# The self-blindable U-Prove scheme from FC'14 is forgeable (short paper)

Eric Verheul[1], Sietse Ringers[2], and Jaap-Henk Hoepman[1]

[1] Radboud University, Nijmegen, The Netherlands
{jhh,e.verheul}@cs.ru.nl
[2] Johann Bernoulli Institute for Mathematics and Computer Science, University of Groningen, The Netherlands
s.ringers@rug.nl

**Abstract.** Recently an unlinkable version of the U-Prove attribute-based credential scheme was proposed at Financial Crypto '14 [9]. Unfortunately, the new scheme is forgeable: if sufficiently many users work together then they can construct new credentials, containing any set of attributes of their choice, without any involvement of the issuer. In this note we show how they can achieve this and we point out the error in the unforgeability proof.

## 1 Introduction

Attribute-based credential schemes [1,3] provide a very secure and privacy-friendly form of identity management. In these schemes, users are granted by an issuer a credential that contains several attributes (generally elements of $\mathbb{Z}_q := \mathbb{Z}/q\mathbb{Z}$ for some number $q$), and when the user shows his credential to a verifier using a ShowCredential protocol, he can choose to reveal some of these while keeping the other ones hidden from the verifier. Some of these schemes offer anonymity in the form of *multi-show unlinkability*: that is, when a verifier runs the ShowCredential protocol twice and both times the same attributes with the same values were disclosed to it, then it cannot tell whether it was shown one credential twice, or two different credentials that happened to disclose the same attributes.

A well-known and very efficient attribute-based credential scheme is U-Prove [6,10]. However, this scheme offers no multi-show unlinkability. In an attempt to fix this, L. Hanzlik and K. Kluczniak proposed in [9], and presented at Financial Crypto 2014, a new scheme that is based on U-Prove but uses a different signature scheme. This signature scheme is based on the self-blindable construction by Verheul [11], and allows a credential to be blinded; i.e., modified into a new valid credential over the same attributes. Although [9] does contain an argument for the unforgeability of their scheme, we show here that this argument contains an error, and that the proposed construction is forgeable, in the sense that if sufficiently many users collude then they can construct new credentials containing arbitrary attributes of their choice, without involvement of the issuer.

## 2 The credential scheme

Hanzlik and Kluczniak [9] present their blindable U-Prove scheme as an extension of the original U-Prove scheme, in the following sense: a self-blindable signature (based on [11]) is added to a U-Prove credential. When showing a credential, the user can then choose to either show his credential using the original linkable U-Prove ShowCredential protocol, or using a new protocol that uses the new self-blindable signature and should offer unlinkability. Since we are concerned only with the forgeability of the self-blindable construction, our description of the credential scheme will omit details that are relevant only to the original construction.

The setup is as follows. $q$ is a prime number of length $k$, and $e\colon G_1 \times G_2 \to G_T$ is a bilinear pairing of Type 2 (see [8], [5, Ch. I, X]), where $q$ is the order of $G_1$, $G_2$ and $G_T$. The issuer's public key is

$$(q, e, g_0, \ldots, g_n, p, p', p_0, p_1),$$

where

- $g_0, \ldots, g_n$ are random generators of $G_1$,
- $p$ and $p'$ are random generators of $G_2$,
- $p_0 = (p')^z$,
- $p_1 = p^f$.

The tuple $(f, z) \in \mathbb{Z}_q^2$ is the issuer's secret key.

A credential consists of the tuple

$$((x_1, \ldots, x_n), (h, h_2, h_3, h_4, \alpha, b_1, b_2))$$

where

- $x_1, \ldots, x_n \in \mathbb{Z}_q$ are the attributes,
- $\alpha, b_0, b_1 \in \mathbb{Z}_q$, chosen by the user during issuing of the credential,
- $h = (g_0 g_1^{x_1} \cdots g_n^{x_n})^\alpha$,
- $h_2 = h^f$,
- $h_3 = h^{b_1} h_2^{b_2}$,
- $h_4 = h_3^z = (h^{b_1} h_2^{b_2})^z$.

The validity of the credential can be checked by

$$e(h, p_1) \stackrel{?}{=} e(h_2, p) \qquad \text{and} \qquad e(h_3, p_0) \stackrel{?}{=} e(h_4, p').$$

Such a credential can be blinded into a new one as follows. Take random $k, \ell \in \mathbb{Z}_q^*$, and set $(\overline{h}, \overline{h}_2, \overline{h}_3, \overline{h}_4) = (h^k, h_2^k, h_3^{k\ell}, h_4^{k\ell})$. Then

$$((x_1, \ldots, x_n), (\overline{h}, \overline{h}_2, \overline{h}_3, \overline{h}_4, \alpha k, b_1\ell, b_2\ell))$$

is a new, valid credential over the same attributes. In [9] a ShowCredential protocol for these credential is provided, in which the credentials are blinded as above. The protocol should offer unlinkability but it is not proven that it does (and we have not checked this).

## 3 Forging new credentials

### 3.1 Constructing signatures on the elements $g_i$

We first show that if sufficiently many users work together, then for each $i$ they can compute a tuple $g_i^f, g_i^z, g_i^{fz}$, even though $f$ and $z$ are private to the issuer. Using these tuples they can easily create new valid credentials over any set of attributes of their choice. Since this will involve many credentials, we will write the elements from the credential of user $j$ with an extra subscript $j$:

$$((x_{1,j}, \ldots, x_{n,j}), (h_j, h_{2,j}, h_{3,j}, h_{4,j}, \alpha_j, b_{1,j}, b_{2,j})).$$

The element $h_j$ is of the form

$$h_j = (g_0 g_1^{x_{1,j}} \cdots g_n^{x_{n,j}})^{\alpha_j}.$$

By blinding the credential with $k = \alpha_j^{-1}, \ell = 1$ (i.e., we raise all group elements of the credential to the power $\alpha_j^{-1}$; note that these numbers are known to the users), we can remove the number $\alpha$ from our considerations, so we will henceforth simply write

$$h_j = g_0 g_1^{x_{1,j}} \cdots g_n^{x_{n,j}}.$$

Let us write $\widetilde{g}_i = g_i^f$. Then we can write $h_{3,j}$ as

$$h_{3,j} = h_j^{b_{1,j}} h_{2,j}^{b_{2,j}} = g_0^{b_{1,j}} \widetilde{g}_0^{b_{2,j}} g_1^{b_{1,j}x_{1,j}} \widetilde{g}_1^{b_{2,j}x_{1,j}} \cdots g_n^{b_{1,j}x_{n,j}} \widetilde{g}_n^{b_{2,j}x_{n,j}}.$$

Setting $x_{0,j} = 1$ and writing $y_{i,j} = b_{1,j}x_{i,j}$ and $\widetilde{y}_{i,j} = b_{2,j}x_{i,j}$, we get

$$h_{3,j} = g_0^{y_{0,j}} \widetilde{g}_0^{\widetilde{y}_{0,j}} g_1^{y_{1,j}} \widetilde{g}_1^{\widetilde{y}_{1,j}} \cdots g_n^{y_{n,j}} \widetilde{g}_n^{\widetilde{y}_{n,j}}, \tag{1}$$

where all numbers $y_{i,j}$ and $\widetilde{y}_{i,j}$ are known to the user.

We know that $h_{4,j} = h_{3,j}^z$, i.e., the discrete log of $h_{4,j}$ with respect to $h_{3,j}$ is $z$. If we raise $h_{3,j}$ to some power and we simultaneously raise $h_{4,j}$ to the same power, then the resulting two elements will still have $z$ as discrete log. The same holds if we multiply two elements $h_{3,j}$ and $h_{3,j'}$ together. In the remainder of this section we will take a number of powers and products of the elements $h_{3,j}$; whenever we write such a power or product, the same power or product for $h_{4,j}$ is implied.

Observe that when raising $h_{3,1}$ to the power $1/\widetilde{y}_{n,1}$ we obtain a product of the generators $g_i$ to certain exponents, where $\widetilde{g}_n$ now has exponent 1. Thus two users 1 and 2 can work together to form the element $h_{3,1}^{1/\widetilde{y}_{n,1}} / h_{3,2}^{1/\widetilde{y}_{n,2}}$, which is of the form

$$\frac{h_{3,1}^{1/\widetilde{y}_{n,1}}}{h_{3,2}^{1/\widetilde{y}_{n,2}}} = g_0^{v_0} \widetilde{g}_0^{\widetilde{v}_0} g_1^{v_1} \widetilde{g}_1^{\widetilde{v}_1} \cdots g_n^{v_n},$$

with $v_i = y_{i,1}/y_{n,1} - y_{i,2}/y_{n,2}$, and similar for $\widetilde{v}_i$. Note that the right hand side no longer contains $\widetilde{g}_n$. If two more users do the same and obtain a similar expression, then the four users can collectively remove $g_n$ in exactly the same fashion, resulting in an expression as above containing only the elements $g_0, \widetilde{g}_0, \ldots, g_{n-1}, \widetilde{g}_{n-1}$.

Continuing in this fashion, $2^{2n+1}$ users can find an element in $G_1$ that is just $g_0$ raised to some power which is known and can easily be removed. If they apply all powers and products in parallel to the corresponding $h_{4,j}$, then they also obtain $g_0^z$. Similarly, they can obtain $\widetilde{g}_0 = g_0^f$, and $\widetilde{g}_0^z = g_0^{fz}$. In fact, they can do this for all elements $g_i, \widetilde{g}_i$, resulting finally in expressions for $g_i^f$, $g_i^z$ and $g_i^{fz}$ for all $i$. Using these elements, anyone can calculate a valid credential over any set of attributes as explained below. The amount of users that need to work together to achieve this ($2^{2n+1}$) is exponential in $n$ (the amount of attributes of the system) but *not* in the security parameter. Therefore, this can be done in polynomial time.

*Remark 1.* An alternative explanation for why this is possible is as follows. Suppose we are given $m$ valid credentials, with $h_{3,j}$ of credential $j$ given by (1). Notice that the operations we apply to the elements $h_{3,j}$ and $h_{4,j}$ above correspond exactly to taking linear combinations of the $h_{3,j}$ and $h_{4,j}$ (although linear combinations are usually written additively instead of multiplicatively). So if we consider the elements $g_i, \widetilde{g}_i$ occurring in $h_{3,j}$ as unknowns, then we can interpret equation (1) as one equation in $2n+2$ unknowns. Thus if we have $m := 2n+2$ credentials, then we obtain $2n+2$ equations in as many unknowns.

Using linear algebra over the field $\mathbb{Z}_q = \mathrm{GF}(q)$, then, we can solve this system of linear equations to the $g_i, \widetilde{g}_i, g_i^{fz} = \widetilde{g}_i^z$, as long as the square matrix of the coefficients,

$$M := \begin{pmatrix} y_{0,1} & \cdots & y_{0,2n+2} \\ \widetilde{y}_{0,1} & \cdots & \widetilde{y}_{0,2n+2} \\ \vdots & \ddots & \vdots \\ y_{n,1} & \cdots & y_{n,2n+2} \\ \widetilde{y}_{n,1} & \cdots & \widetilde{y}_{n,2n+2} \end{pmatrix}$$

is invertible (i.e., its determinant $\det M$ is unequal to 0). Since the numbers $y_{i,j}, \widetilde{y}_{i,j}$ are under our control in a chosen-message attack, this should be easy to achieve. If we write $m_{i,j}$ for the $j$-th entry of the $i$-th row of the inverse $M^{-1}$ of $M$, we obtain

$$g_i = \prod_{j=1}^{2n+2} h_{3,j}^{m_{2i+1,j}}, \qquad \widetilde{g}_i = \prod_{j=1}^{2n+2} h_{3,j}^{m_{2i+2,j}},$$
$$g_i^z = \prod_{j=1}^{2n+2} h_{4,j}^{m_{2i+1,j}}, \qquad \widetilde{g}_i^z = \prod_{j=1}^{2n+2} h_{4,j}^{m_{2i+2,j}}.$$

This also shows that the scheme is already completely forgeable (in the sense that new credentials with arbitrary attributes can be computed) with just $2n+2$ collaborating users, instead of $2^{2n+1}$.

### 3.2 Constructing a forged credential

Using the elements $g_i, g_i^f, g_i^z, g_i^{fz}$ constructed above, a new credential with attributes $x_1, \ldots, x_n$ may be constructed as follows. Choose $b_1, b_2 \in_R \mathbb{Z}_q$ randomly, and set

$$\begin{aligned}
h &= g_0 g_1^{x_1} \cdots g_n^{x_n}, \\
h_2 &= g_0^f (g_1^f)^{x_1} \cdots (g_n^f)^{x_n}, \\
h_3 &= g_0^{b_1} (g_0^f)^{b_2} g_1^{b_1 x_1} (g_1^f)^{b_2 x_1} \cdots g_n^{b_1 x_n} (g_n^f)^{b_2 x_n}, \\
h_4 &= (g_0^z)^{b_1} (g_0^{fz})^{b_2} (g_1^z)^{b_1 x_1} (g_1^{fz})^{b_2 x_1} \cdots (g_n^z)^{b_1 x_n} (g_n^{fz})^{b_2 x_n}.
\end{aligned}$$

Then

$$h_2 = h^f, \qquad h_3 = h^{b_1} h_2^{b_2}, \qquad h_4 = (h^{b_1} h_2^{b_2})^z$$

as required.

## 4 The problem in the unforgeability argument

An argument for unforgeability is given in [9] in section 4, "Security Analysis". The argument is based on the appendix from [11], in which it is argued that credentials of the form

$$h,\ h_2 = h^f,\ h_4 = (h^{b_1} h_2^{b_2})^z \tag{2}$$

are unforgeable. Here, as above, $f$ and $z$ are the issuer's secret key, and the numbers $b_1, b_2$ are part of the credential (i.e., known to the user). However, the difference with Verheul's system is that there $h$ is randomly chosen from $G_1$, and in particular, no participant of the system knows the discrete log of $h$ with respect to any other element from $G_1$, or any DL-representation of $h$ (i.e., an expression of $h$ in terms of powers of $g_0, \ldots, g_n$, such as (3)). By contrast, in Hanzlik and Kluczniak's U-Prove scheme the user knows numbers $\alpha, x_1, \ldots, x_n$ such that

$$h = (g_0 g_1^{x_1} \cdots g_n^{x_n})^\alpha. \tag{3}$$

where the elements $g_0, \ldots, g_n$ are the same for all users. In this case, the argument from [11] does not apply, so that no argument can be based on it.

In addition, we wish to point out that the argument from the appendix in [11] was meant as a sketch, and in particular, there is the following subtlety. It is

argued in the appendix that if an adversary $\mathcal{A}$ manages to forge credentials of the form (2), i.e.

$$(h, h_2, h_4, b_1, b_2) = \mathcal{A}\Big((h_j, h_{2,j}, h_{4,j}, b_{1,j}, b_{2,j})_{j=1,\ldots,m}\Big)$$

where the output $(h, h_2, h_4, b_1, b_2)$ is valid (i.e., satisfying (2)), then either there must exist a $j$ and numbers $k, \ell \in \mathbb{Z}_q$ such that

$$(h, h_2, h_4, b_1, b_2) = (h_j^k, h_{2,j}^k, h_{4,j}^{k\ell}, b_{1,j}\ell, b_{2,j}\ell)$$

or the adversary $\mathcal{A}$ can be used to solve discrete logarithms in $G_1$. However, the argument mentions certain "transformation factors" which are numbers like $k, \ell$ from $\mathbb{Z}_q$, and the algorithm sketched by [11] that uses the adversary $\mathcal{A}$ to compute discrete logarithms would need to know these numbers in order to be able to work. However, it is not clear how to obtain these transformation factors from the adversary $\mathcal{A}$, or even if $\mathcal{A}$ is aware of them. We believe, however, that they can be extracted from the adversary by an extension of the Known Exponent Assumption (see [7], where this assumption was introduced, and for example [2] and [4]).

## References


1. Alpár, G., Hoepman, J., Siljee, J.: The identity crisis. security, privacy and usability issues in identity management. CoRR abs/1101.0427 (2011), http://arxiv.org/abs/1101.0427
2. Bellare, M., Palacio, A.: The knowledge-of-exponent assumptions and 3-round zero-knowledge protocols. In: Franklin, M. (ed.) Advances in Cryptology – CRYPTO 2004, Lecture Notes in Computer Science, vol. 3152, pp. 273–289. Springer Berlin Heidelberg (2004)
3. Bichsel, P., Camenisch, J., Dubovitskaya, M., Enderlein, R.R., Krenn, S., Krontiris, I., Lehmann, A., Neven, G., Nielsen, J.D., Paquin, C., Preiss, F.S., Rannenberg, K., Sabouri, A., Stausholm, M.: D2.2 architecture for attribute-based credential technologies. Technical report, final version, ABC4Trust (2014), https://abc4trust.eu/download/Deliverable_D2.2.pdf
4. Bitansky, N., Canetti, R., Chiesa, A., Goldwasser, S., Lin, H., Rubinstein, A., Tromer, E.: The hunting of the SNARK. IACR Cryptology ePrint Archive 2014 (2014), https://eprint.iacr.org/2014/580
5. Blake, I.F., Seroussi, G., Smart, N.P. (eds.): Advances in Elliptic Curve Cryptography. Cambridge University Press (2005), cambridge Books Online
6. Brands, S.: Rethinking Public Key Infrastructures and Digital Certificates: Building in Privacy. MIT Press (2000)
7. Damgård, I.: Towards practical public key systems secure against chosen ciphertext attacks. In: Feigenbaum, J. (ed.) Advances in Cryptology - CRYPTO '91, 11th Annual International Cryptology Conference, Santa Barbara, California, USA, August 11-15, 1991, Proceedings. Lecture Notes in Computer Science, vol. 576, pp. 445–456. Springer (1991)
8. Galbraith, S.D., Paterson, K.G., Smart, N.P.: Pairings for cryptographers. Discrete Applied Mathematics 156(16), 3113–3121 (2008)

9. Hanzlik, L., Kluczniak, K.: A short paper on how to improve U-Prove using self-blindable certificates. In: Christin, N., Safavi-Naini, R. (eds.) Financial Cryptography and Data Security. pp. 273–282. Lecture Notes in Computer Science, Springer Berlin Heidelberg (2014)
10. Paquin, C., Zaverucha, G.: U-prove cryptographic specification v1.1 (revision 3) (December 2013), `http://research.microsoft.com/apps/pubs/default.aspx?id=166969`, released under the Open Specification Promise
11. Verheul, E.R.: Self-blindable credential certificates from the weil pairing. In: Boyd, C. (ed.) Advances in Cryptology - ASIACRYPT. Lecture Notes in Computer Science, vol. 2248, pp. 533–551. Springer (2001)