Errata

Georgia Tsaloli

I would like thank my opponent Prof. Claudio Orlandi for carefully checking my thesis. Below, I provide some clarifications regarding the raised remarks for each of the thesis' Chapters.

Chapter A:

There have been some typos and/or unclear statements identified, which can be fixed as follows:

1. In pages 29 and 35, the relation $m > n \cdot t$ found in the ShareSecret algorithm is a typo left there (it has been removed from other places of the paper but still exists in these two spots) and should be removed to avoid confusion. Similarly, must be removed from page 41.

2. In page 33. The correctness definition is correct for our construction but in order to have a general correctness definition capturing other constructions, too, we should also add the requirement $y = f(x_1, \ldots, x_n)$ i.e., $\Pr\left[\ \mathbf{Verify}(\tau_1, \ldots, \tau_n, y, \sigma) = 1 \wedge y = f(x_1, \ldots, x_n) \ \right] = 1$ instead of only having this
   $\Pr\left[\ \mathbf{Verify}(\tau_1, \ldots, \tau_n, y, \sigma) = 1 \ \right] = 1$.

3. In pages $35, 37$. In the algorithm ShareSecret, we create polynomials of degree t. We should be more precise, clarifying that $t = m - 1$.

4. In page 37. Observation 1 is incomplete. Therefore, the Discrete Logarithm Problem (DLP) assumption can be stated instead (which is why Observation 1 was mentioned). More precisely, the new statement will be simply the DLP assumption as follows:
   Assumption 1 *For g and g^a given values in a group G, the discrete logarithm problem (DLP) is hard.*

5. A clarification: it is considered that clients cannot be corrupted. Instead, throughout the paper we considered that servers are untrusted (*e.g.,* this is clear in the security definitions as well in several parts describing our setup in the paper). Additionally, addressing such a scenario is also mentioned as a future direction in the Conclusion section of my Introduction to this thesis (page 21).

An important point identified by the opponent is that the publicly encoded values τ_i may leak information about the clients' secret inputs x_i. To overcome this leakage, an approach similar to the one adopted in Chapter C can be applied. More precisely, this means that in the public values τ_i, the inputs of the clients are masked with some randomness. By adopting this countermeasure, the security and verifiability definitions hold. This requires a pre-setup phase for all users, *i.e.,* all users are given a secret key that can be used to generate the randomness, as in Chapter C. In particular, the public encoded values will be $e(g, g^{\tilde{x}_i + R_i})$ instead of $e(g, g^{\tilde{x}_i})$, similarly to the approach in Chapter C. An alternative direction to overcome the identified weakness is using the Pedersen Commitment to mask the clients' inputs x_i's *i.e.,* $e(g, g^{\tilde{x}_i}) \cdot h^{R_i}$, where no one knows discrete log of h in base g and R_i is the randomness.

Following up to the security definition used in Chapter A, in page 34, an explicit list of the public values available to the adversary (e.g., partial values y^j, partial proofs σ^j) can be included, to make the definition more complete.

Lastly, we state in Chapter A that we provide a multiplicative homomorphic secret sharing scheme for computing the product function $f = x_1 \cdot \ldots \cdot x_n$. Indeed, we provide a multiplicative secret sharing (note that we name it multiplicative since the final computed value comes after multiplication over all partial values) for elliptic curve point multiplication instead, which is in fact addition along an elliptic curve.

Chapter C:

1. In pages $66, 69, 71$ degree t needs to be specified as $t = m - 1$ (as in Point 3, Chapter A).

2. In page 65. The last sentence in the introductory paragraph of Section C.4, i.e., "However,...,honest." must be removed since it creates confusion for the setup considered in our work.

Chapter D: We clarify that the users can be corrupted only during Round 2 of the proposed protocol (page 94), i.e., when they use t_{key} secret sharing for exchanging keys and they might try to infer information of other users (i.e., sk_i^{KA}). Additionally, in page 92 we provide the capabilities of the servers in the Threat Model described, which does not include that servers can collude with the users.

Chapter B: In page 51, $p_1 = com(k; r_1)$ should be provided before sending x, and the PRF should be implemented with a random oracle. Alternatively, the coin-flip protocol can be used (in that case, the key should be refreshed at each run of the protocol).