Stateless Hash-Based Digital Signature Standard

Dokumentation

Simon Hensel[1] and Helena Richter[2]

[1] Albstadt-Sigmaringen University, Albstadt, Germany, hensels1@hs-albsig.de
[2] Albstadt-Sigmaringen University, Albstadt, Germany, richtehe@hs-albsig.de

Abstract. In this paper we prove that the One-Time-Pad has perfect security. Pellentesque habitant morbi tristique senectus et netus et malesuada fames ac turpis egestas. Donec odio elit, dictum in, hendrerit sit amet, egestas sed, leo. Praesent feugiat sapien aliquet odio. Integer vitae justo. Aliquam vestibulum fringilla lorem. Sed neque lectus, consectetuer at, consectetuer sed, eleifend ac, lectus. Nulla facilisi. Pellentesque eget lectus. Proin eu metus. Sed porttitor. In hac habitasse platea dictumst. Suspendisse eu lectus. Ut mi mi, lacinia sit amet, placerat et, mollis vitae, dui. Sed ante tellus, tristique ut, iaculis eu, malesuada ac, dui. Mauris nibh leo, facilisis non, adipiscing quis, ultrices a, dui.

Keywords: Something · Something else

1 Introduction

In the realm of modern cryptography, ensuring data integrity and authenticity is a cornerstone of secure communication and digital transactions. Digital Signature Algorithms (DSAs) are pivotal in this context, enabling entities to validate the provenance and integrity of data without the need for direct interaction. Among the diverse array of DSAs, the Stateless Hash-based Digital Signature Algorithm (Stateless Hash-DSA) emerges as a robust and efficient cryptographic solution tailored for secure, lightweight, and future-proof applications.

This project aims to explore and implement the Stateless Hash-DSA, an advanced algorithm designed to leverage the inherent strength of hash functions while addressing challenges associated with stateful signature schemes. Unlike traditional approaches that rely on maintaining and managing state information, Stateless Hash-DSA eliminates the complexities of state management, thereby reducing operational risks such as accidental state reuse—a common vulnerability in stateful systems.

By implementing the Stateless Hash-DSA, this project contributes to the advancement of secure digital signature techniques suitable for environments where simplicity, efficiency, and resistance to quantum adversaries are crucial. The following documentation outlines the theoretical foundations, implementation details, and practical applications of the Stateless Hash-DSA.

2 Hash-based Cryptography

Stateless hash algorithms are cryptographic methods that leverage hash functions to ensure security while eliminating the need to maintain state information during operation. In traditional stateful algorithms, maintaining a record of past operations is essential to prevent vulnerabilities such as key or signature reuse. However, this reliance on state can introduce operational complexities and risks, particularly in distributed or constrained

environments.
By contrast, stateless hash algorithms operate without requiring a persistent state, relying solely on the cryptographic strength of hash functions to secure data. This design simplifies implementation, reduces the risk of errors associated with state management, and enhances resilience against attacks that exploit state inconsistencies. Stateless approaches are especially valuable in applications like digital signatures, where lightweight, efficient, and secure operations are critical.
Stateless hash algorithms are also well-suited for post-quantum cryptography, leveraging the inherent robustness of hash functions against quantum adversaries, ensuring their applicability in future cryptographic systems.

3 Digital Signature Algorithm (DSA)

The Stateless Hash-based Digital Signature Algorithm (SLH-DSA) is a cryptographic scheme that combines the strength of hash-based security with the simplicity and efficiency of a stateless design. SLH-DSA is a robust alternative to traditional digital signature algorithms, addressing challenges such as state management, accidental key reuse, and scalability in distributed or resource-constrained environments.
At its core, SLH-DSA employs cryptographic hash functions to generate secure, verifiable signatures. Unlike stateful algorithms that require persistent tracking of used keys or states to ensure security, SLH-DSA eliminates this dependency by deriving keys and signatures dynamically in a manner that guarantees their uniqueness and integrity. This stateless approach significantly reduces the operational risks and complexities associated with managing and safeguarding state information.

The Algorithm can be divided into 4 Steps:

1. **Key Generation** of private and public key
2. **Key Distribution** of public key
3. **Signature Generation** by sender
4. **Signature Verification** by receiver

SLH-DSA is especially useful for modern applications where lightweight and scalable solutions are critical. It is resilient to common vulnerabilities found in stateful systems and offers enhanced security against evolving threats, including those posed by quantum computing. Furthermore, its reliance on well-studied cryptographic hash functions ensures that it remains a practical and secure choice for both current and future cryptographic needs.
This algorithm is a key step toward creating secure, efficient, and future-proof digital systems, making it a compelling choice for developers and security professionals seeking innovative cryptographic solutions.

4 Implementing the Algorithm

Morbi luctus, wisi viverra faucibus pretium, nibh est placerat odio, nec commodo wisi enim eget quam. Quisque libero justo, consectetuer a, feugiat vitae, porttitor eu, libero. Suspendisse sed mauris vitae elit sollicitudin malesuada. Maecenas ultricies eros sit amet ante. Ut venenatis velit. Maecenas sed mi eget dui varius euismod. Phasellus aliquet volutpat odio. Vestibulum ante ipsum primis in faucibus orci luctus et ultrices posuere cubilia Curae; Pellentesque sit amet pede ac sem eleifend consectetuer. Nullam elementum,

urna vel imperdiet sodales, elit ipsum pharetra ligula, ac pretium ante justo a nulla. Curabitur tristique arcu eu metus. Vestibulum lectus. Proin mauris. Proin eu nunc eu urna hendrerit faucibus. Aliquam auctor, pede consequat laoreet varius, eros tellus scelerisque quam, pellentesque hendrerit ipsum dolor sed augue. Nulla nec lacus.

4.1 Python

4.2 C