5G-AKA: A Formal Verification

David Clayton and Ira Harmon

December 11, 2018

Abstract: The 5th generation of cell phone technology is scheduled to be deployed by 2021. It will connect more people around the world than any prior generation. The 5G protocol suite includes modifications of proven protocols as well as new protocols. However, the foundation of 5G security rests upon the 5G-AKA protocol. Since inception, 5G-AKA has gone through multiple revisions due to discovered vulnerabilities. In this paper the most recent version of the protocol is tested using the Tamarin Prover and recommendations are made that would improve its overall security.

1 Introduction

By 2019 over 5 Billion people are expected to own a mobile phone. Currently over 62 percent of the world population uses mobile phones. As cell phones become more pervasive their use touches every aspect of modern life: Facebook updates, news, and banking transactions are all increasingly done via cell. At this critical time in the evolution of cellular technology, 3GPP, the body that standardizes cell phone protocols is preparing to deploy 5G. And while 5G promises to connect more users with better service than previous generations, the unrestrained growth of the technology makes the security implications of 5G critical. 5G-AKA (Authentication and Key Agreement) is the first line of defense in securing mobile communications. The protocol authenticates the user and distributes long term keys. The most recent version of the protocol is outlined in 3GPP Publication TS 33.501 V15.2.0. In this paper we validate the most recent version of the protocol through symbolic analysis.

2 Related Work

The Tamarin-Prover is software for the symbolic analysis and verification of security protocols.

3 Background

3.1 The 5G-AKA Protocol

5G-AKA is similar to authentication protocols used in previous generations. 5G allows for authentication via 5G-AKA or EAP-AKA. For the purposes of this paper, the focus is 5G-AKA. There are at least 4 actors in the 5G authentication process.

1) **UE** (user equipment) - The UE is a mobile device. It is identified by its SUPI (SUbscription Permanent Identifier). The SUPI serves the same purpose as the IMSI in previous generations.

2) **SEAF** (SEcurity Anchor Function) - The SEAF is co-located with the AMF (core Access Management Function). The SEAF creates a key, K_{seaf}, that is used to encrypt all communications during authentication [11]. This key is also used to derive the session key post authentication. The SEAF communicates with the UE via the SUCI (SUbscriper Concealed Identifier). This equivalent to the TMSI in previous generations.

3) **AUSF** (AUthentication Server Function) - The AUSF handles authentication requests for the 3GPP network and non-3GPP network. It informs the UDM of successful and failed authentications.

4) **UDM** (Unified Data Management) - The ARPF (Authentication credential Repository and Processing Function) is co-located with the UDM. The ARPF is located in the home network of the UE. The ARPF stores the long term key of the UE. This is the same key stored on the SIM card of the UE. The ARPF also stores the SUPI associated with each SUCI. It communicates an authentication vector back to the AUSF after receiving an authentication request. Because of its credential store it is the most secured component in the network [4].

Figure 2.1 gives a high level overview of the connections between the four components.

The protocol begins with an authorization request. The request can be initiated by the UE or the SEAF. Authentication can use either 5G-AKA or EAP-AKA protocol. The 5G-AKA protocol provides proof of authentication to the home network while EAP-AKA does not.

Assuming authentication is initiated by the UE, the UE sends a N1 message to the SEAF. The message contains the UE's SUCI or 5G-GUTI. The 5G-GUTI similar to the SUCI is a temporary identifier used by the UE for over air communications. The SEAF sends a Nausf_UEAuthentication_Authenticate Request containing the UE SUCI or SUPI and the serving network name to the AUSF. The AUSF temporarily stores the serving network name, and determines wether the network the request was received from has the right to use that name. If the serving network name and the network the message was received from do not match, the authentication process is halted and a failure message is sent to the serving network. If the SN name and the network the message is received from agree the AUSF passes the information from the SEAF on in an Nudm_UEAuthentication_Get Request to the ARPF. The ARPF chooses whether

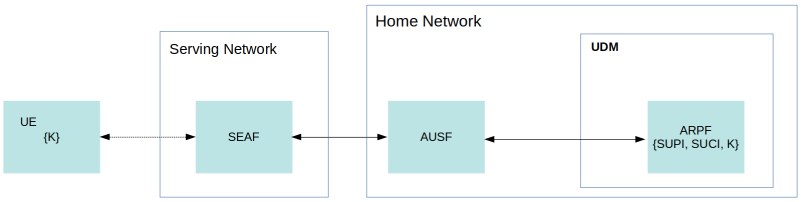

Figure 1: Dashed lines indicate insecure connections.

EAP-AKA or 5G-AKA is used for authentication. It also determines the SUPI of the UE if the SUCI was sent in the message.

Up to this point the process is the same for EAP-AKA and 5G-AKA. Once the determination is made which protocol to use, the two protocols diverge. In 5G-AKA ARPF generates an authentication vector, for each Nudm_UEAuthentication_Get Request received. The authentication vector consists of <RAND, AUTN, XRES*, K_{ausf}>. The AUTN is the AUthentication TokeN. And XRES* is the expected response from the UE. The AV is returned to the AUSF in a Nudm_UEAuthentication_Get Response. If the SUCI was included in the request the SUPI is included in the response. Upon receipt the AUSF stores XRES* and the SUCI or SUPI temporarily.

The AUSF generates HXRES* which is the SHA-256 hash of XRES*. It generates K_{seaf} from K_{ausf}. XRES* is replaced by HXRES* and K_{ausf} is replaced by K_{seaf} inside the AV. Before being sent in an Nausf_UEAuthentication_Authenticate Response to the SEAF, K_{seaf} is removed from the AV. Once received by the SEAF, the SEAF sends RAND, and AUTN to the UE in an Authentication Request message.

The UE determines the freshness of the AV based off the AUTN. If the AV is fresh, then it is accepted by the UE. Using USIM, the UE computes a response, RES, CK, IK. CK stands for cipher key. IK stands for integrity key. CK and IK are 128-bits and derived from K, the UE's permanent key, using RAND.

$CK = f_1(K||RAND)$ where f_1 is a key generating function.
$IK = f_2(K||RAND)$ where f_2 is a key generating function.

From RES, CK, and IK the UE computes K_{ausf}, K_{seaf}, and RES*. The UE sends a NAS Authentication Response message to the SEAF containing RES*. The SEAF computes the hash of RES* and compares it to the cached value, HRES*. If the two are the same authentication is considered successful for the serving network. The SEAF sends RES* and the SUPI or SUCI of the UE to AUSF in a Nausf_UEAuthentication_Authenticate Request.

Upon receiving the Nausf_UEAuthentication_Authenticate Request the AUSF checks the freshness of the AV. If the AV is considered stale, then from the point of view of the home network, authentication is considered a failure. If the AV is fresh, RES* is compared to the cached value, XRES*, if the two are in agreement, authentication is considered successful for the home network. The AUSF then sends a Nausf_UEAuthentication_Authenticate Response message back to the SEAF indicating the result of home network authentication. If authentication was successful for the home network, the K_{seaf} is included in the Nausf_UEAuthentication_Authenticate Response message. If a SUCI or SUPI was included in the initiating message a SUPI is also included in the response. No communication services are provided to the UE until the SEAF is given the UE SUPI. Figure 2.2 summarizes the transactions in a protocol diagram.

3.2 Formal Verification

Formal verification uses mathematical concepts to ensure that a security protocol fulfills its intended design. Today software based verification methods allow protocols to be systemically analyzed for flaws. There are two broad categories of verification, model checking and logical inference. Model checking provides an exhaustive search of the model by exploration of each state and its transitions. Logical inferences is a formal mathematical verification that is seldom automated and relies on the verifier's understanding of the system. Currently model checking is the preferred method of verification. It is increasingly replacing verification by testing or simulation.

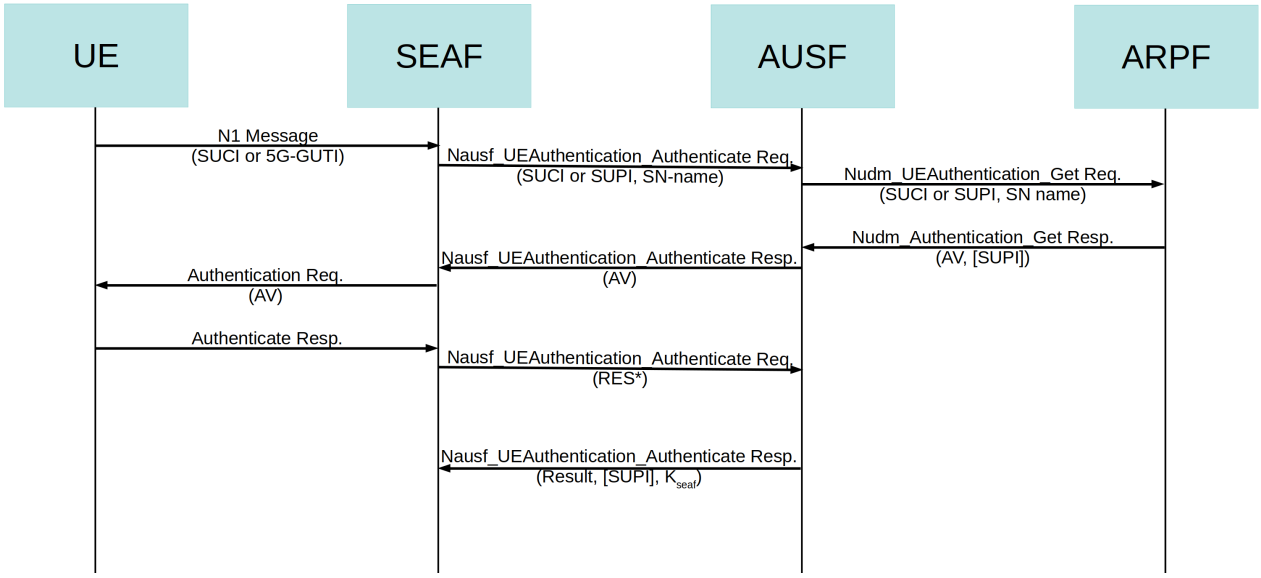

Figure 2: 5G-AKA protocol sequence diagram.

The Tamarin prover falls into the model checking class of verifier. It is a symbolic modeling and analysis tool for security protocols. The verifier uses a highly specific programming language to capture the states and transitions of the protocol in combination with the Dolev-Yao adverserial model. Tamarin represents the protocol as a state machine, with the state of the system represented by a multi-set of facts. Rules define state transitions. Each rule is composed of facts. Rules have a right side, an action and a left side. The state is initialized to an empty multiset. A rule can only be executed if the facts on the left-side of the rule are available in the multiset. There are two types of facts, linear and persistent. Linear facts are consumed when a rule is executed meaning they are no longer available for future rules. Persistent facts can be used by several rules. In the Tamarin language, the qualities of the protocol the tester intends to prove and written as lemmas. An example of a rule and a lemma are shown below.

```
rule Reveal_ltk:
[ !Ltk(A, ltk) ]
--[ LtkReveal(A) ]->
[ Out(ltk) ]

lemma Client_session_key_secrecy:
" /* It cannot be that a  */
not(
Ex S k #i #j.
/* client has set up a session key 'k' with a server'S' */
SessKeyC(S, k) @ #i
/* and the adversary knows 'k' */
& K(k) @ #j
/* without having performed a long-term key reveal on 'S'. */
& not(Ex #r. LtkReveal(S) @ r)
)
"
```

Each rule is named. In this case the name is "Reveal_ltk" The explanation mark indicates that the fact is persistent.

References

[1] 3rd generation partnership project; technical specification group services and system aspects; security architecture and procedures for 5g system 3gpp ts 33.501. Technical report, 3rd Generation Partnership Project, 650 Route des Lucioles Sophia Antipolis Valbonne France, 9 2018.

[2] Jari Arkko, Karl Norrman, Mats Näslund, and Bengt Sahlin. A usim compatible 5g aka protocol with perfect forward secrecy. In *Trustcom/BigDataSE/ISPA, 2015 IEEE*, volume 1, pages 1205–1209. IEEE, 2015.

[3] 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. ACM, 2018.

[4] Cas Cremers, Marko Horvat, Jonathan Hoyland, Sam Scott, and Thyla van der Merwe. A comprehensive symbolic analysis of tls 1.3. In *Proceedings of the 2017 ACM SIGSAC Conference on Computer and Communications Security*, pages 1773–1788. ACM, 2017.

[5] Martin Dehnel-Wild and Cas Cremers. Security vulnerability in 5g-aka draft (3gpp ts 33.501 draft v0.7.0). Available at https://www.cs.ox.ac.uk/5G-analysis/(2018/02/08).

[6] Roger Piqueras Jover and Vuk Marojevic. Security and protocol exploit analysis of the 5g specifications. *arXiv preprint arXiv:1809.06925*, 2018.

[7] Suvansh Lal, Mohit Jain, and Vikrant Chaplot. Approaches to formal verification of security protocols. 2011.

[8] Simon Meier, Benedikt Schmidt, Cas Cremers, and David Basin. The tamarin prover for the symbolic analysis of security protocols. In *International Conference on Computer Aided Verification*, pages 696–701. Springer, 2013.

[9] Anand R Prasad, Sivabalan Arumugam, B Sheeba, and Alf Zugenmaier. 3gpp 5g security. *Journal of ICT Standardization*, 6(1):137–158, 2018.

[10] The Tamarin Team. Tamarin prover manual. Available at https://https://tamarin-prover.github.io/manual/tex/tamarin-manual.pdf, note = Online; accessed 27 October 2018.

[11] Xiaowei Zhang, Andreas Kunz, and Stefan Schröder. Overview of 5g security in 3gpp. In *Standards for Communications and Networking (CSCN), 2017 IEEE Conference on*, pages 181–186. IEEE, 2017.