Side Channel Resistant Sphincs+

Scott Fluhrer, Principal Engineer
Foundational Security Technologies, Cisco Systems

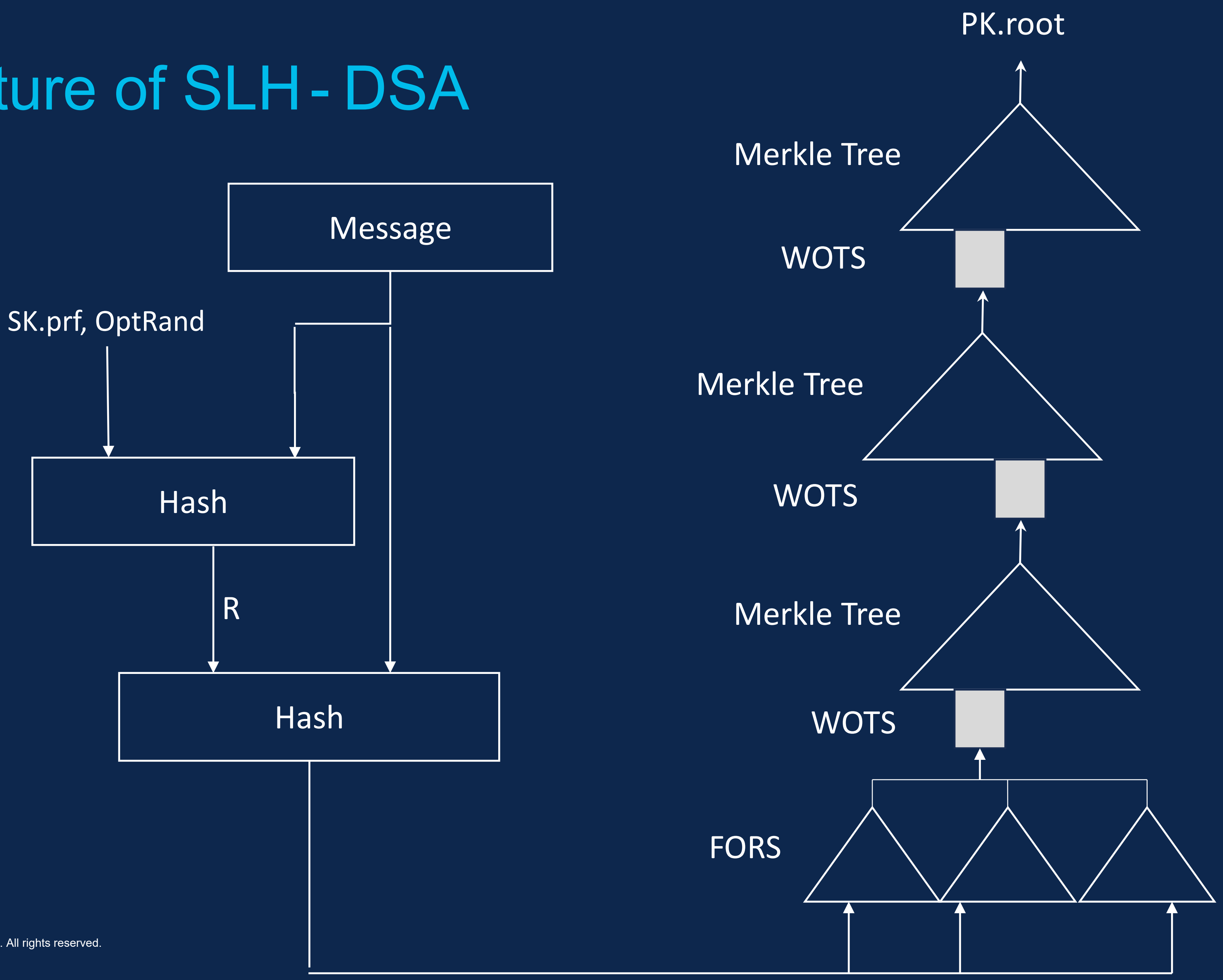
April 10, 2024

SLH- DSA (Sphincs+)

Signature Scheme based on hash functions.

- FIPS 205.
- Conservative design.
- Not stateful – works like any other standard signature scheme.
- Drawbacks: slow signature generation, large signatures.

The structure of SLH - DSA



SLH- DSA (Sphincs+)

Side channel resistance:

- Inherently strong against timing and cache side channels.
- But how about electronics - based side channels?

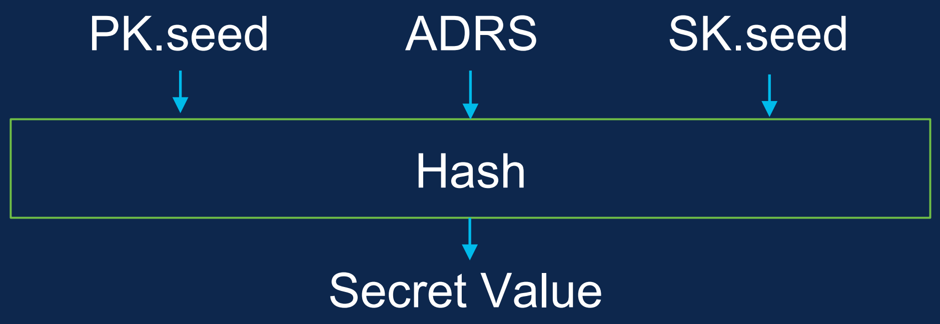
Electronics- based sign channels are ones where the attacker listens into the circuit gates as they are performing the operation.

- Examples include DPA and EMF attacks.

One major issue Sphincs+ has with electrical side channels

SLH- DSA uses a lot of secret values internally.

To generate them, it does this operation:



ADRS is known and varies per Secret Value.

SK.**seed is the ‘key to the kingdom’.**

The adversary “hears” the SK.seed interact with a number of different ADRS values.

How to defend

The standard method: use a threshold implementation of the hash function.

- Logical values split up into multiple (N) physical shares.
- Randomness injected to prevent correlations of fewer than N values.

Proposed alternative method: Determanism.

- Make sure every secret internal value is used only in a handful of contexts.
- For each context, use the exact same inputs each time.

Why does Determanism protect us?

Each secret value is made up of a number of 64 bit words.

By limiting the number of contexts, the attacker will see a specific 64 bit word operated on in a handful of different ways.

Because the gates operate on all 64 bits at once, the noise they produce are added together.

Hence, the attacker doesn't have enough information to recover the full state.

However, he can get partial information (e.g. the hamming weight); we need to take that into account.

How can we apply this idea to Sphincs+

There is no obvious way to apply this to the PRF function.

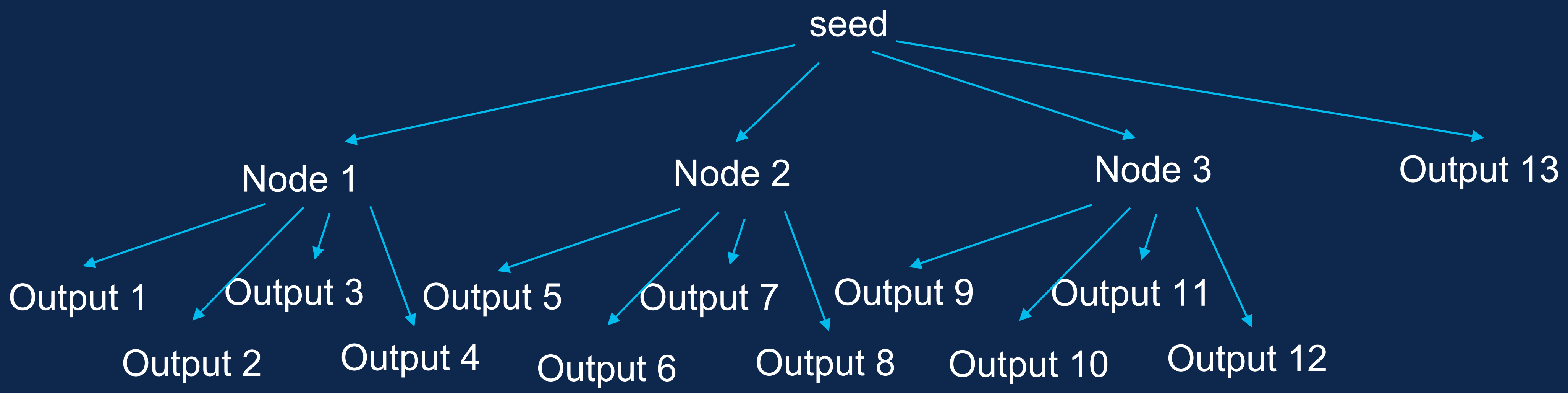
- So, we change the PRF function.
- This means that we are no longer strictly “SLH-DSA”.
- However, because the verifier doesn’t see the PRF, we still generate SLH-DSA compatible signatures.

We also limit ourselves to SHAKE parameter sets.

- SHAKE is far friendlier to threshold implementations (which we will use at one point).
- SHAKE can generate longer outputs efficiently.

Protecting PRF

We change the PRF to use a tree - based approach.



Each output is used either as a PRF output or the seed for a lower level tree.
Each internal value is 3n (384, 576, 768 bits long).

Protecting the F function

This is the single input hash function used at the base of the FORS trees and during the WOTS chaining.

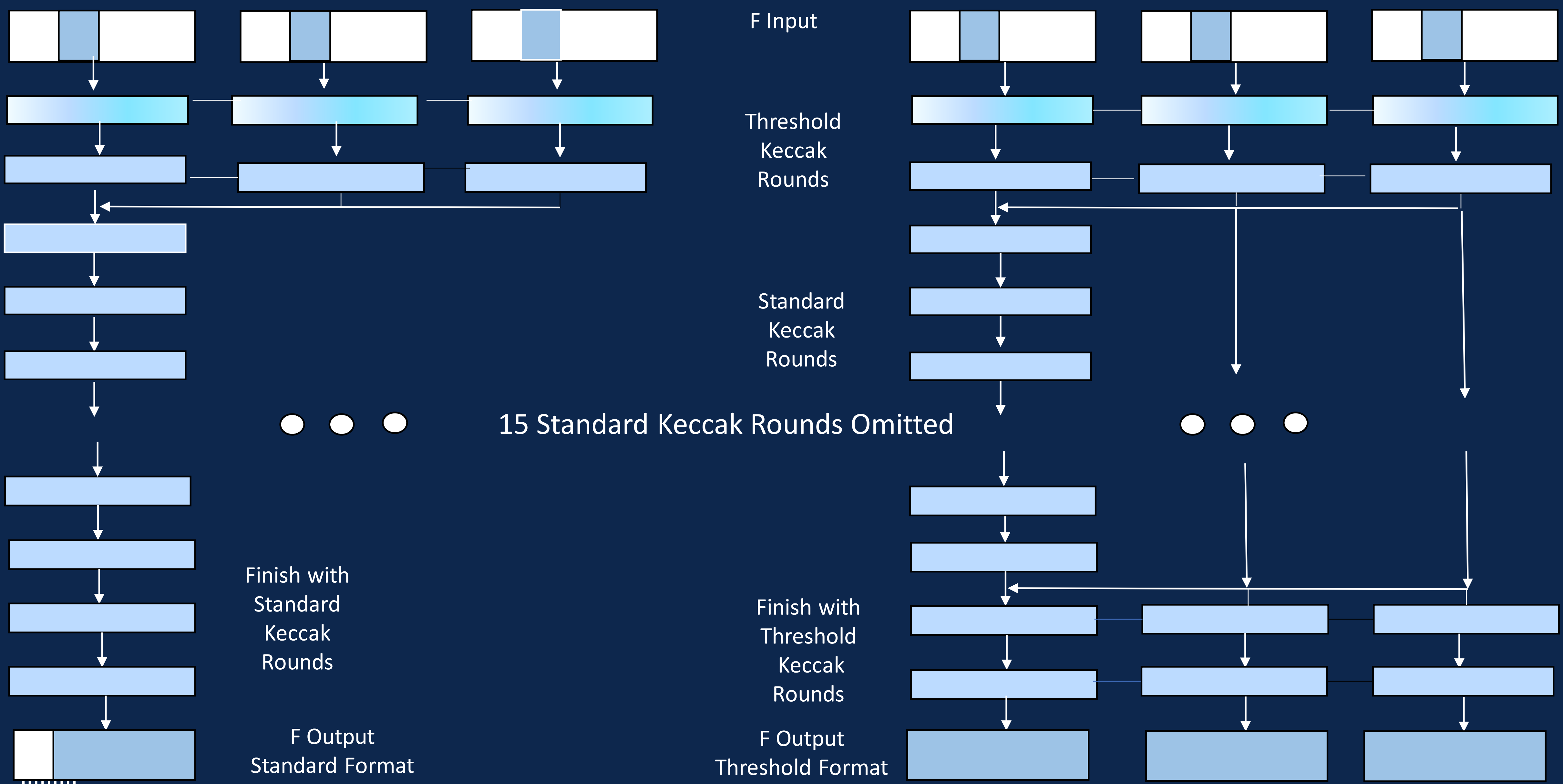
Each input is either from the PRF or a previous F function.

We interpret the 3n bit input as three n - bit shares.

- We place each n bit share into a 3 separate inputs to a threshold SHAKE implementation

Threshold F function – public output

Threshold F function – secret output



Protecting the SK.prf value

This is expensive to protect, so we don't.

SK.prf is there as a 'belt - and- suspenders' approach to make sure that R is unpredictable, even with bad or no entropy.

So, we punt and mandate 'you must have good entropy when generating signatures'.

Summary

We give an alternative way to side channel protect Sphincs+

We have implemented it – performance is 1.7x slower than the reference code

https://github.com/sphincs/sidechannel - resistent

Cavaet: this approach needs serious vetting before it should be used in practice.

The bridge to possible