Blake-Wilson, Johnson & Menezes Protocol Revisited

Hai Huang, Zhenfu Cao

Department of Computer Science and Engineering, Shanghai Jiaotong University, 800 Dongchuan Road, Shanghai, 200240, People's Republic of China

Abstract

In this paper, we investigate the famous Blake-Wilson, Johnson & Menezes (BJM) authenticated key exchange protocols. We observe that the Corrupt query in the BJM model is not very reasonable, i.e. it fails to model the adversary's capability well. We modify the BJM model by providing it with a new Corrupt query. By this way, we bring the BJM model further to the practice. Besides, our modification has a significant impact on the security proofs of the BJM protocols. Specifically, the security proofs using CDH assumption will no longer work in the modified BJM model. With slight modification, we show that the BJM protocols are secure in the modified BJM model under the gap Diffie-Hellman assumption (GDH).

Key words: Authenticated key exchange, Provably secure, BJM protocol, Gap Diffie-Hellman;

1. Introduction

Key exchange (KE) allows two parties, Alice (A) and Bob (B), to establish a shared session key via unsecured channels. Later, the shared session key can be used to ensure data confidentiality and integrity between A and B using efficient symmetric encryptions and message authentication codes (MAC). The classical Diffie and Hellman [6] (DH) key exchange protocol first provides a solution to this issue.

Authenticated key exchange (AKE) assures both parties that no other parties aside from their intended peers *may* learn the established session key. A key exchange (KE) protocol is said to provide key confirmation, if both parties are sure that the intended peers really hold the session key. A protocol which is an authenticated key exchange with key confirmation protocol is called AKC protocol [4].

The authenticated key exchange protocols have been established to be surprisingly difficult to design and there are a large number of possible attacks

Email addresses: `chinesechess@sjtu.edu.cn` (Hai Huang), `zfcao@cs.sjtu.edu.cn` (Zhenfu Cao)

against the protocols. The traditional trial-and-error design approach is not very desirable and has led to the situation that the flaws in the AKE protocols have taken many years to discover. This has highlighted the importance of examining the AKE protocols in the formal security model.

Bellare and Rogaway (BR93) [2] first propose a formal security model for authentication and key distribution. They consider a scenario in which both two parties share a long term key and want to establish a common session key. In the BR93 model, the probabilistic polynomial time (PPT) adversary M has full control of the communication network and its capability is modeled by providing it with Send, Reveal oracle queries. The Send query allows the adversary to send the message of its choice to some party and obtains the response, while Reveal query allows the adversary to expose the session keys of some parties.

Subsequent work by Bellare and Rogaway (BR95) [3] considers key distribution in the three-party server-based setting (3PKD) in which party A who shares a long term key K with the server wants to establish a session key with another party B (with the help of the server). In addition to the existing oracle queries in the BR93 model, they additionally introduce a new Corrupt(A, K') query by which the adversary obtains the long term key and the internal state of party A, and replaces it with a new value K' of its choice. From that point on, the server will use the revised long term key. If the adversary later wants to modify A's long term key to new one K'' she can issue a Corrupt(A, K'') query (ignoring the returned string). The BR95 model is further developed by papers [1, 4, 5, 7] later.

Blake-Wilson, Johnson & Menezes (BJM) [4] first consider the issue of authenticated key exchange in the asymmetric (public key) setting in which each party holds a public key/private key pair and wants to establish a common session key. This famous paper has the significant influence on the subsequent work in the key exchange field.

- The paper first proposes the formal definition of secure AKE and AKC protocols, both of which have later been accepted as the standard definition for the key exchange protocols.

- The paper first introduces several important concepts such as key compromise impersonation (KCI), unknown key-share (UKS), both of which are extensively studied by later work.

- The paper presents four key exchange protocols in which the protocol 1 and protocol 2 are shown to be secure three-pass AKC protocols provided that computational Diffie-Hellman (CDH) problem and message authentication code (MAC) algorithm are secure. The protocol 3 and protocol 4 are two-pass AKE protocols. The paper gives a short proof for protocol 3 and a conjectured security for protocol 4 which is further analyzed and improved by later papers [10, 8].

1.1. Corrupt queries in BR95 and BJM model

The BR95 model considers the symmetric setting in which party A who shares a long term key K with the server wants to establish a session key with another party B (with the help of the server).

- **Corrupt**(A, K') in the BR95 model: The adversary obtains the long term key K of party A from the oracle. Then the oracle replaces K with a new value K' of the adversary's choice, i.e. $K \leftarrow K'$. From that point on, the corrupted party A will use the revised long term key K'. If the adversary later wants to modify A's long term key to new one K'' she can issue a Corrupt(A, K'') query (ignoring the returned string).

Actually, in the practice (the symmetric setting), if the adversary want to update the party A's long term key K, it is *necessary* for the adversary to present the new long term key K' of its choice to the server with which it should share the long term key K'. Otherwise, the key exchange protocol will not proceed correctly. So the Corrupt query in the BR95 models the real world (the symmetric setting) well.

Note that the Corrupt query in the BJM model (the public key setting) directly follows from that of the BR95 model (the symmetric setting). Assume that $P_A = g^{S_A}$ is the party A's public key and S_A is the private key.

- **Corrupt**(A, S'_A) in the BJM model: adversary obtains the party A's private key S_A from the oracle and the oracle updates the S_A with a new value S'_A of the adversary's choice, i.e. $S_A \leftarrow S'_A$. From that point on, all the parties will use party A's new public key $P'_A = g^{S'_A}$.

Unfortunately, we find that the Corrupt query in the BJM model does not model the real world (the public key setting) well. For example, in the practice if the adversary want to modify the private key S_A of some corrupted party A with new private key S'_A of its choice, the adversary is supposed to present the corresponding public key $P'_A = g^{S'_A}$ to the certification authority (CA) which issues a certificate binding the new public key P'_A with party A's identity. [1] Anyway, it is *hardly* possible for the adversary to present the new private key S'_A to anyone else (including CA). So we argue that the Corrupt query in the BJM model (the public key setting) which inherits that of the BR95 model (the symmetric setting) fails to model the real world well.

1.2. Our Contributions

In view of the weakness of the Corrupt query in the BJM model, in this paper, we replace it with a new Corrupt(A) query in which the adversary obtains

[1]No other actions by the CA are required or assumed except that CA had better check the validity of the public key P_A, e.g. checking if $P'_A \in \mathbb{G}$ (even a proof of possession of the private key is not mandatory in the practice).

the party A's private key S_A and takes fully control of the party A [2]. From that point on, the adversary can arbitrarily replace S_A with new private key S'_A *without* presenting it to the oracle, and party A will use the new public key P'_A later.

In practical terms this means that the security of our protocols does not depend on the certification authority requiring registrants of public keys to present its private key or prove knowledge of the corresponding private keys.

We stress that the modification not only brings the security model further to the practice, but also has a significant impact on the security proofs of the BJM protocols. Specifically, the protocol 1 and protocol 2 will *no longer* be provably secure under the CDH assumption if we examine them in the security model with modified Corrupt query.

Finally, we slightly modify the BJM protocol 1 and protocol 2 and show that both of them are secure in the modified BJM model (with new Corrupt query) under the gap Diffie-Hellman assumption (GDH) [11].

2. Preliminaries

In this section, we present several established results and tools needed in this paper. Let the value κ be the security parameter. Let p, q be primes, where $q|p-1$. Let $\mathbb{G} = \langle g \rangle$ be the cyclic group of order q. Define CDH(U,V):=Z, where U=g^u, V=g^v and Z=g^{uv}.

2.1. DDH Assumption

Consider the two probability distributions (U,V,CDH(U,V)) and (U,V,Z) where U,V,Z $\in \mathbb{G}$.

For any probabilistic polynomial time algorithm A,

$$|\Pr[\text{A(p, q, g, U}=g^u, \text{V}=g^v, \text{CDH(U,V))=1}]$$
$$-\Pr[\text{A(p, q, g, U}=g^u, \text{V}=g^v, \text{Z)=1}]| \leq \epsilon(\kappa)$$

where $u, v \in \mathbb{Z}_q$ and $\epsilon(\kappa)$ is negligible. The probability is taken over the coin tosses of A, the choice of p, q, g and the random choices of u, v in $\mathbb{Z}_q$.

2.2. CDH Assumption.

For any probabilistic polynomial time algorithm A,

$$\Pr[\text{A}(p, q, g, \text{ U}=g^u, \text{ V}=g^v) = \text{CDH(U,V)}] \leq \epsilon(\kappa).$$

where $u, v \in \mathbb{Z}_q$ and $\epsilon(\kappa)$ is negligible. The probability is taken over the coin tosses of A, the choice of p, q, g and the random choices of u, v in $\mathbb{Z}_q$.

[2]The later papers [7, 9] even allow the adversary to reveal the party's private key without fully control the party. By this way, their models cover the KCI attack.

2.3. GDH Assumption.

For any probabilistic polynomial time algorithm A,

$$\Pr[\mathrm{A}^{\mathrm{DDH}(\cdot)}(p, q, g,\ \mathrm{U}{=}g^u,\ \mathrm{V}{=}g^v) = \mathrm{CDH(U,V)}] \leq \epsilon(\kappa)$$

where $u, v \in Z_q$, and where $\epsilon(\kappa)$ is negligible. The DDH$(\cdot)$ denotes that A has oracle access to DDH. The probability is taken over the coin tosses of A, the choice of p, q, g and the random choices of u and v in Z_q.

2.4. MAC.

For any probabilistic polynomial time algorithm A,

$$\Pr[K \in_R \{0,1\}^\kappa;\ \mathrm{A}^{\mathrm{MAC}_K(\cdot)}(\kappa) = (m, a{=}\mathrm{MAC}_K(m))] \leq \epsilon(\kappa)$$

where $\epsilon(\kappa)$ is negligible. The MAC$(\cdot)$ denotes that A has oracle access to MAC. Note that we require that m has not been queried against the MACing oracle.

3. Review of BJM's work

3.1. Security Model

In this section, we review the BJM security model. For more details, see [4].
Participants. We model the protocol participants as a finite set $\mathcal{U}$ of fixed, polynomial size with each $A \in \mathcal{U}$ being a probabilistic polynomial time (PPT) Turing machine. Each protocol participant $A \in \mathcal{U}$ may execute a polynomial number of protocol instances in parallel. We will refer to s-th instance of principal A communicating with peer B as $\Pi^s_{A,B}(A, B \in \mathcal{U})$ (*a session* or *an instance*).
Adversary Model. The adversary M is modeled as a PPT Turing machine and has full control of the communication network and may eavesdrop, delay, replay, alter and insert messages at will. We model the adversary's capability by providing it with oracle queries.

- **Reveal**$(\Pi^s_{A,B})$ The adversary obtains the session key of $\Pi^s_{A,B}$, provided that the session holds a session key.
- **Corrupt**(A, S'_A) The adversary obtains the private key S_A of party A, and the oracle replaces party A's private key with new one S'_A, i.e. $S_A \leftarrow S'_A$. From that point on, all the parties will use party A's new public key $P'_A = g^{S'_A}$.
- **Send**$(\Pi^s_{A,B}, m)$ The adversary sends the message m to the session $\Pi^s_{A,B}$ and gets a response according to the protocol specification.
- **Test**$(\Pi^s_{A,B})$ Only one query of this form is allowed for the adversary. Provided that the session key is defined, the adversary M can execute this query at any time. Then with probability 1/2 the session key and with probability 1/2 a uniformly chosen random value $\zeta \in \{0,1\}^\kappa$ is returned.

Let No-Matching denote the event that there exists a session $\Pi^s_{A,B}$ which has accepted, but there exists no session $\Pi^t_{B,A}$ which has engaged in a matching conversation [3] to $\Pi^s_{A,B}$. Further, we require that both of party A and B are uncorrupted. Let No-Matching$^{nm}_{M,\Sigma}(\kappa)$ be the probability that the event No-Matching occurs.

Definition 1 (Freshness for AKE Protocols). *Let instance $\Pi^s_{A,B}$ be a completed session, which is executed by party A with another party B. We define $\Pi^s_{A,B}$ to be* fresh *if it has accepted, neither A nor B has been corrupted, and neither $\Pi^s_{A,B}$ nor its matching conversation $\Pi^t_{B,A}$ (if exists) has been revealed.*

Definition 2 (AKE Security). *As a function of the security parameter κ, we define the advantage $Adv^{ake}_{M,\Sigma}(\kappa)$ of the PPT adversary M in attacking protocol Σ as*

$$Adv^{ake}_{M,\Sigma}(\kappa) \overset{def}{=} |2 \cdot Succ^{ake}_{M,\Sigma}(\kappa) - 1|$$

*Here $Succ^{ake}_{M,\Sigma}$ is the probability that the adversary queries **Test** oracle to a* fresh *instance $\Pi^s_{A,B}$, outputs a bit $\hat{b}$ such that $\hat{b} = b$, where the bit b is used by the **Test** oracle. We call the authenticated key exchange protocol Σ to be AKE secure if for any PPT adversary M the function $Adv^{ake}_{M,\Sigma}(\kappa)$ is negligible.*

Definition 3 (AKC Security). *A protocol Σ is a secure AKC protocol if*

1. *In the presence of the benign adversary on $\Pi^s_{A,B}$ and $\Pi^t_{B,A}$, both oracles always accept holding the same session key SK, and this key is distributed uniformly at random on $\{0,1\}^\kappa$;*

and if for every adversary M

2. *If uncorrupted oracles $\Pi^s_{A,B}$ and $\Pi^t_{B,A}$ have matching conversation then both oracles accept and hold the same session key SK.*
3. *No-Matching$^{nm}_{M,\Sigma}(\kappa)$ is negligible.*
4. *$Adv^{ake}_{M,\Sigma}(\kappa)$ is negligible.*

3.2. BJM Protocol 1

In this section, we review the BJM protocol 1 in Figure 1 (also Figure 1 in [4]). The discussion on the BJM protocol 2 is to a large extent similar and deferred to Appendix B.

Let the value κ be the security parameter. Let $\mathbb{G} = \langle g \rangle$ be a cyclic group of order q with a generator $g \in \mathbb{G}$. Let H_1,H_2 be two hash functions modeled as random oracles in the security proof. Let $\mathrm{MAC}_K(m)$ be a message authentication code on message m keyed with K. Let $(P_A{=}g^{S_A}, S_A)$ be party A's public key/private key pair and $(P_B{=}g^{S_B}, S_B)$ be party B's public key/private key pair respectively.

[3] Informally, matching conversations mean that two sessions hold the same transcripts which are the concatenations of sent and received messages during the communication.

$$\begin{array}{lcl}
A^{(P_A=g^{S_A})} & & B^{(P_B=g^{S_B})} \\
 & \xrightarrow{\quad g^{R_A} \quad} & \\
 & \xleftarrow{\mathrm{MAC}_K(2,B,A,g^{R_B},g^{R_A})} & \\
K = H_1(g^{S_A S_B}) & \xrightarrow{\mathrm{MAC}_K(3,A,B,g^{R_A},g^{R_B})} & K = H_1(g^{S_A S_B}) \\
SK = H_2(g^{R_A R_B}) & & SK = H_2(g^{R_A R_B})
\end{array}$$

Figure 1: BJM Protocol 1

Theorem 1 (Theorem 8 in [4]). *BJM protocol 1 in Figure 1 is a secure AKC protocol in the sense of Definition 3 provided the CDH and MAC are secure and H_1 and H_2 are independent random oracles.*

Below we review the basic idea of the proof, which is helpful to understand our improvement.
Condition 1 and 2: The proofs are omitted due to the relative simplicity.
Condition 3: They prove that No-Matching$^{nm}_{M,\Sigma}(\kappa)$ is negligible. Recall that No-Matching is event that there exists a session $\Pi^s_{A,B}$ which has accepted, but there exists no session $\Pi^t_{B,A}$ which has engaged in a matching conversation to $\Pi^s_{A,B}$. Note that we require that both of party A and B are uncorrupted.

- Case 1. Before they prove that No-Matching$^{nm}_{M,\Sigma}(\kappa)$ is negligible, they first prove that the probability that the event A_κ happens is negligible under CDH assumption. The event A_κ happens when there exists two uncorrupted parties A, B for which $g^{S_A S_B}$ is queried of H_1 by the adversary M.

- Case 2. They prove that if event A_κ does not occur, the probability No-Matching$^{nm}_{M,\Sigma}(\kappa)$ is negligible under MAC assumption.

Condition 4: They prove that Adv$^{ake}_{M,\Sigma}(\kappa)$ is negligible.

4. Improvement on BJM Model

As shown in the introduction, upon receipt of the Corrupt(A, S'_A) queries, the oracle gives party A's private key S_A to the adversary M and replaces S_A with a new private key S'_A of the adversary M's choice. From this point on, party A is fully under control of the adversary M. However, we stress that in the real world if the adversary wants to replace the old private key of party A with new one, it is not necessary for the adversary M to show the new private key S'_A to anyone else (including CA). So we claim that the definition of the Corrupt query in the BJM model is not very desirable. We modify the Corrupt query as follows:

- **Corrupt(A)**: The adversary obtains the party A's private key S_A. From now, the adversary takes full control of the party A and can arbitrarily replace the old private key S_A with new one S'_A of its choice *without* presenting the new private key to the oracle. From that point on, the corrupted party A will use the new public key P'_A.

Note that the modification not only brings the BJM security model further to the practice, but also has a significant impact on the security proofs of the BJM protocols.

We claim that the existing proof of Case 1 (Theorem 8 in [4]) for the BJM protocol 1 will no longer hold under the CDH assumption in the modified BJM model. Recall that in the Case 1 we want to prove that if there exists an adversary M making event A_κ happens, then we can construct a CDH problem solver F. At the beginning of F, it embeds the CDH instance into the public keys of both parties A and B (see the Theorem 8 of [4] for the details). In other words, during the simulation, F knows neither A's private key nor B's private key.

Unfortunately, if the adversary M makes Corrupt(C) query and subsequently replaces party C's old private key S_C with new private key S'_C of its choice, then the subsequent Send queries to the uncorrupted party A (or B) by the corrupted party C controlled by the adversary M can no longer be responded correctly. The problem is that the simulator F can not compute the key $\mathrm{H}_1(g^{S_A S'_C})$ of MAC since it does not know party C's new private key S'_C while the adversary M can.

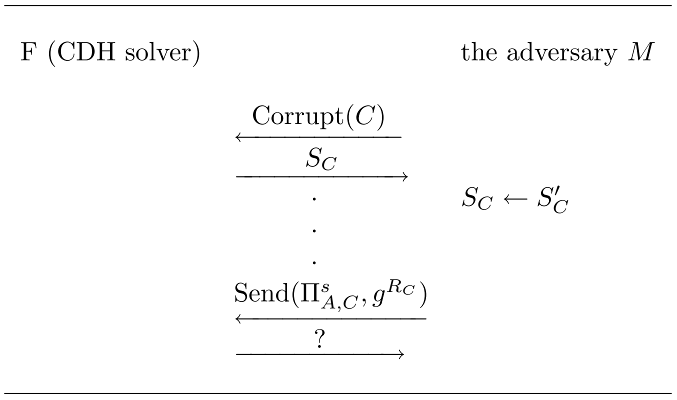

Figure 2: A simulation of BJM protocol 1 in the modified BJM model

For clarity of exposition, we illustrate in Figure 2 that the security proof of the BJM protocol 1 is not correct in the modified BJM model. According to the protocol specification, the simulator F is supposed to respond with $\mathrm{MAC}_K(2, A, C, g^{R_A}, g^{R_C})$ where $K = H_1(g^{S'_C S_A})$. However, F knows neither S'_C nor S_A, and thus it can not produce the MAC value correctly while the adversary M can. In other words, the simulation provided by F will no longer be accurate.

Remark: In the original BJM model, if the adversary want to replace the corrupted party C's private key S_C, it is required to present the new private key S'_C to the oracle, i.e. Corrupt(C, S'_C). With the new private key S'_C, the simulator F can respond correctly the corrupted party C's Send queries to the uncorrupted party A, since F can compute the key $K = H_1(g^{S_A S'_C})$ of MAC. So the BJM protocol 1 can be proven secure under CDH assumption in the original BJM model.

5. New Security Proof in the Modified BJM Model

Before we present our new proof, we give a slightly modified BJM protocol 1 which is identical to the original BJM protocol 1 except that it adds the parties' identities A, B into H_1, i.e. $\mathrm{H}_1(\cdot, A, B)$. By this way, we can simplify the new security proof of the modified BJM protocol 1.

$$
\begin{array}{lcl}
A^{(P_A = g^{S_A})} & & B^{(P_B = g^{S_B})} \\
 & \xrightarrow{\quad g^{R_A} \quad} & \\
 & \xleftarrow{\mathrm{MAC}_K(2, B, A, g^{R_B}, g^{R_A})} & \\
K = H_1(g^{S_A S_B}, \underline{\mathbf{A,B}}) & \xrightarrow{\mathrm{MAC}_K(3, A, B, g^{R_A}, g^{R_B})} & K = H_1(g^{S_A S_B}, \underline{\mathbf{A,B}}) \\
SK = H_2(g^{R_A R_B}) & & SK = H_2(g^{R_A R_B})
\end{array}
$$

Figure 3: Modified BJM Protocol 1

We show that the modified BJM protocol in Figure 3 is secure in the modified BJM model under the gap Difffie-Hellman (GDH) assumption. Specifically, we have following theorem.

Theorem 2. *The modified BJM protocol 1 in Figure 3 is a secure AKC protocol in the sense of Definition 3 (with modified Corrupt query) provided the GDH and MAC are secure and H_1 and H_2 are independent random oracles.*

Proof (sketch). Compared to the proof of Theorem 8 in the original BJM paper [4], the security proof of the modified BJM protocol 1 is identical except for the proof for Case 1 in the Condition 3, in which we make use of the GDH assumption instead of CDH assumption. We explain roughly the idea as follows:

Assume that the adversary has corrupted the party C and replaces its private key S_C with new one S'_C. For the Send queries to uncorrupted party A by the corrupted party C, the simulator F does not know how to compute response message $\mathrm{MAC}_{H_1(g^{S_A S'_C}, A, C)}(\cdot)$ since F knows neither S_A nor S'_C.

The problem is how to provide consistent answers to the adversary M. For example, if F responds with $\mathrm{MAC}_K(\cdot)$ where $K \in \{0,1\}^\kappa$ is chosen randomly, and later M queries H_1 on the explicit value $(g^{S_A S'_C}, A, C)$(the adversary can compute them since it knows S'_C), the answers provided by F will no longer be consistent. It is for this purpose that we use the $\mathrm{DDH}(\cdot,\cdot,\cdot)$ oracle to judge whether or not the adversary queries H_1 on value $(g^{S_A S'_C}, A, C)$. The identities (A, C) acts as the indicators by which F knows when and how to call the DDH oracle. Specifically, upon receipt of $\mathrm{H}_1(\hat{Z}, A, C)$, the simulator F calls $\mathrm{DDH}(P_A, P'_C, \hat{Z})$ where $P_A = g^{S_A}, P'_C = g^{S'_C}$. Otherwise (as the original BJM protocol 1 does), for each query of the form $\mathrm{H}_1(\hat{Z})$, the simulator F must call all the oracles of the form $\mathrm{DDH}(P_{ID_i}, P_{ID_j}, \hat{Z})$ $(ID_i, ID_j \in \mathcal{U})$ until the predicates return 1.

The details of proof for Case 1 will be presented in Appendix A.

6. Conclusions

In this paper, we investigate the famous Blake-Wilson, Johnson & Menezes (BJM) authenticated key exchange protocols. We observe that the Corrupt query in the BJM model is not very reasonable. We modify the BJM model by providing it with a new Corrupt query. Our modification ont only brings the BJM model further to the practice, but also has a significant impact on the security proof of the BJM protocols. Specifically, the security proofs using CDH assumption will no longer work in the modified BJM model. Finally, we present the slight modified BJM protocols and show that they are shown secure in the modified BJM model under the *gap* Diffie-Hellman assumption (GDH).

References

[1] Bellare, M., Pointcheval, D., Rogaway, P., 2000. Authenticated key exchange secure against dictionary attacks. In: EUROCRYPT. Vol. 1807 of Lecture Notes in Computer Science. Springer, pp. 139–155.

[2] Bellare, M., Rogaway, P., 1993. Entity authentication and key distribution. In: Stinson, D. R. (Ed.), CRYPTO. Vol. 773 of Lecture Notes in Computer Science. Springer, pp. 232–249.

[3] Bellare, M., Rogaway, P., 1995. Provably secure session key distribution: the three party case. In: STOC. ACM, pp. 57–66.

[4] Blake-Wilson, S., Johnson, D., Menezes, A., 1997. Key agreement protocols and their security analysis. In: Darnell, M. (Ed.), IMA Int. Conf. Vol. 1355 of Lecture Notes in Computer Science. Springer, pp. 30–45, the full version is available at http://www.math.uwaterloo.ca/ajmeneze/publications/agreement.ps.

[5] Canetti, R., Krawczyk, H., 2001. Analysis of key-exchange protocols and their use for building secure channels. In: Pfitzmann, B. (Ed.), EUROCRYPT. Vol. 2045 of Lecture Notes in Computer Science. Springer, pp. 453–474.

[6] Diffie, W., Hellman, M. E., 1976. New directions in cryptography. IEEE Transactions on Information Theory IT-22 (6), 644–654.

[7] Krawczyk, H., 2005. HMQV: A high-performance secure Diffie-Hellman protocol. In: Shoup, V. (Ed.), CRYPTO. Vol. 3621 of Lecture Notes in Computer Science. Springer, pp. 546–566.

[8] Kudla, C., Paterson, K. G., 2005. Modular security proofs for key agreement protocols. In: Roy, B. K. (Ed.), ASIACRYPT. Vol. 3788 of Lecture Notes in Computer Science. Springer, pp. 549–565.

[9] LaMacchia, B. A., Lauter, K., Mityagin, A., 2007. Stronger security of authenticated key exchange. In: Susilo, W., Liu, J. K., Mu, Y. (Eds.), ProvSec. Vol. 4784 of Lecture Notes in Computer Science. Springer, pp. 1–16.

[10] Lauter, K., Mityagin, A., 2006. Security analysis of KEA authenticated key exchange protocol. In: Yung, M., Dodis, Y., Kiayias, A., Malkin, T. (Eds.), Public Key Cryptography. Vol. 3958 of Lecture Notes in Computer Science. Springer, pp. 378–394.

[11] Okamoto, T., Pointcheval, D., 2001. The gap-problems: A new class of problems for the security of cryptographic schemes. In: Kim, K. (Ed.), Public Key Cryptography. Vol. 1992 of Lecture Notes in Computer Science. Springer, pp. 104–118.

Appendix A

Let A_κ be the event that, during the adversary M's experiment, there exists a pair uncorrupted parties $A, B \in \mathcal{U}$ for which the value of the form $(g^{S_A S_B}, A, B)$ is queried against H_1 by the adversary M.

Case 1: Suppose that $\Pr[A_\kappa]$ is non-negligible. In this case we construct from the adversary M an algorithm F which is used to solve the GDH problem with non-negligible probability.
F's operation: F takes a GDH problem instance (U=g^u,V=g^v), and tries to compute CDH(U,V)=g^{uv} with access to the DDH$(\cdot,\cdot,\cdot)$ oracle.

F randomly picks two parties $A, B \in \mathcal{U}$, guessing that the adversary M will query H_1 with the value of the form $(g^{S_A S_B}, A, B)$. F assigns the public/private pairs for all parties except for A's and B's. F sets A's public key to be U, and B's public key to be V respectively.

F answers all Reveal queries and H$_2$ queries as specified by protocol. F answers all Send queries as specified by protocol except for Send queries to party A and B. Below we just describe F's actions when the adversary M makes the Send queries to the party B (the Send queries to party A can be similarly dealt with). Note that party C may be corrupted.

In the simulation below, simulator F introduces a list L^{list}, which stores entries with the form $(ID_i, ID_j, *)$. The third element records the key K of MAC, which is either $CDH(P_{ID_i}, P_{ID_j})$ or a random value in group $\mathbb{G}$.

- **Send**$(\Pi_{B,C}, \mathcal{M}{=}\lambda)$: The simulator F chooses $R_B \in \mathbb{Z}_q$ and responds with g^{R_B}.
- **Send**$(\Pi_{B,C}, \mathcal{M}{=}g^{R_C})$: The simulator F maintains an initially empty list L^{list} with entries of the form $(ID_i, ID_j, *)$. F looks in L^{list} for the entry of the form $(C, B, *)$.
  - If finds it, F responds with $\mathrm{MAC}_K(2, B, C, g^{R_B}, g^{R_C})$ where K is the third element of this entry in L^{list}.
  - Otherwise, F chooses $K \in \{0,1\}^{\kappa}$ at random and responds with $\mathrm{MAC}_K(2, B, C, g^{R_B}, g^{R_C})$. Then F stores the new entry of the form (C, B, K) in L^{list}.
- **Send**$(\Pi_{B,C}, \mathcal{M}{=}\mathrm{MAC}_K(2, C, B, g^{R_C}, g^{R_B}))$: F looks in L^{list} for the entry of the form $(B, C, *)$.
  - If finds it, F checks if $\mathcal{M}{=}\mathrm{MAC}_K(2, C, B, g^{R_C}, g^{R_B})$ where K is the third element of this entry in L^{list}.
    - If the predicates evaluates to 1, F responds with $\mathrm{MAC}_K(3, B, C, g^{R_B}, g^{R_C})$.
    - Otherwise, F rejects this session.
  - Otherwise (no such entry exists), F rejects this session.

  F answers all H_1 queries as specified by protocol except for H_1 queries on $(\hat{Z}, C, B)$ and $(\hat{Z}, B, C)$ (the queries on $(\hat{Z}, C, A)$ and $(\hat{Z}, A, C)$ are similarly dealt with).

- $\mathbf{H}_1(\hat{Z}, C, B)$ $\big($or $\mathbf{H}_1(\hat{Z}, B, C)\big)$: F maintains an initially empty list H_1^{list} with entries of the form $(\hat{Z}, ID_i, ID_j, *)$. The simulator F responds to these queries in the following way:
  - If $(\hat{Z}, C, B, *)$ $\big($or $(\hat{Z}, B, C, *)\big)$ is already there, then F responds with the stored fourth value h.
  - Otherwise, F checks if $\hat{Z}{=}g^{S_C S_B}$ by calling oracle $\mathrm{DDH}(P_C, P_B, \hat{Z})$ where $P_C = g^{S_C}$,$P_B = g^{S_B}$. If the DDH predicates evaluates to 1, F looks in L^{list} for the entry of the form $(C, B, *)$ $\big($or $(B, C, *)\big)$.
    - If F finds it, F stores new entry of the form $(\hat{Z}, C, B, K)$ $\big($or $(\hat{Z}, B, C, K)\big)$ in H_1^{list}, where K is the third element of this entry in L^{list}. Then, F responds with K.
    - Otherwise (no such an entry in L^{list}), F randomly chooses $h \in \{0,1\}^k$, stores a new entry of the form (C, B, h) $\big($or $(B, C, h)\big)$ in L^{list}. Also, F stores a new entry of the form $(\hat{Z}, C, B, h)$ $\big($or $(\hat{Z}, B, C, h)\big)$ in H^{list} and responds with h.

  - Otherwise (DDH predicates evaluates to 0), F chooses $h \in \{0,1\}^k$ at random, sends it to the adversary M and stores the new tuple $(\hat{Z}, C, B, h)$ $\big($or $(\hat{Z}, B, C, h)\big)$ in H_1^{list}.

- **Corrupt**(ID_i):
  - If $ID_i{=}A$ or $ID_i{=}B$, F aborts.
  - Otherwise, F returns the party ID_i's private key to the adversary.

Now, if the adversary makes event A_κ happen, i.e. it has made queries H_1 oracle on the the value of the form $(g^{S_A S_B}{=}g^{uv}, A, B)$, for each entry of the form $(\hat{Z}, A, B)$ in H_1^{list}, F calls the DDH$(U, V, \hat{Z})$. If the DDH predicates evaluate to 1, F stops and outputs $\hat{Z}$=CDH(U,V). The probability that F outputs the value CDH(U,V) is

$$\Pr[F] \geq \frac{Pr[A_\kappa]}{T_1(\kappa)^2}$$

where $T_1(\kappa)$ is the maximum number of parties in the system. So we can conclude that if $Pr[A_\kappa]$ is non-negligible then F can solve the GDH problem with non-negligible probability. This contradicts the assumed hardness of the GDH problem.

Appendix B

1. BJM protocol 2

$$\begin{array}{lcl}
A^{(P_A = g^{S_A})} & & B^{(P_B = g^{S_B})} \\
 & \xrightarrow{g^{R_A}} & \\
 & \xleftarrow{\mathrm{MAC}_K(2, B, A, g^{R_B}, g^{R_A})} & \\
K = H_1(g^{R_A R_B}, g^{S_A S_B}) & \xrightarrow{\mathrm{MAC}_K(3, A, B, g^{R_A}, g^{R_B})} & K = H_1(g^{R_A R_B}, g^{S_A S_B}) \\
SK = H_2(g^{R_A R_B}, g^{S_A S_B}) & & SK = H_2(g^{R_A R_B}, g^{S_A S_B})
\end{array}$$

Figure 4: BJM Protocol 2

2. Modified BJM protocol 2

We give a slightly modified BJM protocol 2 in Figure 5 which is identical to the original BJM protocol 2 in Figure 4 (also Figure 2 in [4]) except that it adds the value *sid* into both H$_1$ and H$_2$, i.e. H$_1(\cdot, \cdot, sid)$, H$_2(\cdot, \cdot, sid)$, where $sid = (g^{R_A}, g^{R_B}, A, B)$.

Theorem 3. *The modified BJM protocol 2 in Figure 5 is a secure AKC protocol in the sense of Definition 3 (with modified Corrupt query) provided the GDH and MAC are secure and H_1 and H_2 are independent random oracles.*

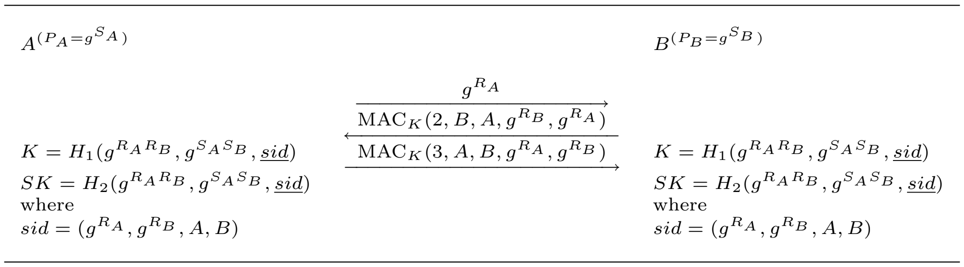
$A^{(P_A=g^{S_A})}$
$B^{(P_B=g^{S_B})}$
g^{R_A}
$\mathrm{MAC}_K(2,B,A,g^{R_B},g^{R_A})$
$\mathrm{MAC}_K(3,A,B,g^{R_A},g^{R_B})$
$K = H_1(g^{R_A R_B}, g^{S_A S_B}, \underline{sid})$
$SK = H_2(g^{R_A R_B}, g^{S_A S_B}, \underline{sid})$
where
$sid = (g^{R_A}, g^{R_B}, A, B)$
$K = H_1(g^{R_A R_B}, g^{S_A S_B}, \underline{sid})$
$SK = H_2(g^{R_A R_B}, g^{S_A S_B}, \underline{sid})$
where
$sid = (g^{R_A}, g^{R_B}, A, B)$

Figure 5: Modified BJM Protocol 2

Proof (sketch). Compared to the proof of Theorem 9 in the original BJM protocol 2 [4], the security proof of the modified BJM protocol 2 is to a large extent similar except for the proof for Case 1 in the Condition 3, in which we make use of the GDH assumption instead of CDH assumption.

The modified BJM Protocol 2 is different from the modified BJM Protocol 1 in that it hashes sid into both H_1 and H_2, where sid includes the identities of parties and the ephemeral public keys.

Assume that the adversary has corrupted the party C, updates its private key to be S'_C, and establishes a session key SK with the uncorrupted party B. For the Reveal queries to party B, the simulator does not know how to return the real session key SK, since F can not compute it without S_B and S'_C. If F responds with a random value $SK \in \{0,1\}^\kappa$, and later M queries H_2 on the explicit value $(g^{R_C R_B}, g^{S'_C S_B}, g^{R_C}, g^{R_B}, C, B)$(the adversary can compute them since it knows S'_C and chooses R_C itself), the answers provided by F will no longer be consistent.

It is for this purpose that F uses DDH oracles to check the validity of the H_2 queries. Specifically, upon receipt of $\mathrm{H}_2(\hat{Z}_1, \hat{Z}_2, g^{R_C}, g^{R_B}, C, B)$, the simulator calls both $\mathrm{DDH}(P'_C, P_B, \hat{Z}_2)$ and $\mathrm{DDH}(g^{R_C}, g^{R_B}, \hat{Z}_1)$ to judge whether or not the adversary queries H_2 on the correct value. The value $sid = (g^{R_C}, g^{R_B}, C, B)$ acts as the indicators for the simulator F.

To avoid repetition, we just give the sketch of the simulator F, in which party C may be corrupted and A, B are uncorrupted.

- **Corrupt**(ID_i): The simulator F's action is the same as that of Theorem 2.

- **Send**: The simulator F's action is similar to that of Theorem 2 except that it maintains a list L^{list} with the form of $(g^{R_{ID_i}}, g^{R_{ID_j}}, ID_i, ID_j, *)$ instead of the form of $(ID_i, ID_j, *)$.

- $\mathbf{H}_1(\hat{Z}_1, \hat{Z}_2, g^{R_C}, g^{R_B}, C, B)$ $\big($or $\mathbf{H}_1(\hat{Z}_1, \hat{Z}_2, g^{R_B}, g^{R_C}, B, C)\big)$: The simulator F maintains an initially empty list H_1^{list} with entries of the form $(\hat{Z}_1, \hat{Z}_2, g^{R_{ID_i}}, g^{R_{ID_j}}, ID_i, ID_j, *)$ instead of the form of $(\hat{Z}, ID_i, ID_j, *)$. As a result, F must call both $\mathrm{DDH}(P_C, P_B, \hat{Z}_2)$ and $\mathrm{DDH}(g^{R_C}, g^{R_B}, \hat{Z}_1)$ oracles to keep the consistency of H_1^{list} and L^{list}.

- **Reveal**$(\Pi^s_{B,C})$: The simulator F maintains an initially empty list T^{list} with entries of the form $(g^{R_{ID_i}}, g^{R_{ID_i}}, ID_i, ID_j, *)$. F looks in T^{list} for the entry of the form $(g^{R_C}, g^{R_B}, C, B, *)$.
  - If finds it, F responds with SK, which is the fifth element of this entry in T^{list}.
  - Otherwise, F chooses $SK \in \{0,1\}^\kappa$ at random and stores the new entry of the form $(g^{R_C}, g^{R_B}, C, B, SK)$ in T^{list}.
- $\mathbf{H}_2(\hat{Z}_1, \hat{Z}_2, g^{R_C}, g^{R_B}, C, B)$ $\big($or $\mathbf{H}_2(\hat{Z}_1, \hat{Z}_2, g^{R_B}, g^{R_C}, B, C)\big)$: F maintains an initially empty list H_2^{list} with entries of the form $(\hat{Z}_1, \hat{Z}_2, g^{R_{ID_i}}, g^{R_{ID_j}}, ID_i, ID_j, *)$. The simulator F calls both DDH$(P_C, P_B, \hat{Z}_2)$ and DDH$(g^{R_C}, g^{R_B}, \hat{Z}_1)$ oracles to keep the consistency of H_2^{list} and T^{list}.