

SPHINCS Interim Report

Daniel Kirkpatrick
Vedanth Narayanan
February 18, 2016

Introduction

Have a brief intro about Digital Signatures. Then talk about the relevance of SPHINCS. Important to make note of how SPHINCS integrates multiple technologies and wraps it all together.

Preliminaries

This section is utilized to briefly talk about existing signature schemes. The sole reason for this is so future references to the specific schemes are not ambiguous.

Lamport Signature Scheme

The Lamport Signature Scheme was created by Leslie Lamport in 1989, and it is the simplest signature scheme that exists. It is also the first One-Time signature that was invented. This scheme makes use of a cryptographic hash function that has already been predetermined. Ideally, the hash output For our purposes, Sha-256 is used to aid in explaining the scheme.

WOTS

The primary idea behind the scheme is to break the messages into little blocks, that get processed together, and having an input run through a hash function several times. The number of iterations entirely depends on the message that needs to be signed. WOTS was built on top of the Lamport signature scheme, and the expectation is for it to be intuitive in its logic, but it's not the case. The complexity of the scheme is heavily influenced by the logic in figuring out the number of iterations necessary for a value to go through the hash function.

WOTS+

WOTS+ is very similar to WOTS, except for the addition of XORing random elements every time a value is iterated over hash function. In the key generation phase, WOTS+ generates a set of random numbers that will serve for XORing. Just like the keys are

split into chunks, so are the random elements. They get incorporated in the following recursive chaining function

$$c_i^k(x, \mathbf{r}) = f_k(c_k^{i-1}(x, \mathbf{r}) \oplus r_i) \quad (1)$$

The equation is strictly $i > 0$, but in the case of $i = 0$, $c_0^k(x, \mathbf{r}) = x$. The equation is clever in that it makes sure to XOR different values in every iteration.

Lamport+ Signature Scheme

Lamport+ Signature Scheme is the new scheme we are proposing. It not only brings the simplicity of the original Lamport scheme, but also pulls in elements of the WOTS+ scheme. Our hope is that the original scheme's security is withheld, if not enhanced. Please note that the security of the proposed scheme has not been proven, but it can very well be inferred from the previous. Similar to how WOTS+ introduces XORing of randomized elements to WOTS, the same principle is introduced to Lamport. The following is meant to give an idea of the new scheme, before we extrapolate it.

Key Pair Generation: n is the number of random pairs to be generated. Random element $r \in \{0, 1\}^n$ is chosen. The secret key is $sk = ((sk_{0,0}, sk_{0,1}), (sk_{1,0}, sk_{1,1}), \dots, (sk_{n,0}, sk_{n,1}))$. Each key is n -bits long. Let $f_k : \{0, 1\}^n \rightarrow \{0, 1\}^n | k \in \mathcal{K}_n$. The cryptographic hash function f_k outputs a n -bit value. Public keys are derived by $pk = f_k(sk_{0,1} \oplus r)$. The whole set of sk is run through the hash function, so get a bijective public key set. The public key is the following

$$\begin{aligned} pk &= ((pk_{0,0}, pk_{0,1}), (pk_{1,0}, pk_{1,1}), \dots, (pk_{n,0}, pk_{n,1})) \\ &= ((f_k(sk_{0,0} \oplus r), f_k(sk_{0,1} \oplus r)), (f_k(sk_{1,0} \oplus r), f_k(sk_{1,1} \oplus r)), \dots, \\ &\quad (f_k(sk_{n,0} \oplus r), f_k(sk_{n,1} \oplus r))) \end{aligned} \quad (2)$$

Signature Generation: The message to be signed is m . This message is hashed by $h(m) = f_k(m \oplus r)$. The output of the hash is now n -bits. Based on every single bit (0 or 1), the corresponding key from a bit pair is selected. These keys make up the signature of the message.

Signature Verification: At this point, the verification is almost trivial. The verifier first obtains the hash of the message. Based on the bits of the hash, they pick out the corresponding keys from Public Key of the signer. Now the signature that the signer gave the verifier is hashed, and ideally the values should equal the values that the verifier picked out of the public key. If and when the values don't match is when we know something is wrong. Also, make a note that when something is hashed, the random element is XORed in. Thus, this is something that the signer passes off to the verifier.

Lamport+ Hash Chain

Hash chains aid One-Time signatures to be used multiple times with a single key. This can be applied to the Lamport+ signature scheme. The underlining idea here is that after the public key is generated in the scheme, another set of keys are generated based on the previous public key getting hashed.

There are two important things to make note of here. First off, we introduce a new parameter, which we call $l \in \mathbb{N}, l > 0$. This parameter helps keep track of how long the chain is, or many times new public keys have been derived. The second change involves generating a new randomized element for every new hashed public key. The random element from the previous level should not be used again. Now \mathbf{r} is a set, $\mathbf{r} = (r_1, r_2, \dots, r_l)$. Once again, this concept is borrowed from WOTS+, where the random element changes based on the iteration number.

Lamport+ Hash Tree

Note that this section hasn't been fully developed yet, but it is still in the works. If Lamport+ is going to get incorporated into SPHINCS, then it needs to be in a tree fashion.

Benchmark

This section is for a little graph of RSA and ECDSA times.

Challenges

The single biggest challenge for us was primarily getting acquainted with the material. To properly, and thoroughly, understand SPHINCS we needed to get caught up with a lot of reading. There were multiple papers that required time and dedication to fully understand. Understanding the tools and technologies is crucial if we want to be successful. On top of this, we had the added challenge of figuring out how to piece together the technologies, and how SPHINCS uses them.

Conclusion

References