheme are

$$\mathsf{params} = (p, \mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T, e, g_1, \{u_i\}_{i\in[1,U]}, g_2, \mathcal{H}).$$

  The public and private keys of the tracing authority are $\mathsf{pk}_{\mathsf{ins}} = (h_1, h_2)$ and $\mathsf{sk}_{\mathsf{ins}} = \alpha$.

- KeyGen: this algorithm outputs a pair of private and public keys for each participating entity (regular user or issuer). For a regular user (*i.e.*, with input bit $b = 0$), it outputs $(\mathsf{sk}_u, \mathsf{pk}_u)$ where sk_u is picked uniformly at randomly in $\mathbb{Z}_p$ and pk_u is the couple $(X_u, Y_u) = (g_1^{\mathsf{sk}_u}, e(g_1, g_2)^{\mathsf{sk}_u})$. For the issuer (*i.e.*, with input bit $b = 1$), it outputs $(\mathsf{sk}_{\mathsf{is}}, \mathsf{pk}_{\mathsf{is}})$ where $\mathsf{sk}_{\mathsf{is}} = (s_{\mathsf{is}}, x_{\mathsf{is}})$ with s_{is} picked uniformly at randomly in $\mathbb{Z}_p$ and $x_{\mathsf{is}} = g_1^{x_{\mathsf{is}}}$ and $\mathsf{pk}_{\mathsf{is}}$ is the couple $(X_{\mathsf{is}}, Y_{\mathsf{is}}) = (e(g_1, g_2)^{s_{\mathsf{is}}}, h_2^{s_{\mathsf{is}}})$

[4]In [KL16a], Kaaniche and Laurent do not give the security properties required for this hash function.

- Issue: this algorithm takes as input a pre-shared set of attributes $\mathcal{S} \subseteq \{1, \ldots, U\}$. The set of attributes $\mathcal{S}$ is defined as follows: $\mathcal{S} = \{a_1, a_2, \ldots, a_N\}$, where $N \leq U$ is the number of attributes. It also takes as input the public key of the user pk_u and the private key of the issuer $\mathsf{sk}_{\mathsf{is}}$. It picks uniformly at randomly r in $\mathbb{Z}_p$ and outputs a *credential* C defined as

$$C = \big(C_1, C_2, C_3, \{C_{4,i}\}_{i \in [1,N]}\big) = \Big(x_{\mathsf{is}} \cdot [X_u^{\mathcal{H}(\mathcal{S})^{-1}}] \cdot h_1^r, g_1^{-r}, g_2^r, \{u_{a_i}^r\}_{i \in [1,N]}\Big)$$

  where $\mathcal{H}(\mathcal{S}) = \mathcal{H}(a_1)\mathcal{H}(a_2) \ldots \mathcal{H}(a_N)$.

- Obtain: this algorithm takes as input the credential C, the public key of the user pk_u, the public key of the issuer $\mathsf{pk}_{\mathsf{is}}$ and the set of attributes $\mathcal{S}$. The algorithm outputs 1 if the following equation holds:

$$e(C_1, g_2) \overset{?}{=} X_{\mathsf{is}} \cdot e(X_u^{\mathcal{H}(\mathcal{S})^{-1}}, g_2) \cdot e(h_1, C_3) \tag{1}$$

  and 0 otherwise.

- Show $\leftrightarrow$ Verify: this two-party protocol works as follows:
  - Verify: this algorithm takes as input the public parameters params, the public key of the tracing authority $\mathsf{pk}_{\mathsf{ins}}$, the public key of the issuer $\mathsf{pk}_{\mathsf{is}}$ and a signing predicate Υ. In a first step, It picks uniformly at random a message $m \in \mathbb{Z}_p$ and sends m to the user.
  - Show: this algorithm takes as input the public parameters params, the secret key of a user sk_u, a set of attributes $\mathcal{S}$, the public key of the tracing authority $\mathsf{pk}_{\mathsf{ins}}$, the public key of the issuer $\mathsf{pk}_{\mathsf{is}}$, a *credential* C over the set of attributes $\mathcal{S}$ for pk_u, the *signing predicate* Υ and the message m sent by Verify:
    1. The user first randomize his credential C in the following way: it selects uniformly at random an integer $r' \in \mathbb{Z}_p$ and sets

$$\begin{cases} C'_1 &= C_1 \cdot h_1^{r'} = x_{\mathsf{is}} \cdot [X_u^{\mathcal{H}(\mathcal{S})^{-1}}] \cdot h_1^{r+r'} \\ C'_2 &= C_2 \cdot g_1^{-r'} = g_1^{-(r+r')} \\ C'_3 &= C_3 \cdot g_2^{r'} = g_2^{r+r'} \\ C'_{4,i} &= C_{4,i}^r \cdot u_i^{r'} = u_i^{r+r'} \end{cases}$$

       and the re-randomized credential is

$$C' = \big(C'_1, C'_2, C'_3, \{C'_{4,i}\}_{i \in [1,N]}\big) = \Big(x_{\mathsf{is}} \cdot [X_u^{\mathcal{H}(\mathcal{S})^{-1}}] \cdot h_1^{r+r'}, g_1^{-(r+r')}, g_2^{r+r'}, \{u_{a_i}^{r+r'}\}_{i \in [1,N]}\Big)$$

    2. The user then generates a vector $\Omega \in \mathbb{G}_2^{\ell}$ that depends on C'_3, its set of attributes $\mathcal{S}$ and the predicate Υ and a group element $B \in \mathbb{G}_1$ that depends on $\{C'_{4,i}\}_{i \in [1,N]}$, its set of attributes $\mathcal{S}$ and the predicate Υ (but the details are not relevant for our attacks).
    3. The user selects a random $r_m \in \mathbb{Z}_p$ and computes the couple

$$(\sigma_1, \sigma_2) = (C'_1 \cdot B \cdot g_1^{r_m \cdot m}, g_1^{r_m})$$

    4. Finally, the user computes an accumulator on non-revealed attributes as:

$$A = g_2^{\mathsf{sk}_u \cdot \mathcal{H}(\mathcal{S}_{\mathcal{H}})^{-1} r_m^{-1}}$$

       and outputs the *presentation token* $\Sigma = (\Omega, \sigma_1, \sigma_2, C'_1, C'_2, A, \mathcal{S}_R)$.
  - Verify: In a second step, given Σ the algorithm computes $A_R = \sigma_2^{\mathcal{H}(\mathcal{S}_{\mathcal{H}})^{-1}}$. It picks uniformly at random $k-1$ integers $(\mu_2, \ldots, \mu_k)$ and generates from these integers and the access structure corresponding to the predicate Υ, ℓ integers $\tau_i \in \mathbb{Z}_p$ for $i \in \{1, \ldots, \ell\}$. It accepts the presentation token as valid (*i.e.*, it outputs 1) if and only if the following equation holds:

$$e(\sigma_1, g_2) \overset{?}{=} X_{\mathsf{is}} e(A_R, A) e(C'_2, h_2) \prod_{i=1}^{\ell} e(u_{\rho(i)} h_1^{\tau_i}, \omega_i) e(\sigma_2, g_2^m) \tag{2}$$

    It is worth noting that this equation is satisfied for any choice of the integers $(\mu_2, \ldots, \mu_k)$.

- Inspec : this algorithm is performed by the tracing authority using the secret key $\mathsf{sk}_{\mathsf{ins}}$ to decrypt the Elgamal ciphertext (C_1, C_2). The details are not relevant to our attacks and are therefore omitted.

4 Cryptanalysis of Kaaniche-Laurent scheme

Kaaniche and Laurent stated three security claims in their paper [KL16a, Theorems 2, 3, 4, p. 295] and mentioned that the detailed security proofs are given in the unpublished note [KL16b]. The theorems assert that the proposal described above achieves the seven security properties outlined above. In this section, we present polynomial-time attacks which show that these security claims do not hold.

4.1 Description of the forgery attacks.

- $\mathsf{MC\text{-}Game}$: an attack in this game is easily obtained by remarking that in the OBTAIN algorithm, the validity of some elements in a credential are not checked. An adversary can simply query a credential for an arbitrary set of attributes $\mathcal{S} = \{a_1, a_2, \dots, a_N\}$, where $N \leq U$ and a public key $\mathsf{pk}_u = (X_u, Y_u)$. It obtains a credential C defined as

$$C = \big(C_1, C_2, C_3, \{C_{4,i}\}_{i\in[1,N]}\big) = \Big(x_{\mathsf{is}} \cdot [X_u^{\mathcal{H}(\mathcal{S})^{-1}}] \cdot h_1^r, g_1^{-r}, g_2^r, \{u_{a_i}^r\}_{i\in[1,N]}\Big)$$

where $\mathcal{H}(\mathcal{S}) = \mathcal{H}(a_1)\mathcal{H}(a_2)\dots\mathcal{H}(a_N)$. For any set of attributes $\mathcal{S}' = \{a_1, a_2, \dots, a_M\}$, it can simply sets

$$C' = \big(C'_1, C'_2, C'_3, \{C'_{4,i}\}_{i\in[1,M]}\big)$$

where

$$\begin{array}{rcl} C'_1 &=& C_1 \cdot X_u^{-\mathcal{H}(\mathcal{S})^{-1}} \cdot X_u^{\mathcal{H}(\mathcal{S}')^{-1}} = x_{\mathsf{is}} \cdot [X_u^{\mathcal{H}(\mathcal{S}')^{-1}}] \\ C'_2 &=& C_2 \\ C'_3 &=& C_3 \end{array}$$

and the values $\{C'_{4,i}\}_{i\in[1,M]}$ are picked arbitrarily in $\mathbb{G}_1$. The forged credential C' clearly satisfies (1) for the set of attributes $\mathcal{S}'$ since the original credential C does for the set of attributes $\mathcal{S}$ (since the validity of the elements $\{C'_{4,i}\}_{i\in[1,M]}$ is not checked).

- $\mathsf{MU\text{-}Game}$: an adversary eavesdropping the communication of a presentation protocol for a signing predicate Υ and a public key $\mathsf{pk}_u = (X_u, Y_u)$ is able to impersonate the corresponding user in any following session for the same predicate Υ. Indeed, such an adversary can store the challenge message m sent by the verifier and the couple

$$(\sigma_1, \sigma_2) = (C'_1 \cdot B \cdot g_1^{r_m \cdot m}, g_1^{r_m})$$

sent in the presentation token $\Sigma = (\Omega, \sigma_1, \sigma_2, C'_1, C'_2, A, \mathcal{S}_R)$ by the user. In a next session, the verifier will send a message $m' \neq m$ and the adversary can simply compute

$$\sigma'_1 = \sigma_1 \cdot \sigma_2^{m'-m} = \sigma C'_1 \cdot B \cdot g_1^{r_m \cdot m'}$$

and send the presentation token $\Sigma = (\Omega, \sigma'_1, \sigma_2, C'_1, C'_2, A, \mathcal{S}_R)$ which is accepted by the verifier.

- $\mathsf{Col\text{-}Game}$: an adversary can easily generate a valid credential for any user on a set of attributes $\mathcal{S}$ if it is given a credential for one user for $\mathcal{S}$. Let us assume that is knows a credential for an arbitrary set of attributes $\mathcal{S}_1 = \{a_1, a_2, \dots, a_N\}$, where $N \leq U$ for a given public/private key pair $(\mathsf{pk}_{u_1}, \mathsf{sk}_{u_1})$:

$$C = \big(C_1, C_2, C_3, \{C_{4,i}\}_{i\in[1,N]}\big) = \Big(x_{\mathsf{is}} \cdot [X_{u_1}^{\mathcal{H}(\mathcal{S})^{-1}}] \cdot h_1^r, g_1^{-r}, g_2^r, \{u_{a_i}^r\}_{i\in[1,N]}\Big)$$

where $\mathcal{H}(\mathcal{S}) = \mathcal{H}(a_1)\mathcal{H}(a_2)\dots\mathcal{H}(a_N)$. For any public/private key pair $(\mathsf{pk}_{u_2}, \mathsf{sk}_{u_2})$ different from $(\mathsf{pk}_{u_1}, \mathsf{sk}_{u_1})$, the adversary can sets

$$C' = \big(C'_1, C_2, C_3, \{C_{4,i}\}_{i\in[1,N]}\big)$$

where

$$C'_1 = C_1 \cdot X_{u_1}^{-\mathcal{H}(\mathcal{S})^{-1}} \cdot X_{u_2}^{\mathcal{H}(\mathcal{S}')^{-1}} = x_{\mathsf{is}} \cdot [X_{u_2}^{\mathcal{H}(\mathcal{S}')^{-1}}]$$

As in the attack in the $\mathsf{MC\text{-}Game}$ game, one can easily verify that C' satisfies (1) for the set of attributes $\mathcal{S}$ for the key pair $(\mathsf{pk}_{u_2}, \mathsf{sk}_{u_2})$. In the $\mathsf{Col\text{-}Game}$ game, an adversary can simply query one credential C_1 over $\mathcal{S}_1$ for $(\mathsf{pk}_{u_1}, \mathsf{sk}_{u_1})$ and one credential over a set $\mathcal{S}_2$ for $(\mathsf{pk}_{u_1}, \mathsf{sk}_{u_1})$ for which that there exists a signing predicate Υ satisfied by $\mathcal{S}_1$ but not by $\mathcal{S}_2$. Using the previous attack, it generates a credential C^* over $\mathcal{S}_1$ for $(\mathsf{pk}_{u_2}, \mathsf{sk}_{u_2})$ which is accepted in any presentation procedure for $\mathcal{S}_1$.

Remark 2 *As mentioned above, since the scheme proposed by Kaaniche and Laurent does not achieve the security formalized by the security games MU-Game and Col-Game, it trivially does not achieve the IA-Game which formalizes that a dishonest user should not be able to make the transcript of presentation procedure trace to another honest user (since this is possible even without knowing any secret keys).*

4.2 Description of the anonymity attacks.

- $\mathsf{PP\text{-}Game}$ and $\mathsf{IS\text{-}Game}$: using the bilinear map e and given two putative public keys, an adversary can associate with certainty a challenge presentation token with its corresponding public key. Given two user public keys $\mathsf{pk}_{u_1} = (X_{u_1}, Y_{u_1})$ and $\mathsf{pk}_{u_2} = (X_{u_2}, Y_{u_2})$ and a valid presentation token $\Sigma = (\Omega, \sigma'_1, \sigma_2, C'_1, C'_2, A, \mathcal{S}_R)$ associated with pk_{u_b} for some $b \in \{1,2\}$, we have $A = g_2^{\mathsf{sk}_{u_b} \cdot \mathcal{H}(\mathcal{S}_H)^{-1} r_m^{-1}}$ for some unknown value $r_m \in \mathbb{Z}_p$ with $\sigma_2 = g_1^{r_m}$ and some known set of attributes $\mathcal{S} = \mathcal{S}_H \cup \mathcal{S}_R$. An adversary can simply compute $A^{\mathcal{H}(\mathcal{S}_H)} = g_2^{\mathsf{sk}_{u_b} r_m^{-1}}$ and output the unique $b \in \{1,2\}$ for which the equality

$$e(\sigma_2, A^{\mathcal{H}(\mathcal{S}_H)}) = e(g_1^{r_m}, g_2^{\mathsf{sk}_{u_b} r_m^{-1}}) = Y_{u_b}$$

holds. The adversary does not need to know the original credential used to generate presentation token. The attack is thus valid in the two games $\mathsf{PP\text{-}Game}$ and $\mathsf{IS\text{-}Game}$ and the proposed scheme is not anonymous.

- $\mathsf{MS\text{-}Game}$: using the previous attack, an adversary can check whether two presentation token were generated using the same user public/private key pair $(\mathsf{pk}_u, \mathsf{sk}_u)$. Actually, an attack can be mounted even without knowing this public key. Given two presentations token $\Sigma^{(1)} = (\Omega^{(1)}, \sigma_1^{(1)}, \sigma_2^{(1)}, C_1'^{(1)}, C_2'^{(1)}, A^{(1)}, \mathcal{S}_R^{(1)})$ and $\Sigma^{(2)} = (\Omega^{(2)}, \sigma_1^{(2)}, \sigma_2^{(2)}, C_1'^{(2)}, C_2'^{(2)}, A^{(2)}, \mathcal{S}_R^{(2)})$, an adversary can compute the two group elements $T_1 = C_1'^{(2)}/C_1'^{(1)}$ and $T_2 := C_2'^{(2)}/C_2'^{(1)}$ in $\mathbb{G}_1$ and evaluate two bilinear maps $e(T_1, g_2)$ and $e(g_1^{-1}, T_2)$ in $\mathbb{G}_T$. If $\Sigma^{(1)}$ and $\Sigma^{(2)}$ were generated using the same credential obtained through the ISSUE algorithm, then the two latter values are equals. Indeed, in this case, we have

$$C_1'^{(1)} = g_1^{-r-r'_1}, C_2'^{(1)} = g_2^{r+r'_1}, C_1'^{(2)} = g_1^{-r-r'_2}, C_2'^{(2)} = g_2^{r+r'_2},$$

for some integers r'_1 and r'_2 in $\mathbb{Z}_p$. We thus have $T_1 = g_1^{-(r'_2 - r'_1)}$ and $T_2 = g_2^{r'_2 - r'_1}$ and

$$e(T_1, g_2) = e(g_1^{-(r'_2-r'_1)}, g_2) = e(g_1^{-1}, g_2^{r'_2-r'_1}) = (e(g_1^{-1}, T_2).$$

This equality holds only with negligible probability if $\Sigma^{(1)}$ and $\Sigma^{(2)}$ were generated using two different credentials and the presentation tokens are therefore linkable.

5 Conclusion

In this note, we showed that the anonymous certification primitive proposed by Kaaniche and Laurent at the conference ESORICS 2016 does not achieve the claimed security properties.

Acknowledgments. This work was done while the author was attending the ESORICS 2016. The author is grateful to Maryline Laurent for discussions relating to the work reported here as well as for comments on a preliminary version. The author is supported in part by the French ANR ALAMBIC project (ANR-16-CE39-0006).

References

[BCC+09] M. Belenkiy, J. Camenisch, M. Chase, M. Kohlweiss, A. Lysyanskaya, and H. Shacham. Randomizable proofs and delegatable anonymous credentials. In *Advances in Cryptology – CRYPTO 2009*, *Lecture Notes in Computer Science* 5677, pages 108–125, Santa Barbara, CA, USA, August 16–20, 2009. Springer, Berlin, Germany.

[BCKL08] M. Belenkiy, M. Chase, M. Kohlweiss, and A. Lysyanskaya. P-signatures and noninteractive anonymous credentials. In *TCC 2008: 5th Theory of Cryptography Conference*, *Lecture Notes in Computer Science* 4948, pages 356–374, San Francisco, CA, USA, March 19–21, 2008. Springer, Berlin, Germany.

[CG08] J. Camenisch and T. Groß. Efficient attributes for anonymous credentials. In *ACM CCS 08: 15th Conference on Computer and Communications Security*, pages 345–356, Alexandria, Virginia, USA, October 27–31, 2008. ACM Press.

[Cha85] D. Chaum. Security without identification: Transaction systems to make big brother obsolete. *Commun. ACM*, 28(10):1030–1044, 1985.

[CL01] J. Camenisch and A. Lysyanskaya. An efficient system for non-transferable anonymous credentials with optional anonymity revocation. In *Advances in Cryptology – EUROCRYPT 2001*, *Lecture Notes in Computer Science* 2045, pages 93–118, Innsbruck, Austria, May 6–10, 2001. Springer, Berlin, Germany.

[CL03] J. Camenisch and A. Lysyanskaya. A signature scheme with efficient protocols. In *SCN 02: 3rd International Conference on Security in Communication Networks*, *Lecture Notes in Computer Science* 2576, pages 268–289, Amalfi, Italy, September 12–13, 2003. Springer, Berlin, Germany.

[CL04] J. Camenisch and A. Lysyanskaya. Signature schemes and anonymous credentials from bilinear maps. In *Advances in Cryptology – CRYPTO 2004*, *Lecture Notes in Computer Science* 3152, pages 56–72, Santa Barbara, CA, USA, August 15–19, 2004. Springer, Berlin, Germany.

[GPS08] S. D. Galbraith, K. G. Paterson, and N. P. Smart. Pairings for cryptographers. *Discrete Applied Mathematics*, 156(16):3113–3121, 2008.

[ILV11] M. Izabachène, B. Libert, and D. Vergnaud. Block-wise P-signatures and non-interactive anonymous credentials with efficient attributes. In *13th IMA International Conference on Cryptography and Coding*, *Lecture Notes in Computer Science* 7089, pages 431–450, Oxford, UK, December 12–15, 2011. Springer, Berlin, Germany.

[KL16a] N. Kaaniche and M. Laurent. Attribute-based signatures for supporting anonymous certification. In *Computer Security - ESORICS 2016*, *Lecture Notes in Computer Science* 9878, pages 279–300. Springer, 2016.

[KL16b] N. Kaaniche and M. Laurent. Security analysis of $\mathcal{HABS}$. Unpublished note. 7 pages. `http://www-public.tem-tsp.eu/~laurenm/ABS-AC/securityanalysis.pdf`, 2016.