Details of Attacks and Fix for EAP-AKA'

Kelley Burgin (kburgin@mitre.org)
Paul Rowe (prowe@mitre.org)
Current contact:
Joshua Guttman (guttman@mitre.org)
The MITRE Corporation

In this document we provide the details of two privacy attacks on the EAP-AKA' protocol that have been reported in the literature [1, 2]. We present a proposed improvement designed to mitigate these attacks. By demonstrating how the attacks manifest in the Cryptographic Protocol Shapes Analyzer (CPSA) [3], we can then verify that the modified protocol is no longer subject to the attacks.

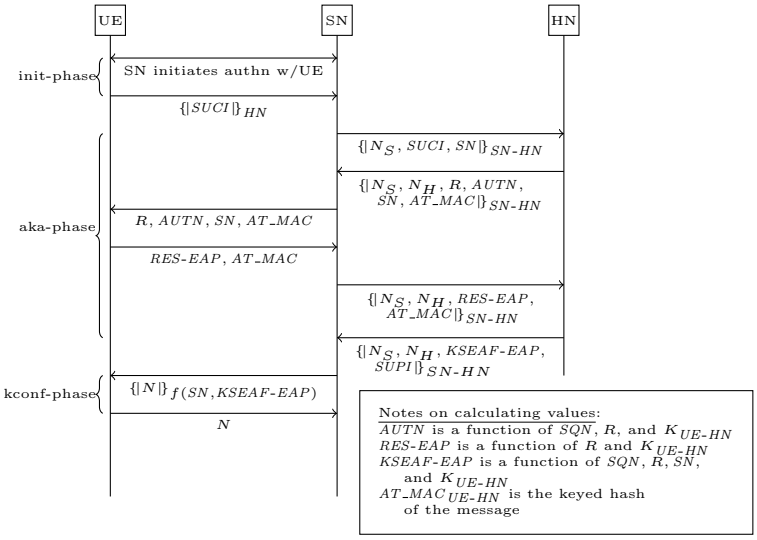

Figure 1: Message diagram of the EAP-AKA' protocol.

Protocol Description

Figure 1 shows the sequences of messages sent by the three parties in EAP-AKA'. In the initialization phase, the user equipment (UE) attempts to connect to a serving network (SN) by sending its identifier (known as a $SUCI$) encrypted for its own home network (HN). The SN forwards this request on a secure channel to the HN who initiates a challenge response protocol with the UE. Since the UE and HN are not within range of each other, the challenge response protocol must be mediated by SN who relays the messages. When the UE receives the challenge message, it will respond in one of several ways

depending on whether certain checks succeed or fail. The AT_MAC is a message authentication code (MAC) computed using a key shared by UE and HN. If the MAC check fails, then the UE will reply with a MAC-FAILURE message (not depicted in Fig. 1). If the MAC check succeeds, but the sequence number embedded in $AUTN$ is incorrect, the UE responds with a SYNC-FAILURE message (also not depicted) that includes enough information for the UE and HN to resynchronize the sequence number. If both checks succeed, the UE replies with an EAP response (denoted $RES\text{-}EAP$ in Fig. 1). The HN will then provide SN with the value $KSEAF\text{-}EAP$ that allows the SN to compute a key it will share with the UE.

Attack Description

To mount a privacy attack, a malicious actor only has to replay the challenge message as it is relayed from the SN to the UE. Since these challenge messages must be sent in the clear, it is easy for a malicious actor to receive and record such messages for later use. Replaying it later is also trivial assuming the adversary has equipment that can spoof a base station. As we explain below, replaying this message later will allow an adversary to determine if the targeted UE is within range of the adversary.

If the replayed message is received by a device that is not the targeted UE, that device will attempt to verify the MAC value by computing its own keyed checksum over the message contents and comparing with AT_MAC. Since the MAC in the replayed message was computed using the key shared only between the targeted UE and the HN, this MAC check will always fail resulting in a MAC-FAILURE message.

However, consider how the targeted device will respond. Since it uses the same key as was used in the AT_MAC of the replayed message, the MAC check will necessarily succeed. However, this message has embedded within it an old sequence number. As long as the UE has advanced its sequence number this will cause a synchronization failure, and it will reply with a SYNC-FAILURE message.

Since the targeted device responds differently from other devices when receiving a replayed message, an adversary can detect the presence of the targeted UE if it receives a SYNC-failure message in response to the replay. This is a violation of privacy for the owner of the UE. It provides a mechanism for an adversary to at least confirm an approximate location of a targeted device (and by proxy, a targeted individual). Worse still, if an adversary has the resources to establish a network of rogue base stations, then a targeted UE can even be tracked, building up a pattern of movement.

We modeled the protocol in CPSA to verify the existence of the attack described above. We had to model the failure modes of user equipment devices to demonstrate the differing response patterns. Figure 2 shows the output of CPSA. Although the two leftmost protocol runs are shown at the same level in the diagram as the successful run of a UE, they could be significantly later, as they are receiving the challenge message replayed from the successful run. While CPSA does not display the messages on its arrows, notice that the leftmost run

is a run of a *UE* whose identity is distince from the targeted *UE*, but the next run to the right is a run of the targeted *UE*.

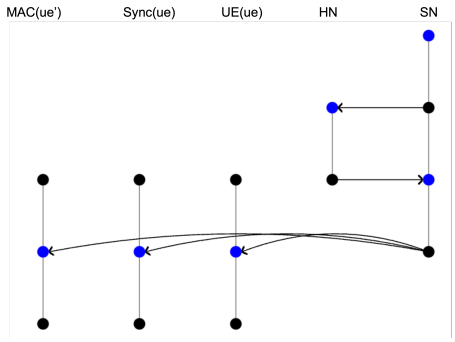

Figure 2: Depiction of privacy attack in CPSA's output.

The replay attack has the additional consequence of allowing the adversary to learn information about how frequently the targeted *UE* has connected. This might reveal usage patterns that should remain private. This is achieved by replaying the *same* challenge message several times to the targeted *UE*. When it replies with a SYNC-FAILURE message, this message contains $SQN \otimes AK$, which is the *UE*'s current sequence number xor'ed with some value derived from the challenge message. When the challenge message is replayed a second time, the *UE* will reply with $SQN' \otimes AK$, which is a new sequence number xor'ed with the *same* AK. The adversary can xor these two values to obtain $SQN \otimes SQN'$. This allows the adversary to infer which bits of the sequence number have changed between the two replays, revealing information about how many times the targeted *UE* has connected in the meantime. Combined with information about where the *UE* is on each successful replay attack, this constitutes a significant privacy violation.

We also used CPSA to verify the existence of this attack. While Fig. 2 shows that two distinct devices respond differently to the same replayed message, Fig. 3 demonstrates that the targeted *UE* always replies with SYNC-FAILURE messages, even upon repeated replayes of the same message. This is what enables the xor attack described above.

Proposed Improvement

The key insight that leads to our proposed improvement is that these attacks would be impossible if the *UE* were unwilling to reply to the challenge unless it was sure the challenge was generated in response to its current connection request. The simplest way to ensure that the *HN*'s challenge message is tied to the initial request is to include a unique number N_U in the *UE*'s initial request. We thus propose to include such a unique value N_U in the initial request and to thread it through all subsequent messages.

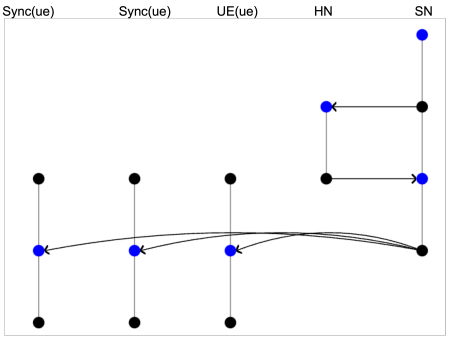

Figure 3: Depiction of xor attack in CPSA's output.

The proposed changes are shown in Fig. 4 where additions and changes are marked in bold text. The values *AUTN*, *RES-EAP*, and *KSEAF-EAP* are computed in essentially the same way as before except that each occurrence of R in the original computation is replaced with the pair R, N_U. When a device receives the challenge message, it should avoid sending any failure messages if the unique value N_U does not match the value it expects, as that is a signal that the message is a replay or is intended for somebody else.

One important feature of this proposed improvement is that the new value N_U need not be random. It suffices to ensure that the *UE* will never send two requests with the same N_U. This is an advantage because it means that this so-

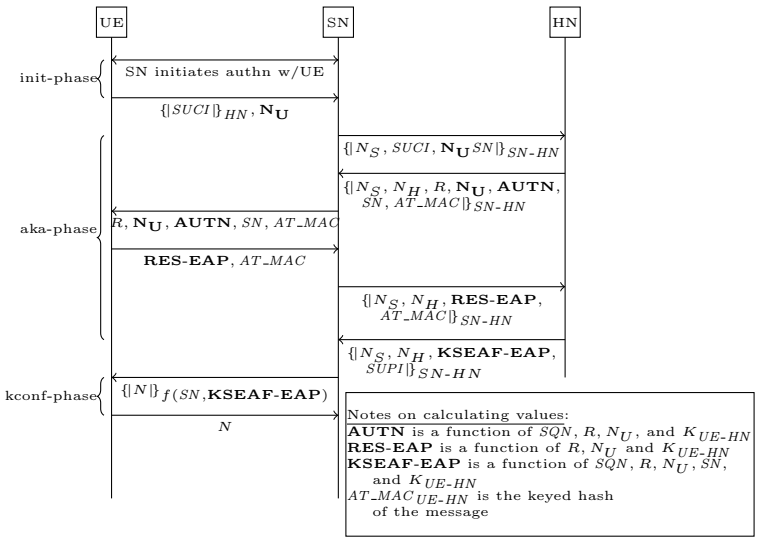

Figure 4: Message diagram of proposed improvement to EAP-AKA' protocol.

lution can also apply to legacy devices that may not be able to reliably produce sufficiently random values. For devices that can generate random values, it is best to choose N_U randomly. For other devices, since using a simple counter would allow somebody to infer usage information and violate the user's privacy, we recommend that the *UE* choose N_U to be the keyed hash of some internal counter value such as $N_U = \#^*(K_{UE\text{-}HN}, ctr)$ where $\#^*(\cdot)$ is a new hash function, initialized so as to be independent from the other hash functions used in the protocol. This will ensure that the N_U values are sufficiently unrelated from one run to the next that an adversary canot infer anything about usage patterns from this value alone.

We modeled this modified version of the protocol in CPSA to determine if the proposed changes would be sufficient to prevent the privacy attacks described above. CPSA allows a user to input a "scenario" and it will show all the ways that scenario might be part of a real execution. When we ask it to show all the ways in which two distinct devices could reply with different error messages in response to the same replayed message, CPSA determined that there were no such executions. In other words, the CPSA analysis shows that the problematic situation depicted in Fig. 2 is not possible (and hence there is no visual output to show in a figure). Similarly, when asked to display all the ways a single device might reply with SYNC-FAILURE messages to the same replayed challenge, CPSA determines that this is impossible. In other words, the problematic situation depicted in Fig. 3 is also cannot occur. We therefore have strong evidence, backed by a formal analysis, that our proposed changes to the protocol are improvements that mitigate the known privacy attacks.

References

[1] David Basin, Jannik Dreier, Lucca Hirschi, Saša Radomirovic, Ralf Sasse, and Vincent Stettler. A formal analysis of 5G authentication. In *Proceedings of the 2018 ACM SIGSAC conference on computer and communications security*, pages 1383–1396, 2018.

[2] Ravishankar Borgaonkar, Lucca Hirschi, Shinjo Park, and Altaf Shaik. New privacy threat on 3G, 4G, and upcoming 5G AKA protocols. *Proceedings on Privacy Enhancing Technologies*, 2019(3):108–127, 2019.

[3] John D Ramsdell, Joshua D Guttman, Moses D Liskov, and Paul D Rowe. The CPSA Specification: A Reduction System for Searching for Shapes in Cryptographic Protocols. Technical report, The MITRE Corporation, 2009.