Attack on U-Prove Revocation Scheme from FC'13 - Passing Verification by Revoked Users *

Lucjan Hanzlik, Kamil Kluczniak, and Mirosław Kutyłowski

Faculty of Fundamental Problems of Technology, Wrocław University of Technology
{firstname.secondname}@pwr.wroc.pl

Abstract. We analyse security of the scheme proposed in the paper "Accumulators and U-Prove Revocation" from the Financial Cryptography 2013 proceedings. Its authors propose an extension for the U-Prove, the credential system developed by Microsoft. This extension allows to revoke tokens (containers for credentials) using a new cryptographic accumulator scheme. We show that, under certain conditions, there exists a weakness that allows a user to pass the verification while using a revoked U-Prove token. It follows that the proposed solution fails to fulfil the primary goal of revocation schemes.
Recently, a closely related system has been published by Microsoft Research in "U-Prove Designated-Verifier Accumulator Revocation Extension, Draf 1 Revision". Our attack does not work for this scheme, but the draft lacks formal justification and we cannot exclude problems of this kind.

Key words: anonymous credential, attribute, U-Prove, revocation, attack

1 Introduction

Anonymous credentials. In this paper we discuss U-Prove [1] - one of the most prominent implementations of anonymous credentials. Today, anonymous credentials is one of the hottest research topics, as they aim to realize the idea of systems where the privacy is protected "by design". Recent developments are driven in particular by increasing (legal) pressure from European Union to deploy such systems.

Anonymous credential is a cryptographic system in which a person receives an authentication token from the trust provider. The token confirms some *attributes* of the owner, e.g. her or his rights to login to some systems. A holder of such a token, say Alice, can use it for authentication. For any subset of attributes A of the attributes contained in the token she can execute an authentication protocol with Bob so that:

- she proves that she holds an authentication token with all attributes from A,

however, at the same time

* This paper was partially supported by grant S30028/I-18 from the Institute of Mathematics and Computer Science of the Wroclaw University of Technology. Part of the work was done by the first author within project 2012-9/4 of the Ventures programme of Foundation for Polish Science, cofinanced from European Union, Regional Development Fund.

– Bob cannot conclude anything about the attributes not contained in A.

Note that "the attributes not contained in A" may include among others identity data such as the first name, the family name, and the personal ID number.

Based on the presented token (and the value of attributes) the verifier can make appropriate decisions. A good example of an attribute is the legal age enabling to engage in civil contracts. Note that this attribute should not be the exact physical age but a logical value *true* or *false* indicating whether a given person reached the age necessary to enter civil contracts.

There are many models of anonymous credentials and subtle differences between them. The functionality discussed in this paper is possibility to revoke an authentication token by the token issuer so that it cannot be used anymore by the token holder.

U-Prove. It is an anonymous credentials system based on the work of Stefan Brands on e-cash [2] and PKI [3]. The original idea evolved into an anonymous credential system. It was implemented by Microsoft under the name U-Prove. For a description of U-Prove and other material we refer the reader to the web page [1] maintained by Microsoft.

One of the major disadvantages of this system is that the standard U-Prove specification does not allow to revoke credentials. If a U-Prove token gets stolen, then the thief can use it freely – the unique security features of anonymous credentials perfectly protect the thief. If a user receives a U-Prove token for some attributes, then he can use it indefinitely, in particular after loosing some of the attributes confirmed by the token. This is a major disadvantage limiting the application scope, since many attributes are temporal: e.g. status of a student, employee of a company, customer of a company, inhabitant of a local community (the attribute enabling to participate actively in democracy on a local level) and so on. A partial solution of this problem is to use U-Prove tokens with a limited validity period.

U-Prove extension. During Financial Cryptography'2013, a scheme expanding U-Prove by revocation procedures has been presented [4]. The solution is fairly complicated in design – this is witnessed for example by the number of variables used in algorithm description. It is an extension of the original scheme which is a very nice property from the business point of view as it does not require rewriting already developed U-Prove products.

In September 2013 Microsoft Research published a technical description [5] closely related to the paper [4]. There some differences between both U-Prove extensions, however the main idea seems to be the same. Maybe the most important visible difference is removing the pairing function (this definitely makes implementation much easier, since we are more flexible about the choice of the underlying algebraic structures). On the other hand, it delegates verification of the presented token back to the system, which is a serious disadvantage from the usability point of view. Our attack shows that there is another important difference.

Neither [4] nor [5] contains a complete security proof. Even the information on the underlying concepts is very sketchy; the form of [5] is closer to an industrial standard specification than to an academic paper. Therefore our strategy is not to find a flaw in

the security proof (as the details are missing), but rather to find a possibility to attack given a concrete scheme specification. Perhaps it even helps to mount an attack as we do not follow the steps of the system designers and do not share the same intuitions. We also do not attempt to indicate the necessary corrections - as it is the responsibility of the designers of a product with strong commercial connotations.

Our Contribution. We show that the extension proposed in [4] has a weakness in the sense that a revoked person may provide a fake authentication token that passes authentication despite the fact that this person's ID is already contained in the accumulator. "Easily" means here that a simple computer program can deliver a fake token that would pass the verification. Of course, derivation of the fake token is different from the original algorithm of creating authentication token described in [4].

The attack concerns the scheme in the form described in [4]. We do not claim that this flaw cannot be corrected (it seems that there is an easy patch). However, at the same time we are far from being able to guarantee that this and similar constructions are free from other security problems.

In Sect. 2 we present chosen details describing the extension from [4]. In Sect. 3 we describe the attack against this scheme.

2 U-Prove Revocation Extension from FC'2013

Below we give a brief description of the extension of U-Prove proposed in [4] and aiming to provide revocation functionality. We describe only the details of the system and its extension which are essential for understanding the discovered weakness of the system. For a more detailed description we refer the reader to the original paper [4].

2.1 Parameters

Beside the standard U-Prove Issuer parameters

$$IP = (UID_P, desc(\mathbb{G}_q), UID_{\mathcal{H}}, (g_0, g_1, \ldots, g_n, g_t), (\mathrm{e}_1, \ldots, \mathrm{e}_n), S))$$

there are parameters related to Blacklist Authority (BA). Namely, BA holds a secret key δ and the following public parameters:

$$param = (q, \mathbb{G}_1, \mathbb{G}_2, \mathbb{G}_T, e, P_1, P_2, P_{pub}, H, K, G_1)$$

where $P_{pub} = P_2^{\delta}$, $K = H^{\delta}$, $G_1, H \in \mathbb{G}_1$, $P_2 \in \mathbb{G}_2$ and $e : \mathbb{G}_1 \times \mathbb{G}_2 \to \mathbb{G}_T$ is a pairing function.

2.2 Blacklist

Instead of a list of identifiers the Blacklist Authority maintains a (public) accumulator holding the identifiers of all revoked users. Each user receives a special attribute, which is the revocation identifier x_{ID}. In addition, a user receives a witness $w = (d, W, Q)$ that can be used to prove that his x_{ID} is not in the accumulator.

The accumulator is the number $V = P_1^{\prod_{i=1}^{k}(\delta + x_{ID_i})}$, where $x_{ID_1}, \ldots, x_{ID_k}$ are the identifiers of the revoked users. All users must update their witness w each time a new identifier is revoked. To enable updating the witness by the users themselves, the Blacklist Authority publishes a vector $t = (P_1^{\delta}, P_1^{\delta^2}, \ldots, P_1^{\delta^k})$ and the revoked identifiers $x_{ID_1}, \ldots, x_{ID_k}$.

2.3 Creating a Proof of not Being Revoked

According to the specification, the standard proof of possession of attributes is extended in the following way:
First, the following numbers are chosen at random from $\mathbb{Z}_q$:

$$x, u, t_1, t_2, t_3, r_x, r_u, r_{t_1}, r_{t_2}, r_{t_3}, r_{\beta_1}, r_{\beta_2}, r_{\beta_3}, r_d, r_{d'}.$$

Then the following numbers are computed:

$$\begin{array}{lll}
X := WH^{t_1}, & Y := QK^{t_1}, & C := G_1^x H^u \\
A := G_1^{r_x} H^u, & R := G_1^{t_1} H^{t_2}, & S := G_1^{d'} H^{t_3}, \\
T_1 := G_1^{r_{t_1}} H^{r_{t_2}}, & T_2 := G_1^{r_{\beta_1}} H^{r_{\beta_2}} R^{-r_x}, & T_3 := G_1^{r_{d'}} H^{r_{t_3}}, \\
T_4 := H^{r_{\beta_3}} S^{-r_d}, & \Gamma := X^{-rx} H^{r_{\beta_1}} K^{r_{t_1}} P_1^{-r_d}. &
\end{array}$$

The next steps are computing

$$a := \mathcal{H}(h^{w_0}(\textstyle\prod_{i \in U} g^{w_i}), \mathcal{H}(X, Y, R, S, T_1, T_2, T_3, T_4, \Gamma, param))$$

and the challenge

$$c := \text{GenerateChallenge}(IP, T, a, m, \emptyset, D; \{x_i\}_{i \in D}),$$

where GenerateChallenge and the parameters involved (apart from a) are some U-Prove parameters independent from the revocation part; the numbers w_0 and w_i, for $i \in U$, are for the standard U-Prove token and have nothing to do with revocation. Finally, c and the other parameters are used to generate the following numbers:

$$\begin{array}{lll}
\beta_1 := t_1 x_{ID}, & \beta_2 := t_2 x_{ID}, & \beta_3 := t_3 d, \\
d' := d^{-1}, & & \\
s_{t_1} := -ct_1 + r_{t_1}, & s_{t_2} := -ct_2 + r_{t_2}, & s_{t_3} := -ct_3 + r_{t_3}, \\
s_{\beta_1} := -c\beta_1 + r_{\beta_1}, & s_{\beta_2} := -c\beta_2 + r_{\beta_2}, & s_{\beta_3} := -c\beta_3 + r_{\beta_3}, \\
s_u := -cu + r_u, & s_x := -cx + r_x, & \\
s_d := -cd + r_d, & s'_d := -cd' + r_{d'}. &
\end{array}$$

They will be used by the verification procedure presented below to reconstruct all the arguments of $\mathcal{H}$ used to compute a. In fact, some of these values are related to the Schnorr signatures. Finally, for the revocation part of the proof, the following tuple is presented:

$$c, s_u, s_x, s_d, s_{d'}, s_{t_1}, s_{t_2}, s_{t_3}, s_{\beta_1}, s_{\beta_2}, s_{\beta_3}, C, X, Y, R, S$$

2.4 Verification

Let T be a U-Prove token and let the tuple $(c, s_u, s_x, s_d, s'_d, s_{t_1}, s_{t_2}, s_{t_3}, s_{\beta_1}, s_{\beta_2}, s_{\beta_3}, C, X, Y, R, S)$ be its extension part. Apart from the standard verification of T, the Verifier performs the following operations:

1. compute the following values:

$$\begin{aligned} \widetilde{T}_1 &= G_1^{s_{t_1}} H^{s_{t_2}} R^c, & \widetilde{T}_2 &= G_1^{s_{\beta_1}} H^{s_{\beta_2}} R^{-s_x}, \\ \widetilde{T}_3 &= G_1^{s_{d'}} H^{s_{t_3}} S^c, & \widetilde{T}_4 &= G_1^{-c} H^{s_{\beta_3}} S^{-s_d}, \\ \widetilde{A} &= G_1^{s_x} H^{s_u} C^c, & \widetilde{\Gamma} &= X^{-s_x} H^{s_{\beta_1}} K^{s_{t_1}} P_1^{-s_d} (V^{-1}Y)^c, \end{aligned}$$

2. verify whether $e(Y, P_2) \stackrel{?}{=} e(X, P_{pub})$,
3. verify that for

$$a := \mathcal{H}((g_0 g_t^{x_t} \prod_{i \in D} g_i^{x_i})^{-c} h^{r_o} (\prod_{i \in U} g_i^{r_i}), \mathcal{H}(\widetilde{A}, X, Y, R, S, \widetilde{T}_1, \widetilde{T}_2, \widetilde{T}_3, \widetilde{T}_2, \widetilde{\Gamma}, param)).$$

we have $c = \text{GenerateChallenge}(IP, T, a, m, \emptyset, D; \{x_i\}_{i \in D})$.

3 The Weakness in the Extention from FC'2013

In this section we show that having a valid U-Prove token T and the corresponding revocation identifier x_{ID}, the adversary can create a non-revocation proof that passes the verification procedure from Sect. 2.4 even if the identifier x_{ID} has been revoked and included in the accumulator V.

To do so, the adversary exploits a weakness in the verification procedure. Namely, it does not verify that the x_{ID} used in the non-revocation proof is the same as the attribute x_{ID} in the U-Prove token. To be more specific, there is no proof of equivalence between those x_{ID}-s. Thus, the adversary may use a valid non-revocation proof for a different token (with a different identifier) or simply use a self-created non-revocation proof. The specific construction of the accumulator and verification procedure allows everyone to create a valid proof (in which we set x_{ID} to 0).

We show how to create such a valid non-revocation proof. First, we show how to compute the parameters X, Y. Then, we show how to create the remaining parameters so that the non-revocation proof passes the verification test from Sect. 2.4.

Computing X and Y. Let us define the following polynomial:

$$f(x) = \prod_{i=1}^{k} (x + x_{ID_i}) = a_n x^n + a_{n-1} x^{n-1} + \ldots + a_1 x + a_0.$$

Then the accumulator V equals $P_1^{f(\delta)}$. Further, we define the following polynomials:

$$\begin{aligned} f'(x) &= f(x) - a_0 = a_n x^n + a_{n-1} x^{n-1} + \ldots + a_1 x \\ g(x) &= f'(x)/x = a_n x^{n-1} + a_{n-1} x^{n-2} + \ldots + a_1. \end{aligned}$$

Thus, we can compute $X := P_1^{g(\delta)}$ and $Y := P_1^{f'(\delta)}$ using the vector t and interpolation in the exponent. Note that:

$$e(Y, P_2) = e(P_1^{f'(\delta)}, P_2) = e((P_1^{f'(\delta)})^{\delta^{-1}}, P_2^{\delta}) = e(P_1^{g(\delta)}, P_{pub}) = e(X, P_{pub})$$

and that $V = YP_1^{a_0}$.

In the above procedure the attacker has to know the vector t in order to perform interpolation in the exponent for computing X and Y. However, there is an option for the extended U-Prove where t and the revoked identifiers are not published. Instead, the users could get the current value of the accumulator. In this case the attack does not work directly.

Nevertheless, the attacker can get the values from $P_1^{\delta}, P_1^{\delta^2}, \ldots, P_1^{\delta^k}$. Assume that all the revoked identifiers belong to the attacker and the identifiers are revoked one by one. As the users first become $V = P_1^{\delta + x_{ID_1}}$, the attacker can compute P_1^{δ} as $V/P_1^{x_{ID_1}}$. After the second revocation the users get $V = P_1^{(\delta+x_{ID_1})(\delta+x_{ID_2})} = P_1^{\delta^2} \cdot (P_1^{\delta})^{x_{ID_1}+x_{ID_2}} \cdot P_1^{x_{ID_1} \cdot x_{ID_2}}$, so the adversary can easily compute $P_1^{\delta^2}$. This procedure can be continued. It suffices to know the identities of the revoked users and the values of the accumulator V to perform the computations and derive the vector t.

Computing of the remaining values. In the previous subsection we have shown how to compute X and Y. Now we will show how to compute the remaining values. To do so, we perform the following steps:

1. Choose $x_1, x_3, r_x, r_u, r'_d, r_{t_1}, r_{t_2}, r_{t_3}, r_{\beta_1}, r_{\beta_2}, r_{\beta_3}$ at random.
2. Compute:

$$\begin{array}{lll} x_2 = a_0^{-1}, & S = G_1^{x_2}, & C = H^{x_3}, \\ R = H_1^{x_1}, & T_1 = G_1^{r_{t_1}} H^{r_{t_2}}, & T_2 = G_1^{r_{\beta_1}} H^{r_{\beta_2}} R^{-r_x}, \\ T_3 = G_1^{r_{d'}} H^{r_{t_3}}, & T_4 = H^{r_{\beta_3}}, & \Gamma = X^{-r_x} H^{r_{\beta_1}} K^{r_{t_1}}, \\ A = G_1^{r_x} H^{r_u}. & & \end{array}$$

3. Compute a and c according to the original specification.
4. Compute

$$\begin{array}{lll} s_d = -cx_2^{-1}, & s'_d = r_{d'} - cx_2, & \\ s_{t_1} = r_{t_1}, & s_{t_2} = r_{t_2} - cx_1, & s_{t_3} = r_{t_3}, \\ s_x = r_x, & s_u = r_u - cx_3, & \\ s_{\beta_1} = r_{\beta_1}, & s_{\beta_2} = r_{\beta_2}, & s_{\beta_3} = r_{\beta_3}. \end{array}$$

5. Return $(s_u, s_x, s_d, s'_d, s_{t_1}, s_{t_2}, s_{t_3}, s_{\beta_1}, s_{\beta_2}, s_{\beta_3}, C, X, Y, R, S)$ as the revocation part of the attribute presentation proof.

Correctness. We will show that the values computes above will pass the verification from Sect. 2.4. For this purpose we have to show that the verification procedure will

deliver the same values as used for computing a by the adversary:

$$\widetilde{T_1} = G_1^{s_{t_1}} H^{s_{t_2}} R^c = G_1^{r_{t_1}} H^{r_{t_2} - cx_1} (H^{x_1})^c = G_1^{r_{t_1}} H^{r_{t_2}} = T_1,$$

$$\widetilde{T_2} = G_1^{s_{\beta_1}} H^{s_{\beta_2}} R^{-s_x} = G_1^{r_{\beta_1}} H^{r_{\beta_2}} R^{-r_x} = T_2,$$

$$\widetilde{T_3} = G_1^{s_{d'}} H^{s_{t_3}} S^c = G_1^{r'_d - cx_2} H^{r_{t_3}} (G_1^{x_2})^c = G_1^{r'_d} H^{r_{t_3}} = T_3,$$

$$\widetilde{T_4} = G_1^{-c} H^{s_{\beta_3}} S^{-s_d} = G_1^{-c} H^{r_{\beta_3}} (G_1^{x_2})^{cx_2^{-1}} = H^{r_{\beta_3}} = T_4,$$

$$\widetilde{A} = G_1^{s_x} H^{s_u} C^c = G_1^{r_x} H^{r_u - cx_3} (H^{x_3})^c = G_1^{r_x} H^{r_u} = A,$$

$$\begin{aligned}\widetilde{\Gamma} &= X^{-s_x} H^{s_{\beta_1}} K^{s_{t_1}} P_1^{-s_d} (V^{-1}Y)^c = X^{-r_x} H^{r_{\beta_1}} K^{r_{t_1}} P_1^{cx_2^{-1}} ((Y P_1^{a_0})^{-1} Y)^c \\ &= X^{-r_x} H^{r_{\beta_1}} K^{r_{t_1}} P_1^{cx_2^{-1}} (P_1^{-a_0})^c = X^{-r_x} H^{r_{\beta_1}} K^{r_{t_1}} P_1^{cx_2^{-1}} P_1^{-cx_2^{-1}} \\ &= X^{-r_x} H^{r_{\beta_1}} K^{r_{t_1}} = \Gamma.\end{aligned}$$

Recall, that X and Y fulfil the equation $e(Y, P_2) = e(X, P_{pub})$. Thus, we will pass all steps of the verification related to the revocation extension. As we have not manipulated the creation of the U-Prove traditional token, it will accepted as well.

Remarks. The reader might ask what is the magic behind the choice of the parameters for the fake proof of the non-revoked status. Definitely, first the parameters for computation of c must be fixed (unless we aim to break the hash function). Then we have to find the other parameters from the proof that during the verification would yield the arguments used originally for computing c.

Our solution has been found by analyzing dependencies. Of course, in a secure design we have a kind of loop: an attempt to cheat leads to setting the same value in different ways to satisfy different equations. This is the basic property of constructions such as Schnorr signatures. Unfortunately, driven by pure intuition and reverse-engineering methodology ("do not try to analyze first all details of the attacked system") we have found a path to set all values in a way to fulfill all equations in a way different than designed by the authors of the extension.

References

1. Microsoft: U-Prove. Webpage of the project (retrieved 2013). Available from: `http://research.microsoft.com/en-us/projects/u-prove/`
2. Brands, S.: Untraceable off-line cash in wallets with observers (extended abstract). In Stinson, D.R., ed.: CRYPTO. Vol. 773 of LNCS, Springer (1993) 302–318
3. Brands, S.A.: Rethinking Public Key Infrastructures and Digital Certificates: Building in Privacy. 1 edn. MIT Press, Cambridge-London (2000). Available from: `http://www.credentica.com/the_mit_pressbook.html`
4. Acar, T., Chow, S.S.M., Nguyen, L.: Accumulators and U-Prove revocation. In Sadeghi, A.R., ed.: Financial Cryptography. Vol. 7859 of LNCS, Springer (2013) 189–196
5. Lan Nguyen, C.P.: U-Prove designated-verifier accumulator revocation extension. Technical Report Draft Revision 1, Microsoft Research (2013)