Attack against a Pairing Based Anonymous Authentication Protocol

Lucjan Hanzlik and Kamil Kluczniak

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

Abstract. Anonymous authentication protocols aim to provide means to anonymously prove membership in a group. Moreover, the membership should not be transferable i.e. a subgroup of members should not be able to help an outsider to gain access on behalf of a group. In this note we present two attacks on a recently published protocol of this kind (ICUIMC '11 Proceedings of the 5th International Conference on Ubiquitous Information Management and Communication, article no. 32) and thereby we show that it failed the security targets for an anonymous authentication protocol.

Keywords: anonymous authentication, attack, pairing.

1 Introduction

With the growth of technology many new kinds of information systems were created. There exist systems which can be accessed by anyone and there exist systems which require the user to authenticate before granting access. There are many ways of authentication such as login and password, asymmetric keys etc. One thing all those methods have in common is that the system always knows which user is trying to gain access. This is very convenient when the user accesses his account in the system. However, there are systems which only verify the users membership in the system e.g. access to buildings, company resources. Obviously, in such cases, some privacy issues arise when using those standard authentication methods. Some users do not want the system to know, how many times and when they accessed it. This issue inspired cryptographers to create anonymous authentication protocols. The idea is simple. We consider a set of users (called provers), a group manager and verifiers which verify the membership of users to a particular group (system), created by the group manager. So, a group manager is responsible for generating the system parameters, publishing a group verification key and issuing secret keys. Each prover, having a secret key, may use it to create a proof for a verifier such that he is certain that the prover is a member of a group which has access to the system. In addition, a verifier should not be able to tell if two attempts to access the system were made using the same keys (ie the same prover).

The notion of anonymous authentication protocols, or sometimes called group identification protocols were first proposed in [1]. Since that time, researchers have studied the problem of authenticating as member of a group. There are many approaches to

V. Geffert et al. (Eds.): SOFSEM 2014, LNCS 8327, pp. 306–314, 2014.

achieve this goals. One of it is using a group signature scheme. In adversity to anonymous authentication schemes, a signer can anonymously sign a message on behalf of the group. The problem of anonymously sign messages were first suggested by Chaum and van Heyst [2]. By now, group signatures come by a very rich literature, [3,4,5,6,7] to name only a few, and already cover many interesting problems.

To construct an authentication scheme from any group signature scheme it suffices, that a prover signs a challenge from a verifier. Another approach is to design a „clean" authentication scheme. Such schemes appeared in the literature in [8,9,10,11,12]. Unfortunately, in all of these schemes the execution time is linear dependent by the number of group members. This property make these schemes highly impractical, especially for large groups, or if the proving algorithm has to be implemented on devices with limited resources (e.g. smart cards). An interesting scheme was proposed in [13] which execution time and message size is independent of the size of the group. However, the schemes introduced [13] are in a bit different notion than group signatures or group authentication schemes, namely it introduces ad hoc anonymous identification. An ad hoc anonymous identification allows to create new groups in an ad hoc formation in the sense of ring signatures. The main contribution of [13] was introducing constant size ring and group signatures supporting efficient identity escrow capabilities. The main practical problem in [13] is that, when a new user joins a group any group member has to update their secret keys.

An anonymous authentication protocol have to fulfill some basic requirements. Such schemes must provide soundness (or unforgeability in case of group signatures), anonymity and correctness. Roughly speaking, by soundness we mean that the verifier rejects with overwhelming probability when the prover is not a legitimate group member. By anonymity we mean that no information about the users identity is revealed to the verifier besides that the prover is a group member. To be more specific, suppose we have two executions of an anonymous authentication protocol, then determining whether the executions were produced by a single prover or two distinct provers should be hard. Correctness of an anonymous authentication scheme guaranties that, if a prover is a member of the group, then a verifier accepts with overwhelming probability. Other desired features include revocation of a single user, and identity escrow. An very important property of anonymous authentication protocols is that only a group manager should be able to add new users to a group. So, we require untransferability of membership. Practically, this means that a coalition of users (possibly malicious) should not be able to create secret keys for a new group member. In some articles like [11] this is called „resilience" or „k-resilience" were k is a threshold number of corrupted users for which a scheme stays still resilient (i.e. more than k legitimate users are needed to create a new group member).

Contribution: In this paper we examine the recent work from [14]. The proposed protocol has the execution time and message size independent of the size of the group. Moreover, unlike the scheme from [13], group members do not have to update their keys when a new user joins the group. However, we encountered some design flaws which allows to perform attacks against the protocol. The first one is a collusion attack in which two cooperating group members, using their secret keys, are able to compute the group manager's secret keys. The second attack is against anonymity, where a

verifier (or an eavesdropper) can easily determine whether two transcripts were produced by the same prover or by distinct provers.

Organization of the Paper: First, we give a full description of the protocol from [14] in section 2. In section 3 we describe a collusion attack against this scheme. Then, we show how to link two protocol executions of the protocol i.e. we describe an attack against anonymity. Finally, we give some conclusions and design suggestions.

2 Protocol Description

Designations: We denote as GM the group manager who's role is to issue secret keys to group members and public group verification keys. In the original protocol description in [14] the role of GM is denoted as TTP (Trusted Third Party). However, we want to stay consistent with other work in this field. We assume also that H is a secure hash function and $||$ means concatenation of bit strings. The scheme from [14] describes also a certificate authority denoted as CA. The role of the certificate authority is to issue certificates on user public keys in order to verify the relevance of a user by the GM. Suppose that G is a point on an elliptic curve, then by $R_x(G)$ we denote the x coordinate of that point and by $R_y(G)$ the y coordinate.

Bilinear maps: Let $\mathbb{G}_1, \mathbb{G}_2$ be cyclic groups of prime order q and let $e : \mathbb{G}_1 \times \mathbb{G}_1 \to \mathbb{G}_2$ be a function with the following properties:

- (bilinearity) For all $P \in \mathbb{G}_1$, $Q \in \mathbb{G}_1$ and $a, b, c \in \mathbb{Z}_q$, we have

$$e(P^a + P^b, Q^c) = e(P, Q)^{a \cdot c} e(P, Q)^{b \cdot c}$$

  and

$$e(P^a, Q^b + Q^c) = e(P, Q)^{a \cdot b} e(P, Q)^{a \cdot c},$$

- (non-degeneracy) For all $P, Q \in \mathbb{G}_1$, $P \neq Q$ we have $e(P, Q) \neq 1$,
- (computability) e can be efficiently computed.

2.1 Setup

The GM chooses a group generator $G \in \mathbb{G}_1$. Then GM chooses two secret keys $s, k_{GM} \in \mathbb{Z}_q$ at random, and computes sG, $Y_{GM} = k_{GM}G$. The values G, sG and Y_{GM} are public, while s and k_{GM} are GM's secrets. We will denote $H_0 = H((R_x(Y_{GM})||R_y(Y_{GM})))$. The value H_0 don't has to be published since any party in the system can compute it by itself.

Each user has a private key k_U and a corresponding public key $Y_U = k_U G$. User public key Y_U are bounded to a unique identifier ID_U with a certificate from CA. We denote the certificate as $cert_{CA}(Y_U, ID_U)$.

2.2 Registration

First, a user U establishes a secure communication channel with the GM. After this step U sends his certificate $cert_{CA}(Y_U, ID_U)$ to the GM. The GM verifies the certificate and extracts the users public key and identifier. Then it performs the following steps:

1. Selects $\omega_{U,1} \in \mathbb{Z}_q$ randomly,
2. Computes $A_U := \omega_{U,1}G - Y_U = \omega_{U,1}G - k_U G$,
3. Computes $\omega_{U,2} := s\omega_{U,1}$,
4. Computes $\gamma_{U,1} := H_0\omega_{U,1} + R_x(A_U)k_{GM}$,
5. Computes $\gamma_{U,2} := H_0\omega_{U,2} + k_{GM} = H_0 s\omega_{U,1} + k_{GM}$,
6. Stores $\gamma_{U,1}$, $\gamma_{U,2}$, A_U, Y_U, $\omega_{U,1}$, $\omega_{U,2}$ and ID_U in GM's internal database,
7. Sends values A_U, $\gamma_{U,1}$ and $\gamma_{U,2}$ to user U.

User U can then verify the validity of these values by checking whether the equations:

$$\gamma_{U,1}G = H_0\omega_{U,1}G + R_x(A_U)Y_{GM} = H_0(A_U + Y_U) + R_x(A_U)Y_{GM}$$

and

$$\gamma_{U,2}G = H_0 s\omega_{U,1}G + Y_{GM} = H_0(sA_U + sY_U) + Y_{GM}$$

hold.

Note that the user is able to compute sA_U by himself. He computes $\gamma_{U,2}G = H_0 s\omega_{U,1}G + Y_{GM} = H_0 sA_U + H_0 sY_U + Y_{GM}$. Since $H_0 sY_U = k_u H_0(sG)$ and sG is public, we have that $sA_U = H_0^{-1}(\gamma_{U,2}G) - Y_{GM} - H_0 sY_U$.

2.3 Computing of Membership Proof

A verifier sends a random nonce N_s and the user U chooses five random values x, t, t_1, x_1 and $\psi \in \mathbb{Z}_q$. Then U computes the following values:

$$\begin{aligned}
t_1G, &\qquad t_2G,\\
zG, &\qquad x_2 := x \cdot x_1^{-1},\\
L := xG, &\qquad sL := x(sG),\\
L_1 := x_1G, &\qquad L_2 := x_2G,\\
sL_1 := x_1(sG), &\qquad sL_2 := x_2(sG),\\
T := \psi G, &\qquad z := t \cdot R_x(A_U),\\
B := t(A_U + T), &\qquad t_2 := t \cdot t^{-1},\\
B_1 := t_1(A_U + T), &\qquad B_2 := t_2(A_U + T),\\
stT := t\psi(sG), &\qquad sB := tsA_U + stT,\\
sB_1 := t_1 sA_U + st_1T, &\qquad sB_2 := t_2 sA_U + st_2T.
\end{aligned}$$

Let M be a message or set of instructions, which user U will send to the verifier. We define $\mathcal{D} = (M, L, sL, L_1, sL_1, L_2, sL_2, zG, t_1G, t_2G, B, B_1, B_2, sB, sB_1, sB_2)$. and the hash of $\mathcal{D}$ is computed as:

$$\begin{aligned}
H(\mathcal{D}) := H(&H(M)||H(L)||H(sL)||H(L_1)||H(sL_1)||H(L_2)\\
&||H(sL_2)||H(zG)||H(t_1G)||H(t_2G)||H(B)||H(B_1)\\
&||H(B_2)||H(sB)||H(sB_1)||H(sB_2)||H(N_s))
\end{aligned}$$

User U computes the last five values λ_1, $\lambda_{2,1}$, $\lambda_{2,2}$, $\lambda_{3,1}$, $\lambda_{3,2}$ as follows:

$$\begin{aligned}\lambda_1 &:= \gamma_{U,1}t + H_0(\psi t - k_U t + xH(\mathcal{D})),\\ \lambda_{2,1} &:= \gamma_{U,2}t_1 + H(\mathcal{D}),\\ \lambda_{2,2} &:= \gamma_{U,2}t_2 + H_0 x_2 H(\mathcal{D}),\\ \lambda_{3,1} &:= t_1(\psi - k_U) + x,\\ \lambda_{3,2} &:= t_2(\psi - k_U) + x.\end{aligned}$$

Finally U sends the values λ_1, $\lambda_{2,1}$, $\lambda_{2,2}$, $\lambda_{3,1}$, $\lambda_{3,2}$, L, sL, L_1, sL_1, L_2, sL_2, zG, t_1G, t_2G, B, B_1, B_2, sB, sB_1, sB_2 along with the message M to the verifier.

2.4 Verification of the Proof

The verifier, holding the values sent by a user U, performs the following steps:

- Computes $zY_{GM} := \lambda_1 G - H_0(B + H(\mathcal{D})L)$.
- Computes t_1Y_{GM} and t_2Y_{GM} as follows:

$$\begin{aligned}t_1Y_{GM} &:= \lambda_{2,1}G + \lambda_{3,1}sG - H_0(sB_1 + H(\mathcal{D})L_1) - sL\\ t_2Y_{GM} &:= \lambda_{2,2}G + \lambda_{3,2}sG - H_0(sB_2 + H(\mathcal{D})L_2) - sL\end{aligned}$$

- For $j = \{1, 2\}$ checks whether the discrete logarithm of t_jG equals the discrete logarithm of t_jY_{GM} with respect to G:

$$e(t_1Y_{GM}, G) \stackrel{?}{=} e(Y_{GM}, t_1G) \qquad e(t_2Y_{GM}, G) \stackrel{?}{=} e(Y_{GM}, t_2G)$$

- Similarly, verifies the relation between the computed value zY_{GM} and zG sent by the user.

$$e(zY_{GM}, G) \stackrel{?}{=} e(Y_{GM}, zG)$$

- Then, he verifies whether the discrete logarithm of L is the multiplication of the discrete logarithms of L_1 and L_2 with respect to G.

$$e(L_1, L_2) \stackrel{?}{=} e(L, G)$$

- Checks the relations:

$$\begin{aligned} e(sB, G) &\stackrel{?}{=} e(B, sG) & e(sB_1, G) &\stackrel{?}{=} e(B_1, sG)\\ e(sB_1, t_2G) &\stackrel{?}{=} e(sB, G) & e(sB_2, t_1G) &\stackrel{?}{=} e(sB, G)\\ e(sL, G) &\stackrel{?}{=} e(sG, L) & e(sL, G) &\stackrel{?}{=} e(sG, L)\\ e(sL_1, G) &\stackrel{?}{=} e(sG, L_1) & e(sL_2, G) &\stackrel{?}{=} e(sG, L_2)\end{aligned}$$

- He accepts only if every of the above equations hold and having $\lambda_{3,1}$ and $\lambda_{3,2}$ the verifier checks whether the following equations hold

$$\begin{aligned}\lambda_{3,1}sG &\stackrel{?}{=} sB_1 + sL & \lambda_{3,1}sG &\stackrel{?}{=} L_1\\ \lambda_{3,2}sG &\stackrel{?}{=} sB_2 + sL & \lambda_{3,2}sG &\stackrel{?}{=} L_2\end{aligned}$$

3 The Flaws and Attacks

In this section we give details about the flaws we found in the design of the examined protocol. We argue, that those flaws are so significant, that the protocol should never be implemented since it do not fulfill the basic security requirements for anonymous authentication schemes namely anonymity and untransferability of group membership.

3.1 Collusion Attack

We investigate the case when two users cooperate. Although, the authors of [14] assume that only one user can become malicious. This is a very strong assumption and may be impractical in real world. Note that the below attack also works if one user registers twice, possible under different identifiers.

Let U_1 and U_2 be two users registered by the GM. Each of them obtains three values from GM. Suppose U_1 received:

$$\begin{aligned} \gamma_{U_1,1} &= H_0 \cdot \omega_{U_1,1} + R_x(A_{U_1}) \cdot k_{GM}, \\ \gamma_{U_1,2} &= H_0 \cdot s \cdot \omega_{U_1,1} + k_{GM} \\ \text{and} \quad & A_{U_1}. \end{aligned}$$

Note that H_0 is a publicly known value, thus we got two equations with three unknown variables $\omega_{U_1,1}$, s and k_{GM}. This gives us an indefinite system of linear equations. However, if another user registers into the GM, we get two additional equations

$$\begin{aligned} \gamma_{U_2,1} &= H_0 \cdot \omega_{U_2,1} + R_x(A_{U_2}) \cdot k_{GM}, \\ \gamma_{U_2,2} &= H_0 \cdot s \cdot \omega_{U_2,1} + k_{GM}, \end{aligned}$$

but only one new unknown variable, namely $\omega_{U_2,1}$. Note that A_{U_2} is given to the user U_2 in the registration phase.

So, if these two users collude, then they have enough information to compute the GM's secrets s and k_{GM}, since we get a definitive system of four linear equations where there are four unknown variables s, k_{GM}, $\omega_{U_1,1}$ and $\omega_{U_2,1}$.

Computing the private keys of GM obviously corrupts the system totally. Two cooperating users having these keys can create new identities at will, so even tracking these „false" identities is highly impractical because of the possible number of identities which can be created.

3.2 Attack against Anonymity

In this section we explore the possibility to break the anonymity of the scheme presented in [14]. We will show that there exists an algorithm $\mathcal{A}$ that given two transcripts of a protocol execution can distinguish if they were produced by the same prover. The idea behind algorithm $\mathcal{A}$ is presented in the proof of lemma 1.

Lemma 1. *Let*

$$\begin{aligned} T_1 = (&\lambda_1, \lambda_{2,1}, \lambda_{2,2}, \lambda_{3,1}, \lambda_{3,2}, L, sL, sL_1, L_2, sL_2, t_1G, \\ &t_2G, zG, B, sB, B_1, sB_1, B_2, sB_2, N_s, H(D)) \\ T_2 = (&\lambda'_1, \lambda'_{2,1}, \lambda'_{2,2}, \lambda'_{3,1}, \lambda'_{3,2}, L', sL', sL'_1, L'_2, sL'_2, t'_1G, \\ &t'_2G, z'G, B', sB', B'_1, sB'_1, B'_2, sB'_2, N'_s, H(D')) \end{aligned}$$

be two transcripts of executions of the protocol from [14]. There exists an PPT algorithm $\mathcal{A}$ that on input:

$$(H_0, G, sG, (\lambda_{2,1}, sL, t_1G, H(D)), (\lambda'_{2,1}, sL', t'_1G, H(D')))$$

outputs, with overwhelming probability, 1 *if* T_1 *and* T_2 *are transcript of communication between a verifier and the same prover and* 0 *otherwise.*

Proof. We will show the construction of algorithm $\mathcal{A}$. First, it computes:

$$\begin{aligned}\lambda_{2,1} \cdot sG - H_0 \cdot H(D) \cdot sL = \\ &= \gamma_{U,2} t_1 sG + H_0 x_1 H(D) sG - H_0 H(D) s x_1 G \\ &= \gamma_{U,2} t_1 sG\end{aligned}$$

and

$$\begin{aligned}\lambda'_{2,1} \cdot sG - H_0 \cdot H(D') \cdot sL' = \\ &= \gamma_{U',2} t'_1 sG + H_0 x'_1 H(D') sG - H_0 H(D') s x'_1 G \\ &= \gamma_{U',2} t'_1 sG.\end{aligned}$$

Note that $\mathcal{A}$ can compute those values using only the received input. Finally, the algorithm outputs 1 if

$$e(\gamma_{U,2} t_1 sG, t'_1 G) \stackrel{?}{=} e(\gamma_{U',2} t'_1 sG, t_1 G)$$

and 0 otherwise. By the bilinearity of the pairing function e this final verification is equal to:

$$e(G,G)^{\gamma_{U,2} t_1 s t'_1} \stackrel{?}{=} e(G,G)^{\gamma_{U',2} t'_1 s t_1}$$

Note that this equation is only valid if $\gamma_{U,2} \stackrel{?}{=} \gamma_{U',2}$, which implies that $U = U'$.

Now, since by lemma 1 there exists such algorithm $\mathcal{A}$, so each verifier can run $\mathcal{A}$ and break the anonymity of any prover in the system. Note that a prover sends the required, by algorithm $\mathcal{A}$, data while sending the membership proof. In addition, imagine that there is an adversary that eavesdrops on the secure communication between provers and a verifier, using for example a man-in-the-middle attack. Such an adversary is therefore able to run algorithm $\mathcal{A}$, since it eavesdrops all necessary input data. Therefore, such attack cannot only be run by an active adversary in form of the verifier, but also by a passive adversary which eavesdrops on the communication between provers and verifiers.

4 Final Comments

4.1 Design Suggestions

The easiest way to secure the scheme against a collusion attack is to store the secret keys is secure memory (e.g. a Hardware Security Module) and assume that users cannot access these keys and the authentication algorithm is also executed in a separated environment. These however, are very strong assumptions which in practice costs additional infrastructure.

If a secret key issuance procedure, issues keys or part of keys which are linear equations, then one has to take into account the number of unknown variables issued to new users. Basically, for each new equation introduced to the system, in order to keep the linear system indefinite, a new unknown variable has to be introduced as well. There are security assumption which cover this problem e.g. one-more Diffe-Hellman assumptions, [15] for instance. The idea behind such type of assumptions is the following: even if the adversary is given l distributions of a given type, he is unable to produce a $l + 1$ distribution. Note that this simulates the collusion of l users who are trying to create a new user.

The protection of user identity is a more complex task. Especially, when we use pairing friendly groups in the system. The use of pairings allows to create new and interesting protocols which overcome some design problems difficult to solve without them. However we must remember that in some cases (e.g. type 1 pairing) the Decisional Diffe-Hellman is easy in the underlying group. The authors of [14] used a type 1 pairing and we exploited the symmetry of e (i.e. $e(P, Q) = e(Q, P)$ for all $P, Q \in \mathbb{G}_1$) to perform a cross attack against anonymity. We encourage them to look at type 3 pairing functions and the XDH (External Diffe-Hellman) assumption (see [16]). In such a scenario the pairing function e takes arguments not from one group $\mathbb{G}_1$ but from two different groups $\mathbb{G}_1$ and $\mathbb{G}_2$. Thus, pairing of type 3 is asymmetric. In addition, if we assume that the XDH assumption holds (it is suspected that it holds for certain MNT curves [17]) we may use the assumption that the DDH problem is intractable in $\mathbb{G}_1$.

4.2 Conclusion

We have shown flaws in the design of the scheme from [14] and we presented attacks exploiting them. It follows that the proposed scheme should not be implemented since it do not fulfill the security requirements for anonymous authentication protocols. We also described some design suggestions which may help the authors of [14] to fix the flaws in their construction and maybe produce a secure version of the proposed anonymous authentication protocol.

Acknowledgments. This work was done as part of the Ventures/2012- 9/4 project financed by the Foundation for Polish Science.

References

1. De Santis, A., Di Crescenzo, G., Persiano, G., Yung, M.: On monotone formula closure of SZK. In: Proceedings of the 35th Annual Symposium on Foundations of Computer Science, SFCS 1994, pp. 454–465. IEEE Computer Society, Washington, DC (1994)
2. Chaum, D., van Heyst, E.: Group signatures. In: Davies, D.W. (ed.) EUROCRYPT 1991. LNCS, vol. 547, pp. 257–265. Springer, Heidelberg (1991)
3. Ateniese, G., Camenisch, J., Hohenberger, S., de Medeiros, B.: Practical group signatures without random oracles. Cryptology ePrint Archive, Report 2005/385 (2005), `http://eprint.iacr.org/`

4. Bellare, M., Micciancio, D., Warinschi, B.: Foundations of group signatures: formal definitions, simplified requirements, and a construction based on general assumptions. In: Biham, E. (ed.) EUROCRYPT 2003. LNCS, vol. 2656, pp. 614–629. Springer, Heidelberg (2003)
5. Boneh, D., Boyen, X., Shacham, H.: Short group signatures. In: Franklin, M. (ed.) CRYPTO 2004. LNCS, vol. 3152, pp. 41–55. Springer, Heidelberg (2004)
6. Boyen, X., Waters, B.: Compact group signatures without random oracles. In: Vaudenay, S. (ed.) EUROCRYPT 2006. LNCS, vol. 4004, pp. 427–444. Springer, Heidelberg (2006)
7. Groth, J.: Fully anonymous group signatures without random oracles. In: Kurosawa, K. (ed.) ASIACRYPT 2007. LNCS, vol. 4833, pp. 164–180. Springer, Heidelberg (2007)
8. Handley, B.: Resource-efficient anonymous group identification. In: Frankel, Y. (ed.) FC 2000. LNCS, vol. 1962, pp. 295–312. Springer, Heidelberg (2001)
9. Jaulmes, É., Poupard, G.: On the security of homage group authentication protocol. In: Syverson, P. (ed.) FC 2001. LNCS, vol. 2339, pp. 106–116. Springer, Heidelberg (2002)
10. De Santis, A., Di Crescenzo, G., Persiano, G.: Communication-efficient anonymous group identification. In: Proceedings of the 5th ACM Conference on Computer and Communications Security, CCS 1998, pp. 73–82. ACM, New York (1998)
11. Boneh, D., Franklin, M.: Anonymous authentication with subset queries (extended abstract). In: Proceedings of the 6th ACM Conference on Computer and Communications Security, CCS 1999, pp. 113–119. ACM, New York (1999)
12. Schechter, S., Parnell, T., Hartemink, A.: Anonymous authentication of membership in dynamic groups. In: Franklin, M. (ed.) FC 1999. LNCS, vol. 1648, pp. 184–195. Springer, Heidelberg (1999)
13. Dodis, Y., Kiayias, A., Nicolosi, A., Shoup, V.: Anonymous identification in ad hoc groups. In: Cachin, C., Camenisch, J.L. (eds.) EUROCRYPT 2004. LNCS, vol. 3027, pp. 609–626. Springer, Heidelberg (2004)
14. Jalal, S., King, B.: A pairing based cryptographic anonymous authentication scheme. In: Proceedings of the 5th International Conference on Ubiquitous Information Management and Communication, ICUIMC 2011, pp. 32:1–32:8. ACM, New York (2011)
15. Boneh, D., Boyen, X.: Short signatures without random oracles and the sdh assumption in bilinear groups. J. Cryptology 21, 149–177 (2008)
16. Ballard, L., Green, M., de Medeiros, B., Monrose, F.: Correlation-resistant storage via keyword-searchable encryption. IACR Cryptology ePrint Archive 2005, 417 (2005)
17. Scott, M., Barreto, P.S.L.M.: Generating more mnt elliptic curves. Des. Codes Cryptography 38(2), 209–217 (2006)