Analysis of Symmetric Key Authenticated Key Exchange Protocols

Khaishagi, Mohammad Zaid and Kim, Anderson Kunho

Georgia Institute of Technology

Abstract. Our work focuses on analyzing different symmetric key Authenticated Key Exchange (AKE) protocols. These include protocols that have linear and non-linear key evolution. The property of Synchronization Robustness is also one of the key focuses of our work. This property states that two parties should be able to synchronize their counters and produce the same session keys; and has two versions: weak Synchronization Robustness and full Synchronization Robustness.
We provide two variations of existing protocols that achieve the same weak Synchronization Robustness property. One of these variations, LP3-NC, achieves the weak Synchronization Property in a more robust manner as compared to one of the protocols that it is based on. The other variation, LP2-FI, achieves a minor efficiency advantage. We give the outline for these protocols and also give informal proofs for their security properties.
We also outline an integration of these protocols with TLS 1.3 Handshake in Full 1-RTT mode and discuss the practicality of this integration in a closed setting with low-performance devices in a small scale network where the devices remain in the network over a long period of time, such as IoT devices. A brief discussion on the benefits of such an integration in setting up pre-shared key material that is required for the symmetric key AKE protocols is also given.

1 Introduction

1.1 Defining Authenticated Key Exchange (AKE)

Authenticated key exchange (AKE) is a method in which two parties can authenticate each other and agree on some shared session key once authenticated. Using a digital signature scheme in addition to a secret sharing protocol such as Diffie Hellman Ephemeral (DHE) provides several attractive security properties such as perfect forward secrecy [2]. However, an AKE protocol can also use symmetric key cryptography to share secrets. Such AKE protocols may be better suited for low-computation devices such as IoT since they can utilize the efficiency advantage of symmetric key protocols over asymmetric counterparts [4].

Once the parties are authenticated and session keys (secrets) are shared, the two parties can continue communicating. As it has already been portrayed, there are multiple aspects of AKE that can be customized and changed to fit efficiency and security needs.

1.2 The AKE Problem

The previous section highlights why the AKE problem is interesting. Previous literature has shown that asymmetric AKE protocols usually achieve desirable security properties such as perfect forward secrecy [2]. On the other hand, symmetric AKE protocols tend to be more efficient due to the nature of symmetric-key cryptography. However, researchers question whether it's possible to achieve stronger levels of security (e.g. perfect forward secrecy and post-quantum security) using the more efficient symmetric key primitives.

1.3 Papers Covered

There are three main papers we cover. The main paper is "Symmetric Key Exchange with Full Forward Security and Robust Synchronization" by Boyd et al. [4]. The majority of our contributions stem from this paper. The other papers we cover are "Symmetric-Key Authenticated Key Exchange (SAKE) with Perfect Forward Secrecy" by Avoine et al. [2]. and "A Cryptographic Analysis of the TLS 1.3 Handshake Protocol" by Dowling et al. [5].

We don't provide a separate preliminaries section. Instead, we are assuming that the reader has an understanding of the papers described in this section before reading our paper. Some of the terms we use are wSR (weak Synchronization Robustness), FS (Forward Security), AKE-M (Authenticated Key Exchange with Mutual authentication), Bd. Gap (Bounded Gap) and CC (Concurrent Correctness).

1.4 Outline of our Paper

We have given a basic idea of AKE protocols and the many potential research problems that stem from them. The rest of the paper will be as follows. In **Section 2**, we will introduce SAKE and SR AKE and compare them. In **Section 3**, we will present our contributions: variations on some of the linear key evolution AKE protocols proposed by Boyd et al. [4], as well as a suggested integration with TLS 1.3 Handshake. In **Section 4**, we will point out potential avenues and paths for future work. In **Section 5**, we will summarize our paper. And in **Section 6** we will self-assess our contributions in this paper and explain their significance.

2 Symmetric Authenticated Key Exchange Protocols

In this section we'll briefly describe and review the symmetric-key authenticated key exchange protocols that we're interested in analyzing and comparing.

2.1 SAKE and SAKE-AM

SAKE stands for Symmetric-key Authenticated Key Exchange and was a protocol designed by Avoine et al. [2]. SAKE-AM stands for SAKE aggressive mode. For the sake of comparison, we will only be focusing on SAKE. We will now present a high level overview of the SAKE protocol.

Let σ_{AB} be the gap between A and B. In SAKE $\sigma_{AB} \in \{-1, 0, 1\}$ which means that A can only be one step ahead/behind or synchronized with B. A uses the three keys it stores K'_j, K'_{j-1}, K'_{j+1} to calculate σ_{AB}. A can do this by computing the message m_B which is sent from B to indicate the current state. After calculating σ_{AB}, A sends a bit ϵ to B to indicate the current state: $\sigma_{AB} \in \{-1, 0\} \implies \epsilon = 0$ and $\sigma_{AB} = 1 \implies \epsilon = 1$. A and B perform different actions depending on the current state:

(1) $\sigma_{AB} = 0$: A and B are synchronized. We can consider this a "normal run" in which A computes a new session key sk using a PRF in addition to updating its master keys K'_j, K'_{j-1}, K'_{j+1}.

(2) $\sigma_{AB} = -1$: A is behind B. A has to resynchronize with B by updating its master keys and then performing the "normal protocol run." A sends m_a to B, who continues the normal protocol run on its end.

(3) $\sigma_{AB} = 1$: A is ahead of B. A waits for B to resyncrhonize. B updates its master keys and continues the "normal run." Once A confirms that it is synchronized with B, A continues the "normal run."

In a successful run of SAKE, both parties will be synchronized, will have updated their master keys, and will share a session key sk. As mentioned above, sk is derived from a PRF with the master key as the key value and pseudorandom values r_a and r_b which are used by A and B to authenticate each other:
$sk \Leftarrow KDF(K_{master}, f(r_a, r_b)) \cdot f(r_a, r_b)$ [2]

2.2 SR protocols

The SR protocols are a set of five protocols designed by Boyd et al. The protocols are as follows: LP3, LP2, LP1, PP2, and PP1. The five protocols are split into protocols with linear key evolution (LP#) and with non-linear key evolution (PP#).

In our paper, we focus more specifically on LP3 and LP2. A review of LP3 and LP2 is given in **Section 3.1**.

2.3 Comparing SAKE and the SR protocols

SAKE and SR protocols are similar in several ways. They both claim to achieve forward secrecy. Additionally, both SAKE and SR protocols are based on many of the same assumptions and building blocks. Both use symmetric keys as the building block for their AKE protocols and MACs for party authentication. Lastly, SAKE also uses linear key evolution, similar to the LP protocols.

The two family of protocols are also different in many ways. SAKE is a five message protocol while SAKE-AM is a four message protocol. LP3 is a three message protocol, LP2 is a two message protocol, and so on. It is already apparent that the SR protocols have a higher likelihood of being more efficient than the SAKE protocols, due to the use of less messages.

SAKE was designed specifically with the fact that it wouldn't be able to handle concurrent sessions in mind, while the SR protocols were the opposite: they were designed to specifically work with concurrent sessions. Another property that the SR protocols were designed with in mind was the "bounded gap for concurrent sessions" property. The bounded gap σ_{AB} is only allowed to be $\sigma_{AB} \in \{-1, 0, 1\}$ in SAKE while in the SR protocols it is allowed to be $-C \leq \sigma_{AB} \leq 1 + C$, where C is number of concurrent sessions. Furthermore, SAKE and SAKE-AM update two keys in their protocols while the SR protocols all update only one key. Lastly, two of the SR protocols, PP2 and PP1, take advantage of non-linear key evolution by using **puncturable PRFs**. [4] [2]

3 Contributions

This section describes two variants that we propose for LP3 and LP2 protocols given by Boyd et al. [4]. These variants make some changes to the original key exchange protocols which provide additional benefits such as reduced message complexity and minor improvements to computational efficiency. These are important in the context of the devices targeted by these protocols—low power devices such as IoT devices. We give informal proofs for the variants about their wSR property as well as other properties, as relevant.

3.1 Review of LP3 and LP2

First, we give a short review of the protocols we are discussing from the paper by Boyd et al. [4]. The protocols that we are discussing in this section are the LP2 and LP3 protocols. The LP3 protocol uses 3 messages between the initiator and the responder to provide FS, AKE-M, wSR, and Bd. Gap properties. It can provide wSR even without MAC security guarantee, but fails in Bd. Gap. It also allows for interchangeable roles. LP2 provides FS, AKE-M and wSR properties. The LP2 protocol is an optimization for the LP3 protocol by giving fixed roles to each party while still achieving synchronization robustness between the two parties for the sake of reducing message complexity to 2 messages. It does technically allow for either party to be the initiator using a 'duplex' mode, but this is essentially having two parallel and opposite unidirectional implementations of LP2. The LP2 and LP3 protocols are shown in Figures 1 and 2.

One of the key points of the LP2 protocol is that it uses a `NextOdd` function in order to prevent an adversary from being able to desynchronize the session keys between the two parties using the session oracles provided to the adversary

```
Initiator                                                   Responder
$\mathsf{CTR}_{AB}, \mathrm{k}^{\mathsf{CTR}}_{AB}, \mathsf{K}^{\mathsf{MAC}}$                $\mathsf{CTR}_{BA}, \mathrm{k}^{\mathsf{CTR}}_{BA}, \mathsf{K}^{\mathsf{MAC}}$
.....................................................................................
$z_0 \leftarrow \mathtt{NextOdd}(\mathsf{CTR}_{AB}) - \mathsf{CTR}_{AB}$
$\mathrm{k}^{\mathsf{CTR}}_{AB}, \mathsf{CTR}_{AB} := \mathsf{Advnc}(\mathrm{k}^{\mathsf{CTR}}_{AB}, \mathsf{CTR}_{AB}, z_0)$
$\sigma_1 \leftarrow \mathsf{Mac}(\mathsf{K}^{\mathsf{MAC}}, A \parallel \mathsf{CTR}_{AB})$
                         $\xrightarrow{\mathsf{CTR}_{AB}, \sigma_1}$
                                                            if $\mathsf{Vrfy}(\mathsf{K}^{\mathsf{MAC}}, A \parallel \mathsf{CTR}_{AB}, \sigma_1) = 0$
                                                              Abort
                                                            $z_1 \leftarrow \mathsf{CTR}_{AB} - \mathsf{CTR}_{BA}$
                                                            if $z_1 < 0$
                                                              Abort
                                                            $\mathrm{k}^{\mathsf{CTR}}_{BA}, \mathsf{CTR}_{BA} \leftarrow \mathsf{Advnc}(\mathrm{k}^{\mathsf{CTR}}_{BA}, \mathsf{CTR}_{BA}, z_1)$
                         $\xleftarrow{\mathsf{CTR}_{BA}, \sigma_2}$
                                                            $\sigma_2 \leftarrow \mathsf{Mac}(\mathsf{K}^{\mathsf{MAC}}, B \parallel \mathsf{CTR}_{BA})$
if $\mathsf{Vrfy}(\mathsf{K}^{\mathsf{MAC}}, B \parallel \mathsf{CTR}_{BA}, \sigma_2) = 0$       $sk_{AB} \leftarrow \mathsf{KDF}(\mathrm{k}^{\mathsf{CTR}}_{BA}, \texttt{"der"})$
  Abort                                                     $\mathrm{k}^{\mathsf{CTR}}_{BA}, \mathsf{CTR}_{BA} \leftarrow \mathsf{Advnc}(\mathrm{k}^{\mathsf{CTR}}_{BA}, \mathsf{CTR}_{BA}, 1)$
$z_2 \leftarrow \mathsf{CTR}_{BA} - \mathsf{CTR}_{AB}$
if $z_2 \neq 0$
  Abort
$sk_{AB} \leftarrow \mathsf{KDF}(\mathrm{k}^{\mathsf{CTR}}_{AB}, \texttt{"der"})$
$\mathrm{k}^{\mathsf{CTR}}_{AB}, \mathsf{CTR}_{AB} \leftarrow \mathsf{Advnc}(\mathrm{k}^{\mathsf{CTR}}_{AB}, \mathsf{CTR}_{AB}, 1)$
```

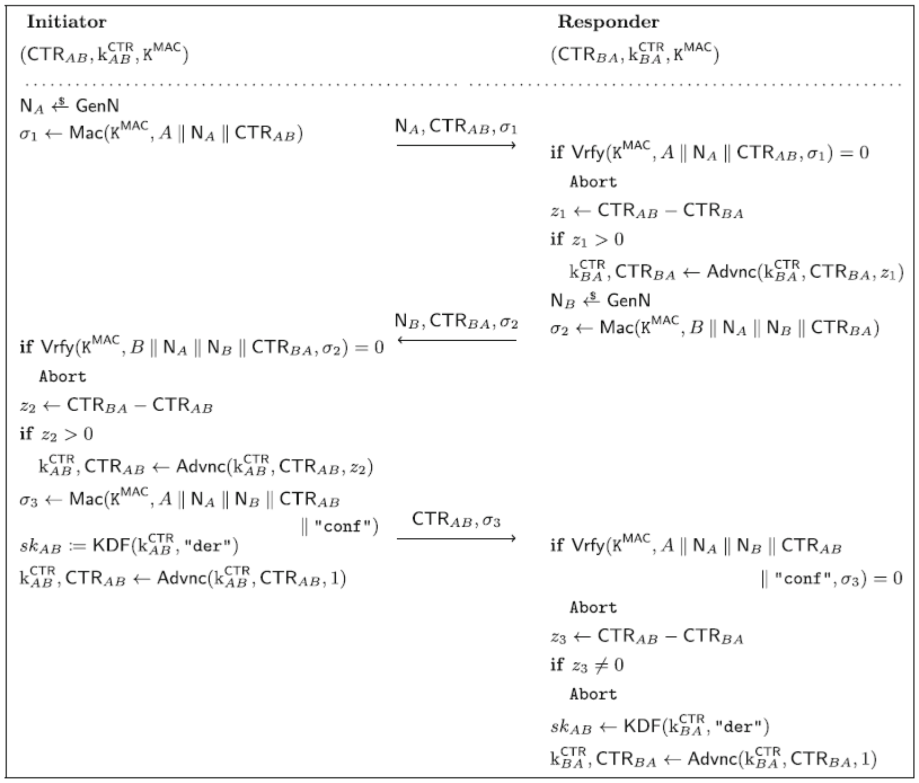
Fig. 1. LP2

in the model that Boyd et al. [4] use to capture multiple concurrent sessions at each party. This `NextOdd` function is intended to serve as a measure to make sure that the initiator always stays ahead of the responder in terms of the key progression and CTR_{AB}. If this was omitted, then it would be possible for the attacker to terminate sessions before msg 2 was received so that the initiator gets ahead of the responder; this could be done over multiple concurrent sessions to desynchronize by arbitrary gap. This would help the adversary win the wSR game in the honest run because the initiator always remains behind the responder with no way to resynchronize over the course of the honest run using only the 2 messages, according to the algorithms described for each party.

3.2 Notations used

We will be referring to the messages 'msg N' where N is the message number in the protocol. So, for example, 'msg 1' refers to the first message delivered between A and B.

We will use the (i, j) notation to denote the protocol state where the latest message received by the initiator (party A) is msg i, and correspondingly msg j for the responder (party B). For example (0,0) denotes A received no messages, and B has received no messages; while (2,1) denotes A received message 2 and



```
Initiator                                                   Responder
$(\mathsf{CTR}_{AB}, \mathsf{k}^{\mathsf{CTR}}_{AB}, \mathsf{K}^{\mathsf{MAC}})$        $(\mathsf{CTR}_{BA}, \mathsf{k}^{\mathsf{CTR}}_{BA}, \mathsf{K}^{\mathsf{MAC}})$
..........................................................................................
$\mathsf{N}_A \xleftarrow{\$} \mathsf{GenN}$
$\sigma_1 \leftarrow \mathsf{Mac}(\mathsf{K}^{\mathsf{MAC}}, A \parallel \mathsf{N}_A \parallel \mathsf{CTR}_{AB})$
                        $\xrightarrow{\mathsf{N}_A, \mathsf{CTR}_{AB}, \sigma_1}$
                                                            if $\mathsf{Vrfy}(\mathsf{K}^{\mathsf{MAC}}, A \parallel \mathsf{N}_A \parallel \mathsf{CTR}_{AB}, \sigma_1) = 0$
                                                              Abort
                                                            $z_1 \leftarrow \mathsf{CTR}_{AB} - \mathsf{CTR}_{BA}$
                                                            if $z_1 > 0$
                                                              $\mathsf{k}^{\mathsf{CTR}}_{BA}, \mathsf{CTR}_{BA} \leftarrow \mathsf{Advnc}(\mathsf{k}^{\mathsf{CTR}}_{BA}, \mathsf{CTR}_{BA}, z_1)$
                                                            $\mathsf{N}_B \xleftarrow{\$} \mathsf{GenN}$
                                                            $\sigma_2 \leftarrow \mathsf{Mac}(\mathsf{K}^{\mathsf{MAC}}, B \parallel \mathsf{N}_A \parallel \mathsf{N}_B \parallel \mathsf{CTR}_{BA})$
                        $\xleftarrow{\mathsf{N}_B, \mathsf{CTR}_{BA}, \sigma_2}$
if $\mathsf{Vrfy}(\mathsf{K}^{\mathsf{MAC}}, B \parallel \mathsf{N}_A \parallel \mathsf{N}_B \parallel \mathsf{CTR}_{BA}, \sigma_2) = 0$
  Abort
$z_2 \leftarrow \mathsf{CTR}_{BA} - \mathsf{CTR}_{AB}$
if $z_2 > 0$
  $\mathsf{k}^{\mathsf{CTR}}_{AB}, \mathsf{CTR}_{AB} \leftarrow \mathsf{Advnc}(\mathsf{k}^{\mathsf{CTR}}_{AB}, \mathsf{CTR}_{AB}, z_2)$
$\sigma_3 \leftarrow \mathsf{Mac}(\mathsf{K}^{\mathsf{MAC}}, A \parallel \mathsf{N}_A \parallel \mathsf{N}_B \parallel \mathsf{CTR}_{AB} \parallel \texttt{"conf"})$
$sk_{AB} := \mathsf{KDF}(\mathsf{k}^{\mathsf{CTR}}_{AB}, \texttt{"der"})$
$\mathsf{k}^{\mathsf{CTR}}_{AB}, \mathsf{CTR}_{AB} \leftarrow \mathsf{Advnc}(\mathsf{k}^{\mathsf{CTR}}_{AB}, \mathsf{CTR}_{AB}, 1)$
                        $\xrightarrow{\mathsf{CTR}_{AB}, \sigma_3}$
                                                            if $\mathsf{Vrfy}(\mathsf{K}^{\mathsf{MAC}}, A \parallel \mathsf{N}_A \parallel \mathsf{N}_B \parallel \mathsf{CTR}_{AB} \parallel \texttt{"conf"}, \sigma_3) = 0$
                                                              Abort
                                                            $z_3 \leftarrow \mathsf{CTR}_{AB} - \mathsf{CTR}_{BA}$
                                                            if $z_3 \neq 0$
                                                              Abort
                                                            $sk_{AB} \leftarrow \mathsf{KDF}(\mathsf{k}^{\mathsf{CTR}}_{BA}, \texttt{"der"})$
                                                            $\mathsf{k}^{\mathsf{CTR}}_{BA}, \mathsf{CTR}_{BA} \leftarrow \mathsf{Advnc}(\mathsf{k}^{\mathsf{CTR}}_{BA}, \mathsf{CTR}_{BA}, 1)$
```

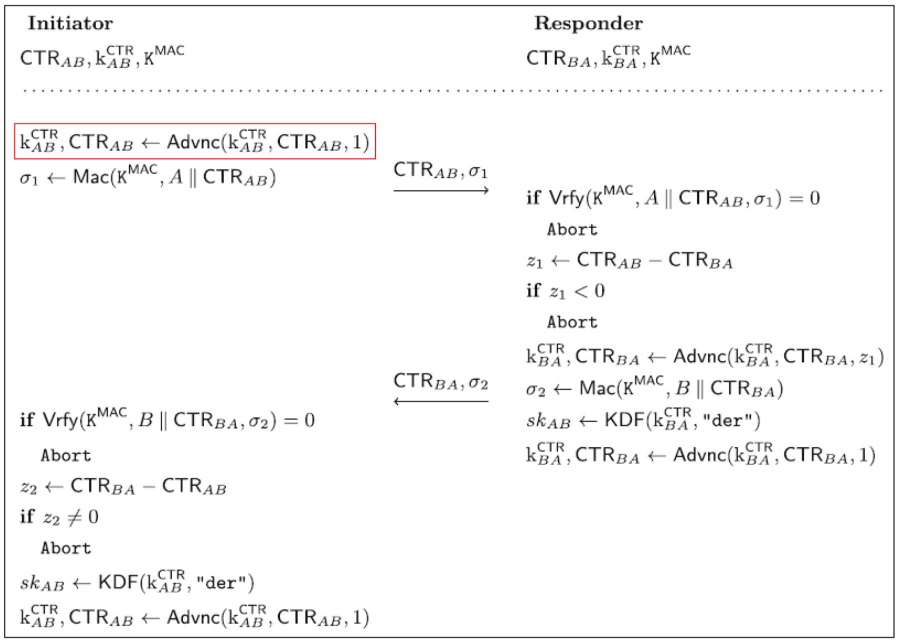
Fig. 2. LP3

B received message 1, which is the final state for LP2 where all messages are delivered.

3.3 Variation on LP2: Fixed Increment (LP2-FI)

In this variation, we modify the LP2 protocol to use a fixed increment in place of the `NextOdd` function. The rest of the protocol remains the same. This would violate the $CTR_{AB} \geq CTR_{BA}$ condition that is described as an invariable for LP2, but we think that this variation would work regardless of that and achieve the same wSR security. This variation is shown in Figure 3.

The $CTR_{AB} \geq CTR_{BA}$ invariability condition is violated when msg 1 is delivered. The party A (initiator) increments the value of CTR_{AB} before sending msg 1. When B receives this, it advances its CTR_{BA} to match A. However, it will send msg 2 and also perform another `Advnc` operation so that it is now one step ahead of A. The adversary can choose to drop this message and terminate the session. Doing so would place A behind B by 1 `Advnc` operation. Thus, the


Initiator | **Responder**

Initiator: $\mathsf{CTR}_{AB}, \mathsf{k}^{\mathsf{CTR}}_{AB}, \mathsf{K}^{\mathsf{MAC}}$

Responder: $\mathsf{CTR}_{BA}, \mathsf{k}^{\mathsf{CTR}}_{BA}, \mathsf{K}^{\mathsf{MAC}}$

Initiator:

$\boxed{\mathsf{k}^{\mathsf{CTR}}_{AB}, \mathsf{CTR}_{AB} \leftarrow \mathsf{Advnc}(\mathsf{k}^{\mathsf{CTR}}_{AB}, \mathsf{CTR}_{AB}, 1)}$

$\sigma_1 \leftarrow \mathsf{Mac}(\mathsf{K}^{\mathsf{MAC}}, A \parallel \mathsf{CTR}_{AB})$

$\xrightarrow{\mathsf{CTR}_{AB}, \sigma_1}$

Responder:

```
if $\mathsf{Vrfy}(\mathsf{K}^{\mathsf{MAC}}, A \parallel \mathsf{CTR}_{AB}, \sigma_1) = 0$
  Abort
$z_1 \leftarrow \mathsf{CTR}_{AB} - \mathsf{CTR}_{BA}$
if $z_1 < 0$
  Abort
$\mathsf{k}^{\mathsf{CTR}}_{BA}, \mathsf{CTR}_{BA} \leftarrow \mathsf{Advnc}(\mathsf{k}^{\mathsf{CTR}}_{BA}, \mathsf{CTR}_{BA}, z_1)$
$\sigma_2 \leftarrow \mathsf{Mac}(\mathsf{K}^{\mathsf{MAC}}, B \parallel \mathsf{CTR}_{BA})$
$sk_{AB} \leftarrow \mathsf{KDF}(\mathsf{k}^{\mathsf{CTR}}_{BA}, \texttt{"der"})$
$\mathsf{k}^{\mathsf{CTR}}_{BA}, \mathsf{CTR}_{BA} \leftarrow \mathsf{Advnc}(\mathsf{k}^{\mathsf{CTR}}_{BA}, \mathsf{CTR}_{BA}, 1)$
```

$\xleftarrow{\mathsf{CTR}_{BA}, \sigma_2}$

Initiator:

```
if $\mathsf{Vrfy}(\mathsf{K}^{\mathsf{MAC}}, B \parallel \mathsf{CTR}_{BA}, \sigma_2) = 0$
  Abort
$z_2 \leftarrow \mathsf{CTR}_{BA} - \mathsf{CTR}_{AB}$
if $z_2 \neq 0$
  Abort
$sk_{AB} \leftarrow \mathsf{KDF}(\mathsf{k}^{\mathsf{CTR}}_{AB}, \texttt{"der"})$
$\mathsf{k}^{\mathsf{CTR}}_{AB}, \mathsf{CTR}_{AB} \leftarrow \mathsf{Advnc}(\mathsf{k}^{\mathsf{CTR}}_{AB}, \mathsf{CTR}_{AB}, 1)$
```

Fig. 3. LP2-FI

invariability condition is broken. It should be noted that this condition is still used as a check at B after receiving msg 1. Boyd et al. [4] require this invariant to hold at all times, even at A before sending msg 1 and even when sessions are terminated. This is the condition that is broken with our modification. However, as we show, it still achieves the same guarantees.

Next we will discuss how the wSR property still holds even though the invariability condition is broken as discussed above, by discussing how the protocol progresses from different starting synchronization gaps. In the protocol, we assume that the two parties start synchronized, i.e., neither is ahead of the other in terms of key advancement. In this case, A advances by 1 at the beginning and the protocol runs as normal with the 2 parties remaining synchronized. It follows the same reasoning and steps as LP2. The second case is when A is ahead of B by arbitrary steps at the start of the honest run. In this case, when the attacker runs the honest session to win the wSR game, upon receiving msg 1 B will calculate the number of steps that it is behind by and then advance to meet A. This will readjust the protocol so that A and B are now synchronized and the protocol will run as normal, as it does in LP2, to complete the Key Exchange. The next case is when A is behind B by 1 step. We will show later that A can be behind B by at most 1 step. In an honest run with this starting state, the two parties are synchronized by the end of the run and generate the same session keys. When msg 1 is sent, A will advance by 1 step before sending, because of the fixed increment at the start of the protocol. This puts both A and B in synchronization. Next, msg 1 gets delivered but B will not advance to meet A because there is no gap between A and B at this point. B will send msg 2 and then advance by 1 additional step putting it ahead of A again. When A receives the message, it performs some checks and then advances by 1 step. This now puts both A and B in synchronization at the end of the honest run and both parties generate the same session keys. This shows that for all three cases—starting at 0, -1 or +x gap—the two parties can still be synchronized after an honest run.

It remains to be shown that the adversary can only desynchronize A and B with a negative gap (when A is behind B) of -1. To show this, let's consider the adversary's abilities to desynchronize the two parties before starting the honest run, we discuss the following cases. We assume that the two parties start in a synchronized state, which the adversary tries to desynchronize using the capabilities available to it in the model used by Boyd et al. [4]. First, if the adversary starts a new session between two paired oracles and drops msg 1 and terminates, then the adversary is able to get A to advance ahead of B. Using multiple concurrent sessions, it is possible to get A arbitrarily ahead of B in this case. Second, the adversary can allow successful delivery of msg 1 but drop and terminate at msg 2. On the reception of msg 1, B will advance to meet A before sending msg 2. After this, B will then once again advance by 1 step as per the protocol. A will not advance because it never receives msg 2 in this case. This puts B ahead of A by 1 step. Now, there are two cases to consider for capabilities the adversary has for getting B ahead of A by an arbitrary gap when starting with a gap of -1: non-concurrent sessions and concurrent sessions.

In the non-concurrent sessions case, the adversary can have only one active session between two parties P_i and P_j using two paired session oracles. Now the adversary attempts to advance B ahead of A by more than 1 step. If the adversary starts a new session, then A will advance 1 step because of the fixed increment. This puts A and B in synchronization. When B receives msg 1, it will not advance to meet A since the counter difference is 0. B will then send msg 2 and advance another step, putting it ahead of A by 1 step again. If the adversary drops and terminates, the situation remains the same with B ahead of A by 1 and the adversary has not been able to increase the gap. If the adversary allows successful delivery, it follows the full protocol and results in synchronization as discussed above. So, the adversary cannot increase the gap by more than |-1| between A and B using non-concurrent sessions.

Now, we consider concurrent sessions. In this, the adversary is allowed to have multiple active sessions without needing to terminate one before starting another. This permits interleaving of messages from different sessions between the 2 parties. However, the honest run must not have any interleaved messages.

When B receives msg 1, it checks if $z_1 < 0$, meaning that CTR_{AB} cannot be less than CTR_{BA} in msg 1. So, any other concurrent session cannot send msg 1 with $CTR_{AB} < CTR_{BA}$. At (0,0), n concurrent sessions can contribute +n gap between A and B. When any one of these sessions reaches (0,1), it sets the gap to 0 and then B does another `Advnc`, the resulting gap is -1. This means that the already active sessions can no longer send msg 1 without aborting because of the $z_1 < 0$ condition check. This then implies that the other n-1 sessions are no longer useful unless the gap becomes 0 or greater, at which point reaching (0,1) would again regress back to the same blocking condition. This continues until all n sessions have reached (0,1) with the gap remaining at -1. Let us also consider a new n+1th session being started at (0,0). This would then set the gap to 0, because of the initial fixed increment. When this session reaches (0,1) it again sets the gap to 0 and then to -1 with the `Advnc` in the final step at B. Thus, we say that even with concurrent sessions, the gap cannot be less than -1.

We are assuming that the block of operations performed at each party are atomic. This is also an implicit assumption in the SR paper. If it is not, then it opens the possibility of race condition attacks. In short, it is possible to have a race condition where two concurrent sessions reach the last step of B with `Advnc(`k_{BA}^{CTR}`,`CTR_{BA}`,1)` simultaneously and advance twice instead of once. This would put B ahead of A by 2. When the honest run is done, it will abort at (0,1) because of the $z_1 < 0$ check. So, the honest run will abort, leading to the wSR game failing.

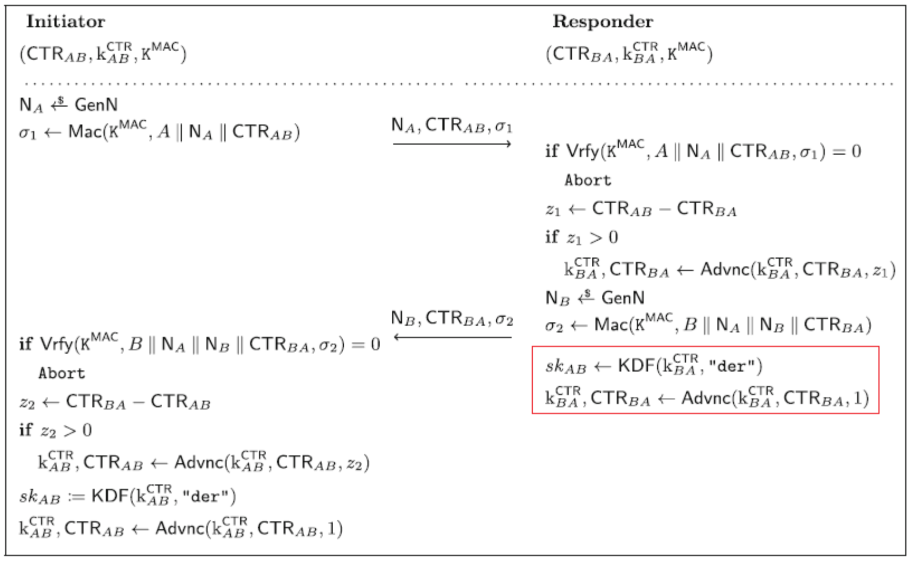
One assumption that is being made here is that MAC security holds. If the adversary is able to forge messages, then it can use this capability to desynchronize A and B by any arbitrary negative gap. This is a fair assumption to make since Boyd et al. prove that MAC security needs to hold for wSR to be satisfied for LP2 [4], and our discussion makes use of this same characteristic.

3.4 Benefits of LP2-FI Variation

LP2-FI allows for omitting any conditional statements as part of `NextOdd` operation. It is known that if-statements and conditional operations are computationally expensive. So, removing this operation makes it more efficient. This is important for low power devices. Otherwise, it should be comparable to LP2 without any significant drawbacks or benefits over it.

3.5 Variation on LP3: No Confirmation Message (LP3-NC)

Our variation on LP3 removes msg 3 from the protocol so that there is no confirmation message sent between A and B; we call this variation LP3-NC. We discuss how this allows for reduced message complexity while providing the same wSR guarantee. We also discuss how, without MAC security, this variation still allows for wSR. This is significant because the LP3 description by Boyd et al. [4] acknowledges that the Bd. Gap property of LP3 is broken when MAC security cannot be assumed. Under this situation, LP3-NC is equivalent to LP3 in both wSR and Bd Gap properties but has the advantage that it uses only 2 messages instead of 3, which is important for low power devices that these protocols are targeting. This is also better than the LP2 protocol which does not give wSR when MAC security is broken.



```
Initiator                                                          Responder
$(\mathsf{CTR}_{AB}, \mathsf{k}^{\mathsf{CTR}}_{AB}, \mathsf{K}^{\mathsf{MAC}})$          $(\mathsf{CTR}_{BA}, \mathsf{k}^{\mathsf{CTR}}_{BA}, \mathsf{K}^{\mathsf{MAC}})$
..............................................................................
$\mathsf{N}_A \xleftarrow{\$} \mathsf{GenN}$
$\sigma_1 \leftarrow \mathsf{Mac}(\mathsf{K}^{\mathsf{MAC}}, A \parallel \mathsf{N}_A \parallel \mathsf{CTR}_{AB})$
                        $\xrightarrow{\mathsf{N}_A, \mathsf{CTR}_{AB}, \sigma_1}$
                                              if $\mathsf{Vrfy}(\mathsf{K}^{\mathsf{MAC}}, A \parallel \mathsf{N}_A \parallel \mathsf{CTR}_{AB}, \sigma_1) = 0$
                                                  Abort
                                              $z_1 \leftarrow \mathsf{CTR}_{AB} - \mathsf{CTR}_{BA}$
                                              if $z_1 > 0$
                                                  $\mathsf{k}^{\mathsf{CTR}}_{BA}, \mathsf{CTR}_{BA} \leftarrow \mathsf{Advnc}(\mathsf{k}^{\mathsf{CTR}}_{BA}, \mathsf{CTR}_{BA}, z_1)$
                                              $\mathsf{N}_B \xleftarrow{\$} \mathsf{GenN}$
                        $\xleftarrow{\mathsf{N}_B, \mathsf{CTR}_{BA}, \sigma_2}$
                                              $\sigma_2 \leftarrow \mathsf{Mac}(\mathsf{K}^{\mathsf{MAC}}, B \parallel \mathsf{N}_A \parallel \mathsf{N}_B \parallel \mathsf{CTR}_{BA})$
if $\mathsf{Vrfy}(\mathsf{K}^{\mathsf{MAC}}, B \parallel \mathsf{N}_A \parallel \mathsf{N}_B \parallel \mathsf{CTR}_{BA}, \sigma_2) = 0$
                                              $sk_{AB} \leftarrow \mathsf{KDF}(\mathsf{k}^{\mathsf{CTR}}_{BA}, \texttt{"der"})$
    Abort
                                              $\mathsf{k}^{\mathsf{CTR}}_{BA}, \mathsf{CTR}_{BA} \leftarrow \mathsf{Advnc}(\mathsf{k}^{\mathsf{CTR}}_{BA}, \mathsf{CTR}_{BA}, 1)$
$z_2 \leftarrow \mathsf{CTR}_{BA} - \mathsf{CTR}_{AB}$
if $z_2 > 0$
    $\mathsf{k}^{\mathsf{CTR}}_{AB}, \mathsf{CTR}_{AB} \leftarrow \mathsf{Advnc}(\mathsf{k}^{\mathsf{CTR}}_{AB}, \mathsf{CTR}_{AB}, z_2)$
$sk_{AB} := \mathsf{KDF}(\mathsf{k}^{\mathsf{CTR}}_{AB}, \texttt{"der"})$
$\mathsf{k}^{\mathsf{CTR}}_{AB}, \mathsf{CTR}_{AB} \leftarrow \mathsf{Advnc}(\mathsf{k}^{\mathsf{CTR}}_{AB}, \mathsf{CTR}_{AB}, 1)$
```

Fig. 4. LP3-NC

The modification that we make does the following: It omits the steps where A sends msg 3 to B. It also omits the operations that B does upon reception of msg 3. It adds a session key derivation and an `Advnc` (by 1 step) operation at the end of the operations that B performs after reception of msg 1. This is easier to conceptualize as taking the session key derivation and `Advnc` operation from the msg 3 reception operations at B and moving these to the end of the msg 1 reception operations, and omitting the sending and receiving of msg 3. This is shown in Figure 4.

To show that this variant satisfies wSR, let's consider the possible cases of desynchronized starting gap between A and B during the honest run. First, if the gap is 0, then the reception of msg 1 at B will put B ahead of A by 1. When A receives msg 2 from B, it does not need to advance to match the counter of B, but it will advance one step after that. So, now A and B are synchronized and derive the same session key.

In the second case, the gap between A and B is +x, meaning that A is ahead of B by x steps in the sense of key advancement and the counter values. When B receives msg 1, it will calculate the counter difference and then advance by that many steps so that its counter is now synchronized with A; it then sends msg 2 and advances one additional step. When A receives msg 2 from B, it does not need to advance because the counter difference is 0, because the counter value sent by B is the value before the final one-step key advancement. It will then advance an additional step so that CTR_{AB} matches CTR_{BA} at the end, and derives the same session key.

The third case is when the gap is -x, meaning that B is ahead of A. When A sends msg 1 to B, B checks the counter difference to find that it does not need to advance to match A. It sends its counter values in msg 2 and then advances by one additional step. When A receives msg 2, it checks the counter difference. It advances by a number of steps equal to this difference, after which it will make one additional `Advnc` operation. This puts them both in synchronization and they derive the same session key.

This shows that wSR can be achieved in this variation while only using 2 messages. It is important to also remember that wSR requires that the adversary does not interfere in the message delivery and protocol execution during the honest run. The cases discussed above, therefore, do not consider cases where an adversary interjects messages or otherwise interferes in the protocol.

We also do not need to separately consider concurrent and non-concurrent sessions because the discussion above shows that wSR can be achieved regardless of starting gap between the two parties. If the adversary can achieve any arbitrary gap between the two parties using concurrent sessions, it would still be corrected during the course of the honest protocol run in the wSR experiment.

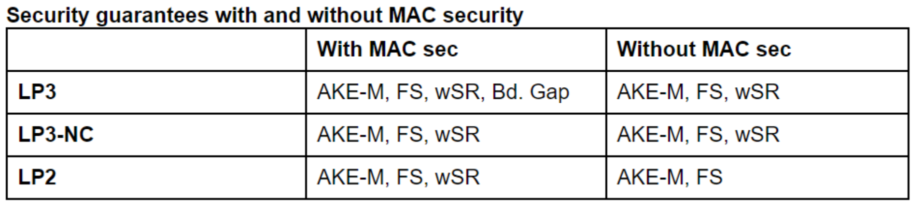
Another important point about the LP3-NC variant is that it achieves wSR without requiring MAC security. This is because the adversary is not allowed to interfere during the honest run so that even with forged messages which could allow the adversary to get either party to advance arbitrarily ahead, it would still be corrected during the honest run as described above. However, it would not provide the Bd. Gap property since if the adversary can forge messages and get either A and B to advance arbitrarily, then there is no bound on the gap between their counters. This is also acknowledged by Boyd et al. in their description of LP3 where they mention that Bd. Gap is not guaranteed for LP3 if MAC security is broken [4].

Bd. Gap is not guaranteed by LP3-NC even with MAC security since in this variant, the adversary can terminate sessions at (0,1) and get B to advance arbitrarily ahead.

3.6 Benefits of LP3-NC

The main benefit of this variant is that it uses only 2 messages as compared to 3 messages in LP3. However, the LP2 protocol also has 2 messages, so it makes sense to compare LP3-NC with LP2 as well as LP3 on their security guarantees.

Under MAC security, LP3-NC provides the same guarantees as LP2: wSR, AKE-M and FS. Without MAC security, however, LP2 does not provide wSR, while LP3-NC does; and LP3-NC becomes equivalent to LP3 since the latter does not guarantee Bd. Gap without MAC security. Also, the security proof for LP2 given by Boyd et al. in their full paper [3] includes a term encapsulating SEUF-CMA-Q security of the chosen MAC scheme. Since MAC security is not required for LP3-NC, based on our informal proofs above, it should have better bounds for wSR security as compared to LP2. We think that it would be equal


Security guarantees with and without MAC security

	With MAC sec	Without MAC sec
LP3	AKE-M, FS, wSR, Bd. Gap	AKE-M, FS, wSR
LP3-NC	AKE-M, FS, wSR	AKE-M, FS, wSR
LP2	AKE-M, FS, wSR	AKE-M, FS

Fig. 5. Security guarantees with and without MAC security

to the Adv^{wSR}_{LP3} which is 0 according to Theorem 15 [4], but we have not proven this.

Another benefit of LP3-NC is that it allows interchangeable roles, similar to LP3. LP2 does not allow for interchangeable roles, except with a 'duplex' mode.

3.7 TLS 1.3 Integration with LP and PP Protocols

The protocols proposed by Boyd et al. [4] are aimed at low performance devices in a closed setting such as wireless battery powered IoT devices. One of the shortcomings of their protocols is that they require pre-shared key material (PSK) on each device. This is why the closed setting is important—it allows for pre-sharing of keys to be done easily and practically without requiring public key infrastructure. However, with things like wireless battery powered IoT devices, there is the issue of introducing new devices to the closed setting. Every new device that is introduced would need to have pre-sharing of keys done with the other devices and this can become troublesome after a few devices. This problem is made more relevant with increased IoT prominence and adoption in everyday usage. To address this, we suggest an integration with TLS 1.3 Handshake protocol for establishing PSK.

To briefly review, TLS 1.3 Handshake has two modes that it can execute: Full 1-RTT mode and PSK mode. The PSK mode also has two additional modes: using ECDHE and without ECDHE. When two hosts communicate for the first time, they must use the Full 1-RTT mode in order to establish keys. During this first handshake, they can also establish PSK keys that can be used for future handshakes using the PSK mode [5]. The PSK mode, however, does make use of public key cryptography; and it is one of the reasons that Boyd et al. mention that their protocols give better performance for low power devices.

Our suggestion is to use the TLS 1.3 Handshake in Full 1-RTT mode with new devices so that they can establish PSKs between them. In TLS 1.3, these would later be used in the PSK mode. However, we suggest that when the two devices need to communicate later, they use the PSKs with the LP and PP protocols. This would essentially be replacing the PSK mode in TLS 1.3.

The benefit of doing this is that it allows for programmatic and dynamic set up of PSKs between the two devices, as opposed to a manufacturer having to install PSKs on all pairs of devices or a user having to manually do it. The only thing that a manufacturer would have to do in this case is to install certificates on these devices. The other benefit is that the first handshake using Full 1-RTT mode is a one-time cost. All key exchange communication following this first handshake would only require symmetric key cryptography. In PSK mode of TLS 1.3 Handshake, there is public key cryptography that is used. So, such an integration only requires public key cryptographic operations in the first handshake and following communications are more efficient owing to strictly symmetric key cryptographic operations in the key exchange.

The one-time cost of using the Full 1-RTT handshake would be problematic if there had been a need to establish PSKs more frequently. However, in a closed environment, such as with IoT, the devices are going to be communicating with only a few other devices that are also part of this small network. Moreover, these devices are also not replaced frequently and are also not discarded quickly. So, it is reasonable to expect that these devices have a long life and the scale of the network is small to the extent that there is no need for frequent first-time handshakes using the Full 1-RTT mode.

So, we make this suggestion of integrating the Full 1-RTT mode as a one-time handshake for the purpose of setting up PSKs on devices when they are introduced to a new closed setting small scale network of low power devices.

4 Future work

One of the avenues of future work is to analyze other symmetric-key AKE protocols such as SAKE* [6], which is a symmetric-key AKE protocol for industrial IoT devices, and SKIA-SH [1] which is a symmetric-key AKE protocol for smart homes.

We also did not cover much of the other protocols proposed by Boyd et al., specifically the ones that use non-linear key evolution by using puncturable PRFs (PPRF). There are different variations of PPRFs that could be investigated. An avenue for future work could certainly be to analyze how PP2 and PP1 would be different (in regards to security properties, efficiency, etc.) if they were to use various different PPRFs.

Additionally, we did not find any cryptographic libraries or tools that offer an implementation of PPRFs. This would certainly be an obstacle for adoption of the non-linear key evolution protocols PP1 and PP2 in practical applications.

Another important thing that we have left for future work is to provide formal proofs for the variant protocols that we have discussed. We have given informal proofs, however, these should be addressed more formally. We have also remarked that we think the advantage of an adversary against the LP3-NC protocol attacking the wSR property would be the same as LP3 but did not prove it. This can also be explored further with a formal analysis and proof.

Similarly, our suggested integration of TLS 1.3 Handshake in Full 1-RTT mode with the LP and PP protocols would also benefit from having a more formal analysis of the security properties that such an integration would provide.

5 Conclusion

We analyzed some specific Symmetric-key AKE protocols. Namely, SAKE/SAKE-AM designed by Avoine et al. [2] and LP3/LP2 designed by Boyd et al. [4]. We discussed how they are quite similar and built on many of the same assumptions (i.e. MAC security, PRF security, etc.), they are also fundamentally different in the security properties they provide.

The main part of our contributions focuses on the protocols by Boyd et al. where we have suggested two variants: LP2-FI for LP2, and LP3-NC for LP3. We give informal proofs showing how LP2-FI achieves similar properties to LP2; and we also informally show the security properties that LP3-NC achieves. We also discuss the properties of LP3-NC under MAC security as well as without it.

We also outline a suggested integration of TLS 1.3 and the LP and PP protocols by Boyd et al. [4]. We discuss how the integration would work and briefly touch on the benefits that this integration might have.

6 Self Assessment

We think that we have done well in analyzing the protocols that we have reviewed from the papers we read.

The highlights of our work are that we have provided a good comparison between SAKE [2] and the SR protocols [4]. The SR protocols build on SAKE, however, Boyd et al. did not give a thorough comparison between them other than to outline the benefits of their own protocols. Our comparison is intended to fill this gap.

Another highlight of our work is that we have provided variations on the LP2 and LP3 protocols as well as given informal proofs showing the security properties of these variations. We have also compared these variation protocols against LP2 and LP3, which they are based on, in regards to the security properties that they provide. The LP3-NC variant is more significant in our opinion because it provides wSR even without MAC security and, as such, has an advantage over LP2. Moreover, it becomes equivalent to LP3 when MAC security is not assumed. It achieves this while only making use of two protocol messages between the initiator and responder. Compared to LP2, it also allows for interchangeable roles without needing a ‘duplex’ mode.

The last contribution that we think is important is the suggested integration of the SR protocols by Boyd et al. [4] with the TLS 1.3 Handshake in Full 1-RTT mode [5]. We have given an outline for this integration and have also provided discussion on the practicality and benefits of it, along with its usefulness in establishing pre-shared key materials that the SR protocols make use of.

In light of these points of significance in our work, we think that we have done well. Regarding the grade, we are hoping for an ‘A’.

References

1. Alzahrani, B.A., Barnawi, A., Albarakati, A., Irshad, A., Khan, M.A., Chaudhry, S.A.: Skia-sh: A symmetric key-based improved lightweight authentication scheme for smart homes. Wireless Communications and Mobile Computing 2022 (2022)
2. Avoine, G., Canard, S., Ferreira, L.: Symmetric-key authenticated key exchange (sake) with perfect forward secrecy. In: Cryptographers' Track at the RSA Conference. pp. 199–224. Springer (2020)
3. Boyd, C., Davies, G.T., de Kock, B., Gellert, K., Jager, T., Millerjord, L.: Symmetric key exchange with full forward security and robust synchronization. Cryptology ePrint Archive (2021)
4. Boyd, C., Davies, G.T., Kock, B.d., Gellert, K., Jager, T., Millerjord, L.: Symmetric key exchange with full forward security and robust synchronization. In: International Conference on the Theory and Application of Cryptology and Information Security. pp. 681–710. Springer (2021)
5. Dowling, B., Fischlin, M., Günther, F., Stebila, D.: A cryptographic analysis of the tls 1.3 handshake protocol candidates. In: Proceedings of the 22nd ACM SIGSAC conference on computer and communications security. pp. 1197–1210 (2015)
6. Fan, Q., Chen, J., Shojafar, M., Kumari, S., He, D.: Sake*: A symmetric authenticated key exchange protocol with perfect forward secrecy for industrial internet of things. IEEE Transactions on Industrial Informatics (2022)