Dear Shepherd,

Thanks for the feedback! Based on the revision requirements, we came up with a revision plan as follows.

We started to revise the manuscript and uploaded the first revision for your review. But we recognize that it may be difficult to track the detailed modifications that we are making in the revision without a high-level explanation of why we choose to revise the manuscript in this way. We apologize for the confusion it may cause to the shepherding process. So, we add the below plan to explain how we understand the revision requirements and our plan to address them. Could you please kindly let us know if the proposed revision plan is appropriate? Any comments or suggestions are highly appreciated.

\textbf{1. Remove or clean-up Section 3.2}

\textbf{Revision 1:} We will remove Section 3.2 as suggested. We agree with the reviewers that the dilemma discussed in this subsection was specific to our proposed solution. So, we plan to remove all the texts related to the identity dilemma in the manuscript, and instead, directly explain the proposed identity transformation concept for securing SSO in this section. Accordingly, we will integrate Section 3.3 of the previous manuscript into Section 3.2 of this revision.

\textbf{2. Refine the security analysis}: Define proper notions to prove if you do a formal proof, and provide better intuitions on the challenges in proving each notion w.r.t the chosen adversarial model. Make clear which assumptions each property relies on (e.g., pull them all into a theorem).

\textbf{Revision 2.1:} We plan to formally define four SSO security requirements, namely RP designation, user identification, integrity, and confidentiality, and two privacy requirements, namely privacy against IdP and privacy against colluding RPs. Meanwhile, we will define an adversarial model and the assumptions for each security/privacy requirement.

\textbf{Revision 2.2:} Then, we will provide a formal proof to prove each security requirement under the adversarial model. In each requirement (or theorem), we will emphasize the arguments known (or unknown) to each component in the SSO scenario and also update Table 3 - The notations in the UPPRESSO protocols, to emphasize an argument known (or unknown) to each component. We will formalize the proofs in the form of six theorems.

\textbf{3. Clarify the protocol design in the write-up} (i.e., the check in 3.3, and the double computation of PID\_RP).

\textbf{Revision 3.1:} Sorry for the confusion. In Step 3.3, “the IdP checks whether the received $PID\_{RP} is valid” is to ensure that the received $PID\_{RP} is a point on the EC curve.

We will rephrase this statement to avoid confusion.

\textbf{Revision 3.2:} In the negotiation of $PID\_{RP}$ between a user and an RP, one component calculates $PID\_{RP} = [t]ID\_{RP}$ and the other needs to check that $PID\_{RP}$ is equal to $[t]ID\_{RP}$; otherwise, attacks happen as below.

Let’s assume a user generates $t$ and calculates $PID\_{RP} = [t]ID\_{RP}$. After receiving $t$ from the user and extracting $PID\_{RP}$ from a token, an RP checks that $PID\_{RP} = [t]ID\_{RP}$, because the correct account $Acct = [u]ID\_{RP}$ is derived only if this equation holds.

An attack could manipulate the account as follows, if the RP does not check this equation. To login as any $Acct$, a malicious user with $ID\_{U'} = u'$ might first generate a random number $t'$, and calculate $[t'u'^{-1}]Acct$ as $PID\_{RP}$ to request an identity token. Then, the IdP will calculate $PID\_{U'} = [u'][t'u'^{-1}]Acct = [t']Acct$.

Without checking $PID\_{RP} = [t']ID\_{RP}$ or not, the RP finally allows the malicious user to login as $[t'^{-1}]PID\_{U'} = Acct$.

Meanwhile, if $t$ is generated and $PID\_{RP}$ is calculated by the RP, a user also needs to by himself check that $PID\_{RP} = [t]ID\_{RP}$; otherwise, an attacker could login as any account as follows.

A malicious user $U'$ initiates a login request to an honest RP, and receives $PID\_{RP}$.

Then it colludes with a malicious RP denoted as $RP'$, which sends $PID\_{RP}$ to an honest user $U$.

Without checking $PID\_{RP} = [t']ID\_{RP'}$, this honest user will present a token binding $PID\_U$ and $PID\_{RP}$ to $RP'$. This token enables malicious $U'$ to login as the honest user's account at the honest RP.

We explain this design (one component calculates $PID\_{RP} = [t]ID\_{RP}$, and the other component checks that $PID\_{RP} = [t]ID\_{RP}$ after receiving $t$.) in the revision.

\textbf{4. Elaborate on the weakness of having t in the IdP context or integrating a suitable MPC scheme}, as noted in the response.

\textbf{Revision 4:} We plan to add a discussion in Section 4.4 to clearly state that $t$ is critical to calculating $PID\_{RP}$ and therefore requires to be processed by only the honest entity (and its script). Since the IdP is assumed honest-but-curious, we extend this trust to the IdP script that user downloads from the IdP. If an IdP inserts malicious scripts to steal $t$ to the IdP web server, this is a violation from the protocol, and these scripts will be seen by all users because these scripts are sent to browsers. So we exclude them in this assumption.

We will also add a discussion on the use of trusted browser extensions to prevent leakage of the RP’s identity (i.e., $ID\_{RP}$).

We do not consider an MPC scheme to calculate $t$. When we consider malicious scripts from the IdP, protecting $t$ is meaningless, because the IdP script knowing $ID\_{RP}$ and $Cert\_{RP}$ will directly send $ID\_{RP}$ to the IdP web server.

\textbf{5. Illustrate the differences compared to PrivacyPass [27] and Trust-Tokens [26], and highlight where the protocol actually shows significant novelty.}

We plan to improve the discussion of related work in Section 2.3 and add comparison with recent related work, including PrivacyPass [27] and Trust-Tokens [26].

\textbf{Revision 5.1:} In particular, we consider PrivacyPass and Trust-Tokens as “anonymous tokens”. PrivacyPass [27] and Trust-Tokens [26], similar as UPRESSO, adopt cryptographic techniques proposed in [1] for OPRFs. We will add discussion about this comparison in terms of cryptographic constructs and techniques in Section 2.3.

\textbf{Revision 5.2:} Then, we plan to elaborate the novelty of UPRESSO, compared with PrivacyPass and Trust-Tokens, from three aspects: (1) UPPRESSO identifies each user at an RP, while PrivacyPass and TrustToken anonymous tokens support anonymous SSO service, which rely on one consistent private key to serve all users. (2) UPPRESSO utilizes the cryptographic technique differently from PrivacyPass and Trust-Tokens, by using them to transform identities in SSO. (3) UPPRESSO supports more privacy requirements, i.e., the unlinkability across RPs, than anonymous SSO enabled using PrivacyPass and Trust-Tokens. This property of the cryptographic skills is not considered in either OPRFs or anonymous tokens.

We will add this new reference to the revised manuscript.

[1] S. Jarecki,A. Kiayias,H. Krawczyk,and J. Xu, "Highly-efficient and composable password-protected secret sharing (or: How to protect your Bitcoin wallet online)," in 1st IEEE European Symposium on Security and Privacy (EuroSP), 2016, pp. 276–291.

\textbf{Revision 6:} We will rigorously improve the writing of the paper and address typos/errors in the presentation.