UCL Crypto Group Technical Report Series

UCL
Université catholique de Louvain

REGARDS

An Attack Against Barua & al. Authenticated Group Key Agreement Protocol

Olivier Pereira, Jean-Jacques Quisquater

GROUPE CRYPTO

http://www.dice.ucl.ac.be/crypto/

Technical Report
CG–2003/3

Place du Levant 3
B-1348 Louvain-la-Neuve, Belgium

Phone: (+32) 10 472541
Fax: (+32) 10 472598

An Attack Against Barua & al. Authenticated Group Key Agreement Protocol

Olivier Pereira[1], Jean-Jacques Quisquater

Oct 27, 2003

Département d'Électricité (DICE), Université catholique de Louvain
Place du Levant 3, B-1348 Louvain-la-Neuve, Belgium
Email: {pereira, quisquater}@dice.ucl.ac.be

1 Introduction

In their paper entitled "Extending Joux's Protocol to Multi Party Key Agreement", Barua, Dutta and Sarkar [1] define a new authenticated group key agreement protocol (which we will call the A-BDS protocol). The security of the unauthenticated version of this protocol relies on the hardness of the Decisional Hash Bilinear Diffie-Hellman problem, and the full (authenticated) protocol is built from the previous one by sending authenticators associated to the different exchanged messages.

In this report, we first describe the A-BDS protocol, then we show that this protocol does not provide the expected key authentication properties. Our attack exploits the lack of explicitness of the adopted authentication mechanism: authenticating the origin of a message does not prevent messages generated during one session from being reused in an other session with common participants.

2 The A-BDS protocol

2.1 Protocol Requirements

We summarize here the different definitions which will be used in the A-BDS protocol description.

Let $\mathcal{G}_1$ and $\mathcal{G}_2$ be two groups of the same prime order q. We view $\mathcal{G}_1$ as an additive group and $\mathcal{G}_2$ as a multiplicative group. Let P be an arbitrary generator of $\mathcal{G}_1$. Assume that the discrete logarithm problem is hard for both $\mathcal{G}_1$ and $\mathcal{G}_2$, $e : \mathcal{G}_1 \times \mathcal{G}_1 \rightarrow \mathcal{G}_2$ to be a cryptographic bilinear map

[1]O. Pereira is a Postdoctoral Researcher of the Belgian National Funds for Scientific Research (F.N.R.S.)

CG–2003/3

©2003 by UCL Crypto Group
For more informations, see
http://www.dice.ucl.ac.be/crypto/techreports.html

(such as a Tate pairing) and $H : \mathcal{G}_2 \to \mathbb{Z}_q^*$ to be a one way hash function. These groups and functions will be used to build keys whose secrecy will rely on the hardness of the Decisional Hash Bilinear Diffie-Hellman (DHBDH) problem. This problem can be summarized as follows: given 5 elements (P, aP, bP, cP, r) of $\mathcal{G}_1$ for some a, b, c, r chosen randomly in $\mathbb{Z}_q^*$ and a hash function $H : \mathcal{G}_2 \to \mathbb{Z}_q^*$, decide whether $r = H(e(P, P)^{abc})$ in a polynomial time and with a non negligible probability.

Consider now a group M of n users $\{M_1, \ldots, M_n\}$ who wish to agree upon a group key. Each of these users has a long-term public key $Q_i \in \mathcal{G}_1$ and has obtained from a key generation center (KGC) the corresponding long-term private key $S_i \in \mathcal{G}_1$ [2] computed as $S_i = sQ_i$, where $s \in \mathbb{Z}_q^*$ is the KGC's long-term secret key. These keys will be used to authenticate the different messages, in combination with the value $P_{pub} = sP$ that the KGC publishes.

The A-BDS protocol is contributive: we will assume that each user $M_i \in \mathsf{M}$ will generate a random secret contribution $s_i \in \mathbb{Z}_q^*$. Furthermore, this protocol is recursive and builds the group key by combining partial keys generated by subgroups U of members of M. In these subgroups, a particular user assumes the role of *representative*: the representative of U, denoted $Rep(\mathsf{U})$ is defined as $M_{min(i):M_i \in \mathsf{U}}$. Finally, the A-BDS protocol will exploit a second hash function: $\widehat{H} : \mathcal{G}_1 \to \mathbb{Z}_q^*$ in order to build authenticators: the value aP will be authenticated by M_i by sending the value $\{\!|aP|\!\}_{S_i} = \widehat{H}(aP)S_i + a^2P$.[3]

2.2 Protocol Execution

The A-BDS protocol is a recursive protocol whose definition is made of three functions. The first one, $\mathsf{KeyAgreement}$, is the main function: it manages the way the group key will be constructed from its different parts. This will be carried out calling two other functions: $\mathsf{CombineThree}$ and $\mathsf{CombineTwo}$. The first one allows three groups of users sharing partial keys to generate a common group key, while the second one does the same for two groups of users.

The execution of the A-BDS protocol by a group M of n users will be carried out as described in the procedure $\mathsf{KeyAgreement}(n, \mathsf{M})$, which is defined as follows:

procedure $\mathsf{KeyAgreement}(m, U_{i+1}, \ldots, U_{i+m})$
if $(m = 1)$ **then**
  $KEY := s_{i+1}$;
end if
if $(m = 2)$ **then**

[2] We will sometimes write S_{M_i} for S_i

[3] We will sometimes write $\{\!|sP|\!\}_{M_i}$ instead of $\{\!|sP|\!\}_{S_i}$

call $\mathsf{CombineTwo}(U_{i+1}, U_{i+2}, s_{i+1}, s_{i+2})$;
Let KEY be the agreed key between users U_{i+1} and U_{i+2};
end if
$n_0 := 0$; $n_1 := \lfloor \frac{m}{3} \rfloor$; $n_3 := \lceil \frac{m}{3} \rceil$; $n_2 := m - n_1 - n_3$;
for i:=1 to 3 **do**
call $\mathsf{KeyAgreement}(n_j, U_{i+n_{j-1}+1}, \ldots, U_{i+n_{j-1}+n_j})$;
$\mathsf{U}_j := \{U_{i+n_{j-1}+1}, \ldots, U_{i+n_{j-1}+n_j}\}$; $\widehat{s}_j := KEY$; $n_j := n_{j-1} + n_j$;
end for
call $\mathsf{CombineThree}(\mathsf{U}_1, \mathsf{U}_2, \mathsf{U}_3, \widehat{s}_1, \widehat{s}_2, \widehat{s}_3)$;
end procedure

This function calls the CombineThree function which is defined as follows:

function $\mathsf{CombineThree}(\mathsf{U}_1, \mathsf{U}_2, \mathsf{U}_3, s_1, s_2, s_3)$
for all $i \in \{1, 2, 3\}$ **do**
$Rep(\mathsf{U}_i)$ computes $P_i := s_i P$ and $T_i := \{\!|s_i P|\!\}_{Rep(\mathsf{U}_i)}$
Let $\{j, k\} := \{1, 2, 3\} \setminus \{i\}$
$Rep(\mathsf{U}_i)$ sends P_i and T_i to all members of U_j and U_k
Each member of U_i verifies:
$e(T_j + T_k, P) =$
$e(\widehat{H}(P_j)Q_{Rep(\mathsf{U}_j)} + \widehat{H}(P_k)Q_{Rep(\mathsf{U}_k)}, P_{pub})e(P_j, P_j)e(P_k, P_k)$;
Each member of U_i computes $KEY := H(e(P_j, P_k)^{s_i})$
end for
end function

At the end of this function, the key computed by the members of U_1, U_2 and U_3 is equal to $H(e(P, P)^{s_1 s_2 s_3})$. The CombineTwo function is similar and will not be used further anymore. We describe it however in order to make the A-BDS protocol definition complete:

function $\mathsf{CombineTwo}(\mathsf{U}_1, \mathsf{U}_2, s_1, s_2)$
for all $i \in \{1, 2\}$ **do**
$Rep(\mathsf{U}_i)$ computes $P_i := s_i P$ and $T_i := \{\!|s_i P|\!\}_{Rep(\mathsf{U}_i)}$
$Rep(\mathsf{U}_i)$ sends P_i and T_i to all members of U_{3-i}
Each member of U_{3-i} verifies:
$e(T_i, P) = e(\widehat{H}(P_i)Q_{Rep(\mathsf{U}_i)}, P_{pub})e(P_i, P_i)$;
end for
$Rep(\mathsf{U}_1)$ chooses $\overline{s}$ randomly in $\mathbb{Z}_q^*$
$Rep(\mathsf{U}_1)$ sends $\overline{s}P$ and $\{\!|\overline{s}P|\!\}_{Rep(\mathsf{U}_1)}$ to the rest of the users
Each member of U_1 and U_2 except $Rep(\mathsf{U}_1)$ verifies:
$e(\{\!|\overline{s}P|\!\}_{Rep(\mathsf{U}_i)}, P) = e(\widehat{H}(\overline{s}P)Q_{Rep(\mathsf{U}_1)}, P_{pub})e(\overline{s}P, \overline{s}P)$;
for all $i \in \{1, 2\}$ **do**
Each member of U_i computes $KEY := H(e(P_{3-i}, \overline{s}P)^{s_i})$
end for
end function

At the end of this function, the key computed by the members of U_1 and U_2 is equal to $H(e(P,P)^{s_1 s_2 \overline{s}})$.

3 An attack against the A-BDS protocol

3.1 What ensures key authentication?

It is claimed in [1] that this protocol ensures implicit key authentication, i.e. that all members of a group are guaranteed that only other group members can compute the key they obtained at the end of a protocol session.

We could wonder why this security property would be guaranteed by the A-BDS protocol. To that purpose, let us consider an example of execution of this protocol by a group M of nine participants $\{M_1, \ldots, M_9\}$. If we follow the KeyAgreement procedure execution, we may observe that three instances of the CombineThree function will be started, for the subgroups $\mathsf{U}_1 = \{M_1, M_2, M_3\}$, $\mathsf{U}_2 = \{M_4, M_5, M_6\}$ and $\mathsf{U}_3 = \{M_7, M_8, M_9\}$ respectively. Each member M_i of these subgroups will send the value $s_i P$ to the other two group members, together with the authenticators $\{\!|s_i P|\!\}_{S_i}$. In our further discussions, we will simply consider that each authenticator $\{\!|s_i P|\!\}_{S_i}$ guarantees that M_i really sent the value $s_i P$ (we do not consider the validity of the authentication mechanism adopted: we just consider it in an abstract way, as a classical signature). Finally, at the end of this first round, the members of each group U_i will share a partial key $\widehat{s}_i = H(e(P,P)^{s_{3i-2} s_{3i-1} s_{3i}})$.

During the second round, the representatives of the three subgroups, namely M_1, M_4 and M_7, will use these partial keys to compute values that will be used by the members of the other subgroups to compute the final group key: $H(e(P,P)^{\widehat{s}_1 \widehat{s}_2 \widehat{s}_3})$. These values will be authenticated in the same way as during the previous step of the protocol.

Let us now examine which authentication guarantees the terms of form $\{\!|s_i P|\!\}_{S_i}$ do offer to the different group members, and consider the messages received by M_1 for instance. During the first stage of the protocol, M_1 receives two values together with authenticators proving that they really have been sent by M_2 and M_3. However, since all protocol messages have the same structure, these authenticators do not provide any information about the context in which M_2 and M_3 generated these messages: if M_1 knows that M_2 and M_3 know $s_2 P$ and $s_3 P$, he cannot say whether they are only known by these two users: they could have been sent during any round of any session of the protocol to which these two users are taking part and, in particular, they could have been generated during the second round of a protocol session in which the attacker is a legitimate group member.

We will now sketch a scenario showing how this lack of explicitness could be exploited by an attacker in order to undermine the implicit key authentication property for this protocol.

3.2 Attack Scenario

Let us consider two sessions of the protocol, the first being executed by a pool of users M_1, where:

$$\mathsf{M}_1 = \{M_a, M_b, M_c, M_d, M_e, M_f, M_g, M_h, M_I\}$$

while the second session is executed by a pool of users M_2, where:

$$\mathsf{M}_2 = \{M_1, M_2, M_g\}$$

In these sessions, the attacker is M_I, and the user M_g is member of the two groups. We also assume that the random contributions to the group key are s_a, ..., s_I during the session executed by the members of M_1 and that the contributions are s'_1, s'_2, s'_g in the second session.

We now summarize the first protocol execution.

1. M_a, M_b and M_c compute a partial key $\hat{s}_1 = H(e(P,P)^{s_a s_b s_c})$
2. M_d, M_e and M_f compute a partial key $\hat{s}_2 = H(e(P,P)^{s_d s_e s_f})$
3. M_g, M_h and M_I compute a partial key $\hat{s}_3 = H(e(P,P)^{s_g s_h s_I})$
4. M_a computes $P_1 = \hat{s}_1 P$ and $\{\!|\hat{s}_1 P|\!\}_{M_a}$ as defined in the CombineThree function, and sends these two values to M_d, M_e, M_f, M_g, M_h and M_I.
5. M_d computes $P_2 = \hat{s}_2 P$ and $\{\!|\hat{s}_2 P|\!\}_{M_d}$, and sends these two values to M_a, M_b, M_c, M_g, M_h and M_I.
6. M_g computes $P_3 = \hat{s}_3 P$ and $\{\!|\hat{s}_3 P|\!\}_{M_g}$, and sends these two values to M_a, M_b, M_c, M_d, M_e and M_f.
7. All members of M_1 check the authenticators and compute the group key as $KEY = H(e(P,P)^{\hat{s}_1 \hat{s}_2 \hat{s}_3})$

We now consider the second protocol execution, during which the message sent by M_g will be replaced by the message $\hat{s}_3 P$, $\{\!|\hat{s}_3 P|\!\}_{M_g}$ he sent during the first protocol execution, its particularity being that M_I knows $\hat{s}_3$:

1. M_1 sends $s'_1 P$ and $\{\!|s'_1 P|\!\}_{M_1}$ to M_2 and M_g
2. M_2 sends $s'_2 P$ and $\{\!|s'_2 P|\!\}_{M_2}$ to M_1 and M_g
3. The attacker intercepts the message that M_g sends to M_1 and M_2, and replaces it with the values $\hat{s}_3 P$ and $\{\!|\hat{s}_3 P|\!\}_{M_g}$ that M_g sent during the previous protocol execution.
4. M_1 and M_2 check the authenticators and compute the group key as $KEY' = H(e(P,P)^{s'_1 s'_2 \hat{s}_3})$ (while M_g is computing the group key as $KEY'' = H(e(P,P)^{s'_1 s'_2 s'_g})$).

But the value KEY' can easily be computed by the attacker who knows $s'_1 P$, $s'_2 P$ and $\hat{s}_3$. So, at the end of this scenario, the attacker is able to compute a key that M_1 and M_2 believe to be out of reach for any user that is not included in the M_2 group.

4 Concluding Remarks

In this report, we examined the authentication mechanism adopted in the A-BDS group key agreement protocol in order to achieve the implicit authentication of the group key. We showed that this mechanism lacks of explicitness, what results in the possibility for an active attacker to undermine the key authentication property.

Replacing the current authentication mechanism by the use of a classical signature scheme and including each subgroup constitution in the signed messages would prevent the exhibited attack. However, we think it would be important to also consider freshness issues in order to prevent the replay of messages containing old, maybe compromised, key contributions. Achieving freshness guarantees at low cost (without using timestamps and with a limited communication overhead) would be an interesting direction for future research.

References

[1] Rana Barua, Ratna Dutta, and Palash Sarkar. Extending joux's protocol to multi party key agreement. Cryptology ePrint Archive, Report 2003/062, 2003. `http://eprint.iacr.org/`.