周报-向嘉豪 (2025 年 1 月 13 日)

摘要：本周研究工作主要聚焦于 SPHINCS$^+$ 签名方案中的两个核心组件：FORS（Forest Of Random Subsets）机制与 HT（Hypertree）树结构。通过深入分析 NIST 提交的源代码实现与相关文献，我们系统地梳理了签名生成过程中的关键步骤与树形结构的层级关系。实验方面，我们使用 Python 实现了 FORS 子树生成与验证的原型系统。

下周计划：1) 完成对 SPHINCS$^+$ 的完整实现，包括 FORS、HT 树等关键组件；2) 进一步研究 SPHINCS$^+$ 的并行计算特性，探索 GPU 加速优化方案。

1 SPHINCS$^+$ (SPX)

1.1 签名算法初步理解

本周，我们通过仔细研读 SPX 签名函数的论文与实现源码，逐渐理清了其核心思想。由于论文多以伪代码展示，并缺少针对签名结构的详细说明，初读时较为费解。但在反复对照源码与文献之后，我们对 SPX 整体的签名流程有了更为清晰的认知。

SPX 的签名依托 Merkle Hash 树，将多个安全私钥 (sk_i) 视为树的叶子节点，多次哈希后得到根节点 (pk) 并对外公开，如图 1所示。对于签名方而言，所有 sk_i 均可使用，但为了保护私钥安全，只会在签名中暴露必须的中间哈希节点与局部私钥。验证方只需据此重构根节点，与公开的 pk 对比一致，即能完成认证。

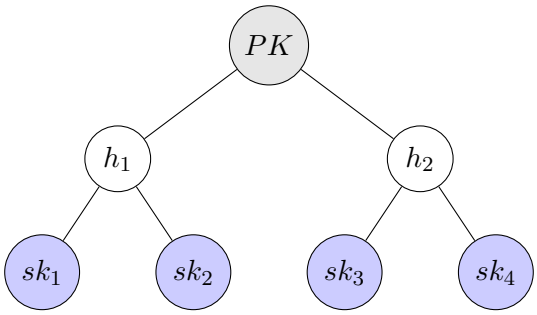

图 1: SPX 中的 Merkle Hash 树结构示意图

1.2 FORS 树

FORS（Forest Of Random Subsets）树由 k 个并排的 Merkle 子树组合而成（图2）。每个子树根节点用于拼接形成 FORS 的签名 SIG_{FORS}. 在验证环节，需要公开相应私钥部分以及各子树的中间哈希节点，以重建并校验每个子树的根节点。FORS 基于消息 m 的哈希值，快速定位并公开对应的 sk_i，然后将所有子树的根节点拼接成一个整体，用于后续 HT 树的输入。

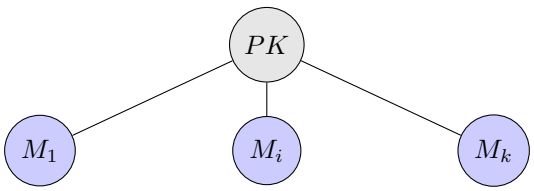

图 2: FORS 树示意图：k 个 Merkle 子树并排组合

1.3 HT 树

HT（Hypertree）结构采用分层聚合的方式（图3）：每层 XMSS 树的根节点作为下一层的叶子节点，最终在顶部生成全球的 PK. 在验证环节，SPX 结合各层子树的 WOTS+ 签名与中间哈希节点来完成验证流程。

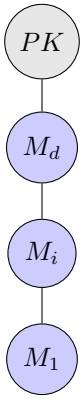

图 3: HT 树示意图：逐层汇聚得到全局公钥