KOY is not UC-Secure

Naman Kumar

January 10, 2024

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 $E|F|G|I|J$ 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 $\mathsf{NewSession}$ 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 $\mathsf{TestPwd}$ which would successfully allow it to choose the sk output by User (recall that a $\mathsf{TestPwd}$ *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 $\mathsf{TestPwd}$ 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 $(\mathsf{pw}^*/\mathsf{pw})^{z_2}$ 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.

2 Proof

We will be working with the following environment and dummy adversary. We note that it follows from the correctness of the protocol that in the real-world protocol execution $\mathcal{Z}$ always outputs 1, since the algorithm of $\mathcal{Z}$ and $\mathcal{A}$ is the same as that of an honest server.

First, we will assume the hardness of CDH over the group $\mathbb{G}$. Let g be a generator and (g^a, g^b) be two elements such that $a, b \xleftarrow{\$} \mathbb{F}_{|\mathbb{G}|}$. We will show that any UC-simulator $\mathcal{S}$ that successfully simulates $\mathcal{A}$ in $\mathcal{F}_{\mathsf{PAKE}}$ can be used to compute g^{ab}.

Formally, assume that there exists a simulator $\mathcal{S}$ such that $\mathcal{Z}$ always outputs 1 in the ideal world. We will use this simulator to compute g^{ab}. Our technique works as follows: first, we send $(\mathsf{NewSession}, \mathsf{sid}, \mathsf{User}, \mathsf{Server}, \mathsf{pw})$ to $\mathcal{F}_{\mathsf{PAKE}}$ and recieve $\mathsf{msg}_1 = Client|\mathsf{VK}|A|B|C|D$ in response from $\mathcal{S}$. We set $\mathsf{pw} = C/g^b$ and sample $x_2, y_2, z_2, w_2, r_2 \xleftarrow{\$} \mathbb{Z}_q$.

$$
\begin{aligned}
\alpha' &:= H(PIDs|A|B|C|D) \\
E &:= g_1^{x_2} g_2^{y_2} h^{z_2} (cd^{\alpha'})^{w_2} \\
F &:= g_1^{r_2} \\
G &:= g_2^{r_2} \\
I &:= h^{r_2} \cdot \mathsf{pw} \\
\beta &:= H(\mathsf{msg}_1|Server|E|F|G|I) \\
J &:= (cd^{\beta})^{w_2}
\end{aligned}
$$

ie. the same computation as that of the honest server with a special choice of pw. Forwarding this to $\mathcal{F}_{\mathsf{PAKE}}$, we recieve $\mathsf{msg}_3 = K|\mathsf{Sig}$ and $(\mathsf{sid}, \mathsf{sk})$ in return.

Environment $\mathcal{Z}$:

1. $\mathcal{Z}$ selects $\mathsf{pw} \stackrel{\$}{\leftarrow} \mathcal{PW}$, where $\mathcal{PW} \subseteq \mathbb{G}$ is the password dictionary. It then sends $(\mathsf{NewSession}, \mathsf{sid}, \mathsf{User}, \mathsf{Server}, \mathsf{pw})$ to User.

2. $\mathcal{Z}$ recieves $\mathsf{msg}_1 = Client|\mathsf{VK}|A|B|C|D$ from $\mathcal{A}$ and samples $x_2, y_2, z_2, w_2, r_2 \stackrel{\$}{\leftarrow} \mathbb{Z}_q$. It then sets

$$\begin{aligned}
\alpha' &:= H(PIDs|A|B|C|D) \\
E &:= g_1^{x_2} g_2^{y_2} h^{z_2} (cd^{\alpha'})^{w_2} \\
F &:= g_1^{r_2} \\
G &:= g_2^{r_2} \\
I &:= h^{r_2} \cdot \mathsf{pw} \\
\beta &:= H(\mathsf{msg}_1|Server|E|F|G|I) \\
J &:= (cd^{\beta})^{w_2}
\end{aligned}$$

   and forwards $\mathsf{msg}_2 = Server|E|F|G|I|J$ to $\mathcal{A}$.

3. $\mathcal{Z}$ recieves $\mathsf{msg}_3 = K|\mathsf{Sig}$ from $\mathcal{A}$ and $(\mathsf{sid}, \mathsf{sk})$ from User.

4. $\mathcal{Z}$ sets $C' = C/\mathsf{pw}$ and then checks if $\mathsf{Vrfy}_{\mathsf{VK}}(\mathsf{msg}_1|\mathsf{msg}_2|K, \mathsf{Sig}) = 1$. If yes, it calculates $\mathsf{sk}_S = A^{x_2} B^{y_2} (C')^{z_2} D^{w_2} K^{r_2}$ and outputs 1 if $\mathsf{sk}_S = \mathsf{sk}$. Otherwise it outputs 0.

Dummy Adversary $\mathcal{A}$:

1. $\mathcal{A}$ recieves $\mathsf{msg}_1 = Client|\mathsf{VK}|A|B|C|D$ from User, which it forwards to $\mathcal{Z}$.

2. $\mathcal{A}$ recieves $\mathsf{msg}_2 = E|F|G|I|J$ from $\mathcal{Z}$ and sends it to User.

3. $\mathcal{A}$ recieves $\mathsf{msg}_3 = K|\mathsf{Sig}$ and forwards this to $\mathcal{Z}$.

Figure 1: Our Setup.