KOY is not UC-Secure

Naman Kumar

December 20, 2023

This document contains a proof that the PAKE protocol of Katz, Ostrovsky and Yung (2001) is not UC-secure.

1 Overview

At a high level, our attack relies on an adversary that completely disregards the presence of the Server and instead interacts with the User while executing the Server's algorithm on its own. In particular, once the protocol is initiated by User, the adversary assumes the role of the server (discarding the actual server in the process, which plays no part in the protocol henceforth) and recieves msg_1. After this, $\mathcal{A}$ is provided the messages $E|F|G|H|I|J$ by $\mathcal{Z}$ which it then forwards to User. User can then run its own session-key generating algorithm (the computation of $E^{r_1}F^{x_1}G^{y_1}(I')^{z_1}J^{w_1}$) and output the session key sk. We note that at this point, $\mathcal{A}$ and $\mathcal{Z}$ have all the information they need to run the server algorithm and ensure that their generated session key is equal to the sk generated by User.

To see why $\mathcal{S}$ cannot simulate this adversary, we attempt an ideal-world execution and pinpoint where our simulation fails. Since $\mathcal{S}$ is allowed to choose the crs, it can sample g_1 at random and set h such that $h = {g_1}^{\ell}$. After receiving the NewSession command from $\mathcal{F}_{\mathsf{PAKE}}$, $\mathcal{S}$ must simulate the User by sampling fresh randomness and computing msg_1. Since $\mathcal{S}$ does not know the password at this point it must guess some pw^* at random. Thus, in msg_1, $C = h^{r_1} \cdot \mathsf{pw}^*$ where pw^* can be no better than a random password sampled from the dictionary. $\mathcal{S}$ must then forward this message to $\mathcal{Z}$ which responds with $E|F|G|I|J$ where pw can be determined as I/F^{ℓ}. Once this has been done, $\mathcal{S}$ can run a TestPwd which would successfully allow it to choose the sk output by User (recall that a TestPwd *must* be run, since we require that msg_1 and msg_2 together with the randomness of the User and $\mathcal{A}$ together determine sk; allowing the simulation to proceed without a TestPwd would result in $\mathcal{F}_{\mathsf{PAKE}}$ outputting a uniformly random key). The problem is that even with this information, $\mathcal{S}$ can still not determine what sk is.

To determine sk, $\mathcal{S}$ can either run Server's algorithm or User's algorithm to generate session keys. Recalling that $E|F|G|I|J$ is provided to $\mathcal{S}$ directly from the environment, $\mathcal{S}$ cannot use Server's algorithm since it would require the determination of all

of x_2, y_2, z_2, r_2, w_2, which is computationally infeasible under the hardness of CDH. Alternately, $\mathcal{S}$ can run User's algorithm to determine sk, for which it already has access to x_1, y_1, z_1, r_1, w_1 and E, F, G, I, J. However, we require that this sk output by the $\mathcal{S}$ must be equal to that which $\mathcal{Z}$ determines using its own execution of the Server's algorithm ($\mathcal{Z}$ knows the randomness generated in the construction of $E|F|G|I|J$ and can thus run this algorithm). We can show that the output of the server's algorithm on $\mathsf{msg}_1, \mathsf{msg}_2$ and msg_3 by $\mathcal{Z}$ will have $(\mathsf{pw}^*/pw)^{z_2}$ as a factor. Except in the $1/\mathcal{D}$ probability case that $\mathsf{pw}^* = \mathsf{pw}$, $\mathcal{S}$ cannot determine z_2 assuming the hardness of CDH and thus will be unable to determine sk.