Security of a Leakage-Resilient Protocol for Key Establishment and Mutual Authentication (Extended Abstract)

Raphael C.-W. Phan[1], Kim-Kwang Raymond Choo[2⋆], Swee-Huay Heng[3]

[1] Laboratoire de sécurité et de cryptographie, EPFL, Lausanne, Switzerland
raphael.phan@epfl.ch
[2] Canberra, Australia
raymond.choo.au@gmail.com
[3] Centre for Cryptography and Information Security (CCIS),
Faculty of Information Science and Technology, Multimedia University, Malaysia
shheng@mmu.edu.my

Abstract. We revisit Shin *et al.*'s leakage-resilient password-based authenticated key establishment protocol (LR-AKEP) and the security model used to prove the security of LR-AKEP. By refining the Leak oracle in the security model, we show that LR-AKE (1) can, in fact, achieve a stronger notion of leakage-resilience than initially claimed and (2) also achieve an additional feature of traceability, not previously mentioned.

1 Introduction

Authenticated Key Establishment protocols (AKEPs) allow two parties to share a secret key based on long-term secrets associated with individual entities (typically passwords). Passwords are strings easily memorized by humans and thus of low entropy. Such protocols are especially popular in computationally restricted devices and those requiring interaction with human users. For example, in practical applications, the secrets derived from passwords are stored in some devices (e.g., a table containing hashed values of passwords kept by a trusted server). A fundamental security threat for password-based AKEPs is, unsurprisingly, dictionary attacks due to low entropy of password-based AKEPs.

We revisit the leakage-resilient password-based AKEPs (LR-AKEPs), first proposed by Shin, Kobara and Imai [7] and subsequently extended in [4, 8–10]. LR-AKEPs, designed to maintain the secrecy of the long-term password even in the case when stored secrets (i.e., functions of the password) are leaked, can be broadly categorised into two families: the Diffie–Hellman-based LR-AKEPs [7–9] and the RSA-based LR-AKEPs [4, 10].

Widely used security models for AKEPs (including password-based AKEPs) include the indistinguishability-based models of Bellare, Pointcheval, and Rogaway [1] model (hereafter referred to as the BPR2000 model) and Canetti and

⋆ The views and opinions expressed in this paper are those of the author and do not reflect those of any organisation with which the author may be affiliated. This research was undertaken in the author's personal capacity.

Krawczyk [2][4]. In the BPR2000 model, leakages of established secret session keys and long-term secrets (e.g., private key or password) are considered by allowing the adversary to have access to the Reveal oracle and the Corrupt oracle respectively. To model leakage-resilience, Shin *et al.* [7] introduced an additional Leak oracle that allows the adversary to learn the stored secrets of unrelated sessions.

The focus of this paper is on the Diffie–Hellman-based LR-AKEP published in ASIACRYPT 2003 [7] (hereafter referred to as "the LR-AKE protocol"). A distinct difference between the LR-AKE protocol and latter extensions [8, 9] is that only one secret is stored on the client in the latter schemes.

We regard our contributions in this paper to be three-fold:

1. **Revised security model**: We refine the original model used by Shin *et al.* to prove the security of the LR-AKE protocol by splitting the Leak oracle into LeakC and LeakS oracles[5]. By so doing, we are able to define *how many* leakages occur on the server side.
2. **Stronger notion of leakage-resilience than that defined by Shin *et al.***: Shin *et al.* proved that the LR-AKE protocol is secure when the leaks do not originate from both the client and servers simultaneously. We demonstrate that the LR-AKE protocol can, in fact, provide an *almost* perfect security level.
3. **Notion of traceability not previously mentioned by Shin *et al.***: We demonstrate that the LR-AKE protocol can provide traceability, which allows us to identify the compromised client or server devices when leakages occur.

2 Revisiting the leakage-resilient AKE protocol of ASIACRYPT 2003

The notation used throughout this paper is as described in Table 1.

The LR-AKE protocol, described in Fig. 1, can be considered a two-party password-based AKE involving a client-server pair where the server is one out of $n-1$ possible servers. The client, C, remembers a chosen password, pw, and stores $n-1$ secret values, h_i $(i = 1, \ldots, n-1)$, derived from pw in C's device. A partial secret value, $h^{p(i).\lambda_i}$ for $1 \leq i \leq n-1$ (not a share) of pw, is registered with each of the $n-1$ servers. This will enable C to establish a session key with any of these servers in subsequent sessions. The underlined values in Fig. 1 represent the stored secrets of the respective client and server. Note that we only present sufficient details to understand this paper and we refer interested reader to [7] for further details.

[4] Interested reader is referred to [3] for a comparison and a discussion of existing security models for AKEPs.

[5] The definitions of oracles A1 through A4 in Section 4 of [9] implicitly split the Leak oracle, thereby distinguishing whether the leakage occurs at the client or at the server.

C	The client with identity ID_C
S_i	The i^{th} server with identity ID_{S_i}, $(1 \leq i \leq n-1)$
G	Finite cyclic group of large prime order q
g, h	Generators of G
r_i	A random value in $(\mathbb{Z}/q\mathbb{Z})^*$
pw	The password chosen by the client
$p(\cdot)$	Random polynomial of degree $n-1$ with coefficients also randomly chosen in $(\mathbb{Z}/q\mathbb{Z})^*$; defined as $p(x) = \Sigma_{j=0}^{n-1} \alpha_j \cdot x^j \mod q$ for which $\alpha_0 = pw$
$h^{p(i).\lambda_i}$	The secret value registered by client with server S_i, where $p(i)$ is a share of (n,n)-threshold secret sharing and λ_i is a Lagrange coefficient. Note that $h^{-p(i).\lambda_i} = h_i \cdot h^{-pw}$, which allows for both client and server to compute the same MAC keys km_c and km_s, respectively.
h_i	Client's stored secret corresponding to server S_i; equals $h^{\Sigma_{l=1,l\neq i}^{n} p(l).\lambda_l}$
Tag_c, Tag_s,Tag_{sk}	Pre-determined distinct values, e.g., $Tag_c = (ID_C \| ID_S \| 00)$, $Tag_s = (ID_C \| ID_S \| 01)$ and $Tag_{sk} = (ID_C \| ID_S \| 11)$
$MAC_k(\cdot)$	A MAC generation function with k as its keying material

Table 1. Summary of notations

Client, C		Server, $S_i (1 \leq i \leq n-1)$
$r_1 \overset{R}{\leftarrow} (\mathbb{Z}/q\mathbb{Z})^*$		$r_2 \overset{R}{\leftarrow} (\mathbb{Z}/q\mathbb{Z})^*$
$y_1 \leftarrow g^{r_1} \cdot \underline{h_i} \cdot h^{-pw}$	$\xrightarrow{y_1}$	$y_2 \leftarrow g^{r_2} \cdot \underline{h^{p(i).\lambda_i}}$
$km_c \leftarrow (y_2.h_i \cdot h^{-pw})^{r_1} = g^{r_1 r_2}$	$\xleftarrow{y_2}$	$km_s \leftarrow (y_1 \cdot h^{p(i)\cdot\lambda_i})^{r_2} = g^{r_1 r_2}$
$v_1 \leftarrow MAC_{km_c}(Tag_c \| y_1 \| y_2)$	$\xrightarrow{v_1}$	$v_2 \leftarrow MAC_{km_s}(Tag_s \| y_1 \| y_2)$
If $v_2 = MAC_{km_c}(Tag_s \| y_1 \| y_2)$,	$\xleftarrow{v_2}$	If $v_1 = MAC_{km_s}(Tag_c \| y_1 \| y_2)$,
then $sk_c \leftarrow MAC_{km_c}(Tag_{sk} \| y_1 \| y_2)$.		then $sk_s \leftarrow MAC_{km_s}(Tag_{sk} \| y_1 \| y_2)$.

Fig. 1. Original LR-AKE protocol of Shin, Kobara, and Imai [7]

Shin *et al.* proved the LR-AKE protocol secure against off-line dictionary attacks even if the stored secrets are leaked from *either* the client or up to all $n-1$ servers, but not from *both* client and $n-1$ servers simultaneously [7, Theorem 1].

3 Refining the oracle for leakage resilience

We now revisit the BPR2000 model used by Shin *et al.* to prove the security of the LR-AKE protocol.

Protocol participants. Let $ID \overset{\text{def}}{=} Clients \cup Servers$ be a non-empty set of protocol participants, or *principals*. We assume $Servers$ consists of $n-1$ servers, $\{S_1, \ldots, S_{n-1}\}$ and at any time a client $C \in Clients$ is interacting with a server $S_i \in Servers$ to establish an LR-AKE session.

PROTOCOL EXECUTION. The adversary, $\mathcal{A}$, controls the communications between the protocol participants by interacting with the set of oracles, $\Pi^i_{U_u,U_v}$, where $\Pi^i_{U_u,U_v}$ is defined to be the i^{th} instantiation of a protocol participant, U_u, in a specific protocol run and U_v is the principal with whom U_u wishes to establish a secret key. $\mathcal{A}$ controls the communication channels via the queries to the targeted oracles. A description of the oracle types is presented as follows. Note that we had split the Leak oracle into LeakC and LeakS oracles as this will allow us to distinguish whether the leakage occurs at the client or at the server.

Send(U_u, U_v, i, m) **query.** This query to an oracle, $\Pi^i_{U_u,U_v}$, computes a response according to the protocol specification and decision on whether to accept or reject yet, and returns them to the adversary $\mathcal{A}$. If $\Pi^i_{U_u,U_v}$ has either accepted with some session key or terminated, this will be made known to $\mathcal{A}$. Note that if $m = *$, then this will result in the instantiation of the oracle $\Pi^i_{U_u,U_v}$ if such an oracle has not been created previously.

Reveal(U_u, U_v, i) **query.** Any oracle, $\Pi^i_{U_u,U_v}$, upon receiving such a query and if $\Pi^i_{U_u,U_v}$ has accepted and holds some session key, will send this session key back to $\mathcal{A}$. The Reveal query is designed to capture this notion.

Corrupt(U_u) **query.** This query captures unknown key share attacks and insider attacks. This query allows $\mathcal{A}$ to corrupt the principal U_u at will, and thereby learn the complete internal state of the corrupted principal. Notice that a Corrupt query does not result in the release of the session keys since $\mathcal{A}$ already has the ability to obtain session keys through Reveal queries.

LeakC$(\ell, \jmath)$ **query.** This query allows $\mathcal{A}$ to learn ℓ $(1 \leq \ell \leq n-1)$ stored secrets h_ι of the client oracle and the corresponding indices ι (for $\iota \in \{1, \ldots, n-1\}, \iota \neq \jmath$) of the leaked secrets.

LeakS(t, ι) **query.** This query to a server oracle, U_v, returns the corresponding stored secrets $h^{p(\jmath).\lambda_\jmath}$ of any t $(1 \leq t \leq n-1)$ servers and their corresponding indices $\jmath$ (for $\jmath \in \{1, \ldots, n-1\}, \jmath \neq \iota$) of these leaked servers.

PROTOCOL SECURITY. Security of the LR-AKE protocol [7] is defined in two stages.

1. Proving that the protocol is secure even when the stored secrets are leaked.
2. The standard indistinguishability-based security proof (of the established session key) as required by the BPR2000 model [1].

A revised security proof for the LR-AKE protocol, presented in Appendix A, demonstrates that the LR-AKE protocol described in Fig. 1 provides both key establishment and mutual authentication.

4 Strengthened notions of leakage resilience and traceability

Shin *et al.* [7, 9] state that one cannot achieve security when there are leakages from both the client and server(s) side (see `Fact 2`) , the situation in which they

call perfect security (see `Goal 1`).

`Goal 1`**: Perfect Security [7, 9].** *Any AKE protocol with 'perfect security' remains secure against leakages from client* ***and*** *server(s),* ***simultaneously****.*

`Fact 2`**. Impossibility of Perfect Security [7, 9].** *Any AKE protocol cannot achieve (strong) security against leakage from* ***both*** *a client and servers simultaneously. If an adversary obtains stored secrets from both a client and servers at the same time, s/he can perfectly simulate the protocol using the leaked secrets. Thus s/he can try the password candidates off-line in parallel.*

Shin *et al.* then argue that the next highest achievable goal is the security of the password against offline dictionary attacks even in the situation when there are "leakages" from *either* the client *or* the server(s) (see `Goal 2` below).

`Goal 2`**: Strong Security [7, 9].** *In absence of 'perfect security', [7, 9] claim that the next highest goal is to achieve so-called 'strong security', i.e. security against the "leakages" from a client and servers,* ***respectively****.*

We can view 'perfect security' described in `Goal 1` as security against leakages from *both* the client *and* the server(s), while 'strong security' described in `Goal 2` as security against leakages from *either* the client *or* the server(s). We argue that their requirement is too strong, i.e., unnecessarily restrictive as `Goal 2` is not the next best security in the absence of `Goal 1`. We can still have security against leakages from *both* the client *and* server(s) with some trade-off.

Relation between the LR-AKE protocol and an (n, n) secret-sharing scheme

The reader might have observed that in the LR-AKE scheme, the n shares of the secret password pw are not separated uniformly among the $n-1$ servers and the client. Therefore, leakage from a client should not be treated in the same way as leakage from a server – leakage of a stored secret from any server contains information about just one share whilst leakage from the client constitutes the entire stored secret, h_i. It should come as no surprise that the LR-AKE protocol will be insecure if there are leakages from the client (i.e., $n-1$ shares are leaked) and one or more servers.

We can, however, relax this strong requirement to achieve perfect security to a certain extent. We termed this as *almost perfect security*, which can be formalized by splitting the Leak queries (as described in Section 3). By having a separate LeakC query for the client oracle and a separate LeakS query for the server oracle, we are able to formally state:

1. whether the leakage is from the client or the server, and
2. how many stored secrets are leaked.

Making the former explicit is useful because leakages from a server contain only information about a particular share, while leakage from a client contains information about $n-1$ shares. Making the latter explicit is also useful because by

knowing which client's stored secret has been compromised, we will know the corresponding compromise at the server(s). Consequently, this allows us to show that the original LR-AKE protocol proposed in [7] can achieve a stronger notion of leakage resilience (in the sense of 'almost perfect security' as described in Goal 3).

`Goal 3`**. Almost Perfect Security.** *Any AKE protocol with 'almost perfect security' remains secure even against leakages from* ***both*** *the client* ***and*** *up to* $n-2$ *servers,* ***simultaneously****.*

Since `Goal 1` trivially implies `Goal 3` and `Goal 3` is a stronger notion than `Goal 2`, we now have the following result.

Theorem 1 The LR-AKE protocol achieves `Goal 3` (almost perfect security) even when both the client and up to $n-2$ servers leak their corresponding stored secrets, as long as the leaked secret(s) h_ι of the client and leaked secret(s) of the server(s) $S_\jmath$ are such that $\jmath \neq \iota$.

Proof Intuition. Recall that:

- a $\mathsf{LeakC}(\ell, \jmath)$ query allows the adversary $\mathcal{A}$ to learn ℓ $(1 \leq \ell \leq n-1)$ stored secrets h_ι of the Client, and the corresponding indices ι (for $\iota \in \{1, \ldots, n-1\}$) of these leaked secrets, thus as long as the LeakS queries are issued only to servers $S_\jmath$ for $\jmath \neq \iota$, there are insufficient shares (since number of leaked shares $< n$) to reveal the shared secret pw.
- a $\mathsf{LeakS}(t, \iota)$ query allows the adversary $\mathcal{A}$ to learn t $(1 \leq \jmath \leq n-1)$ stored secrets $h^{p(\jmath).\lambda_\jmath}$ of t Servers $S_\jmath$, and the corresponding indices $\jmath$ (for $\jmath \in \{1, \ldots, n-1\}$) of these leaked servers, thus as long as the LeakC query returns only stored secrets h_ι of the client such that $\iota \neq \jmath$, there insufficient shares to reveal the shared secret pw.

□

Although we cannot prove the protocol secure when leakages originate from both the client *and all* servers, we can prove that the protocol remains secure when the leakages originate from the client and up to $n-2$ servers, even in the case when the client leaks more than one (up to $n-2$) secret(s). The conditions necessary for achieving `Goal 3` is that the total of the stored secrets leaked by the client and the servers cannot exceed $n-1$, and all their indices are different. Consequently, we can view `Goal 3` as a special case of `Goal 1` in the sense that we can still maintain security when we have leakages from both the client *and* the server(s) simultaneously.

Traceability. In our setting, the stored secrets can be leaked from either the client or server(s), or both. Although it is hard to prevent such leakages, the client would most likely to be interested to know which particular stored secret has been leaked. This is similar to the copyright violator identification and dispute resolution (arbitration) scenario (e.g., in buyer-seller protocols [6]). This

allows us to handle cases where leakages are unavoidable, but future preventive measures can be taken by firstly identifying the compromised client or server devices.

In the context of the LR-AKE schemes, we should be able to determine the compromised site (i.e., which particular server) since every registered stored secret is unique. Consequently, we are able to demonstrate that the original LR-AKE scheme provides *traceability* (i.e., in the event of leakages that compromise the security of the password, it is possible to precisely pinpoint which server(s) leaked). For example, when a password has been compromised, we know that it is likely that the compromise is at the client site (except with negligible probability). We can, therefore, trace which particular server(s) had caused the leakage by simply checking which stored secret(s) of the client, h_i, is (are) leaked.

It appears that the original LR-AKE protocol [7] is the only scheme in their family that provides such a (traceability) feature as in the other variants (e.g., [8–10]), the Client stores only one secret instead of unique ones corresponding to each server in the case of [7].

Acknowledgements

The first author thanks Kazukuni Kobara for clarifying issues related to the LR-AKE scheme and its notion of security. We also thank the anonymous referees for their feedback.

References

1. M. Bellare, D. Pointcheval and P. Rogaway. Authenticated Key Exchange Secure against Dictionary Attacks. *EUROCRYPT 2000*, LNCS 1807, 139–155, 2000.
2. R. Canetti and H. Krawczyk, Analysis of Key-Exchange Protocols and Their Use for Building Secure Channels. *EUROCRYPT 2001*, LNCS 2045, 453–474, 2001.
3. K.-K. R. Choo, C. Boyd and Y. Hitchcock, Examining Indistinguishability-Based Proof Models for Key Establishment Protocols. *ASIACRYPT 2005*, LNCS 3788, 585–604, 2005.
4. H. Fathi, S.-H. Shin, K. Kobara, S.S. Chakraborty, H. Imai and R. Prasad. Leakage-Resilient Security Architecture for Mobile IPv6 in Wireless Overlay Networks. *IEEE Journal on Selected Areas in Communications*, 23(11) 2182–2193, 2005.
5. G. Itkis, L. Reyzin. SiBIR: Signer-Base Intrusion-Resilient Signatures. *CRYPTO 2002*, LNCS 2442, 499–514, 2002.
6. N. Memon and P.W. Wong. A Buyer-Seller Watermarking Protocol. *IEEE Trans. on Image Processing*, 10(4), 2001.
7. S.-H. Shin, K. Kobara and H. Imai. Leakage-Resilient Authenticated Key Establishment Protocols. *ASIACRYPT 2003*, LNCS 2894, 155–172, 2003.
8. S.-H. Shin, K. Kobara and H. Imai. A Simplified Leakage-Resilient Authenticated Key Establishment Protocol with Optimal Memory Size. *ICN(2) 2005*, LNCS 3421, 2005.
9. S.-H. Shin, K. Kobara and H. Imai. A Simple Leakage-Resilient Authenticated Key Establishment Protocol, Its Extensions, and Applications. *IEICE Transactions on Fundamentals of Electronics, Communications and Computer Sciences*, E88-A(3) 736–754, 2005.

10. S.-H. Shin, K. Kobara and H. Imai. Efficient and Leakage-Resilient Authenticated Key Transport Protocol Based on RSA. *ACNS 2005*, LNCS 3531, 269–284, 2005.

A Revised Security Proof

We now provide a sketch of the revised security proof demonstrating that the LR-AKE protocol provides both key establishment and mutual authentication.

Theorem 2 *The LR-AKE protocol is a secure mutual authentication and key establishment (MAKE) protocol if the underlying message authentication (MAC) scheme is secure in the sense of existential unforgeability under adaptive chosen message attack assuming the intractability of the DDH Problem.*

Proof. The proof for **key establishment** generally follows that of Shin *et al.* [9, Theorem 2]. We construct a forger $\mathcal{F}$ against the MAC, using an adversary $\mathcal{A}$ against the protocol. $\mathcal{F}$ now simulates the view of $\mathcal{A}$ in the game simulation. $\mathcal{F}$ answers all Send, Execute, Reveal, LeakC, LeakS, and Corrupt queries similar to the proof simulation of [9, Theorem 2]. At some stage of the game simulation, $\mathcal{A}$ decides to choose a session to be tested and asks the Test query, which is answered by $\mathcal{F}$ in almost the same fashion as the proof simulation presented in the proof for [9, Theorem 2]. Hence, whatever $\mathcal{F}$ can simulate in the proof for [9, Theorem 2], $\mathcal{F}$ can do the same here. After asking the Test query, $\mathcal{A}$ is allowed to further interact with the protocols by asking any Send, Execute, Reveal, LeakC, LeakS, and Corrupt queries of choice, with the exception that $\mathcal{A}$ is not allowed to trivially expose the Test session by asking any Reveal, LeakC, LeakS, or Corrupt queries to the partner or owner associated with the Test session. Eventually, $\mathcal{A}$ outputs the guess bit, b'.

It follows that whatever the MAC forger can simulate in the proof for [9, Theorem 2], our $\mathcal{F}$ can do the same although the converse is not true. Recall that the $SIDs$ of A and B for the protocol described in Fig. 1 are defined to be $y_1||y_2||v_1||v_2$. Let Repeat be the event that a value of SID repeats at some point during the game simulation. It is easy to see that the probability of Repeat happening occurs with probability upper bounded by $\frac{q_s^2}{2^k}$ (where G and q_s is the upper bound on the number of the sessions in the game simulation and k is the security parameter) by a "birthday problem" calculation. Let the advantage of $\mathcal{A}$ in our game simulation be denoted by $\mathsf{Adv}^{\mathcal{A}}(k)$ and the advantage of $\mathcal{A}$ in the game simulation of [9, Theorem 2] be denoted by $\mathsf{Adv}^{\mathcal{A}}{}_{[9]}(k)$. It then follows easily that $\mathsf{Adv}^{\mathcal{A}}(k) \leq \mathsf{Adv}^{\mathcal{A}}{}_{[42]}(k) + \frac{q_s^2}{2^k}$. □

We now prove that the LR-AKE protocol achieves **mutual authentication**. We define as $\mathsf{Event}^{\mathsf{No-Matching}}$ the event that a fresh oracle, Π_U^i, who has been engaged in a conversation and has successfully finished the protocol with a session key output but without a partner oracle [1].

Lemma 1 *The LR-AKE protocol of Shin* et al. *[7] described in Fig. 1 achieves mutual authentication if* $\mathtt{Pr}[\mathsf{Event}^{\mathsf{No-Matching}}]$ *is negligible.*

Proof. Assume that an adversary can violate the mutual authentication with probability ϵ within a time bound t. Similar to our earlier proof, we construct a MAC forger, $\mathcal{F}$, using such an adversary, $\mathcal{A}$. $\mathcal{F}$ now simulates the view of such an adversary, $\mathcal{A}$, in similar fashion as the earlier game simulation.

We consider the probability that $\mathcal{F}$ does not abort the simulation, which can happen under any of the scenarios: (1) abort when being asked some Reveal queries (2) abort when being asked some LeakC queries (3) abort when being asked some LeakS queries (4) abort when being asked some $\mathsf{Corrupt}$ queries. Let q_N be the maximum number of sessions between any two parties in the protocol run and q_P be the maximum number of players in the protocol run.

The probability that $\mathcal{F}$ does not abort for scenarios (1) to (3) are $(q_P^2 q_N - 2)/(q_P^2 q_N)$ respectively, and for (4) is $(q_P - 2)/(q_P^2)$. One may further remark that the simulation is perfectly indistinguishable from a real game, except for a negligible probability. The probability for an oracle to have many partners is bounded by q_P^2/q_N. Therefore, if $\mathcal{F}$ is successful during the simulation (the probability is at least ϵ), then there is a completed/accepted oracle Π_U^i such that Π_U^i has no matching oracle. Since there are at most $q_P^2 q_N$ oracles during the simulation, the probability for this oracle to be the oracle, $\Pi_{U_0}^i$, is $1/(q_P^2 q_N)$. Therefore, the advantage of $\mathcal{F}$ is at least $\epsilon(2^{2^k} - 1)(q_P - 2)((q_P^2 q_N - 2)/(q_P^2 q_N))^3(1/(q_P^7 q_N^3 2^{2^k}))$. However, we know that both q_N and q_P are polynomial in the security parameter k. Hence, the probability of $\mathtt{Pr}[\mathsf{Event}^{\mathsf{No-Matching}}]$ is negligible if the MAC is secure.

□