From: Blumenthal, Uri - 0553 - MITLL <uri@ll.mit.edu> via pqc-forum@list.nist.gov
To: Scott Fluhrer (sfluhrer) <sfluhrer@cisco.com>, Scott Fluhrer (sfluhrer) <sfluhrer=40cisco.com@dmarc.ietf.org>, Mike Ounsworth <mike.ounsworth@entrust.com>, pqc-forum <pqc-forum@list.nist.gov>
CC: pqc@ietf.org, cfrg@irtf.org
Subject: Re: [pqc-forum] RE: [CFRG] Design rationale for keyed message digests in SPHINCS+, Dilithium, FALCON?
Date: Monday, September 19, 2022 11:03:00 AM ET
Attachments: smime.p7m

Scott,

Thank you! My point basically was that with any accepted standard hash-functions (currently, SHA2 and SHA3 families) for “pre-hash”, we’d be OK.

I have no problem with SHA2-512, but even if somebody pre-hashed with SHA2-256, it wouldn’t be a problem.

And an IETF-specified protocol can easily require that hash-function it uses is either collision-resistant or belongs to the “approved” set (like, what you described).

TNX

--

V/R,

Uri

There are two ways to design a system. One is to make it so simple there are obviously no deficiencies.

The other is to make it so complex there are no obvious deficiencies.

- C. A. R. Hoare

> **From:** "'Scott Fluhrer (sfluhrer)' via pqc-forum"
> **Reply-To:** "Scott Fluhrer (sfluhrer)"
> **Date:** Monday, September 19, 2022 at 10:36
> **To:** Uri Blumenthal , "Scott Fluhrer (sfluhrer)" , Mike Ounsworth , pqc-forum
> **Cc:** "pqc@ietf.org" , CFRG

Subject: [pqc-forum] RE: [CFRG] Design rationale for keyed message digests in SPHINCS+, Dilithium, FALCON?

From: CFRG **On Behalf Of** Blumenthal, Uri - 0553 - MITLL
Sent: Monday, September 19, 2022 8:55 AM
To: Scott Fluhrer (sfluhrer) ; Mike Ounsworth ; pqc-forum
Cc: pqc@ietf.org; cfrg@irtf.org
Subject: Re: [CFRG] Design rationale for keyed message digests in SPHINCS+, Dilithium, FALCON?

> On the other hand, if one were to use prehashing, I would argue that the prehash should be with a very conservative hash function (say, SHA-512 or SHA3-512); we are putting all our hybrid eggs in this one hashing basket, and so we should make sure this one basket is a good one.

Do any of the currently standardized hash functions, such as SHA{2,3}-384 or SHA{2,3}-512 (or even SHA{2,3}-256) fail the “very conservative” criteria? Is there any reason to expect a “weak” hash function, any more than you’d expect a “weak” block cipher?

Actually, I did suggest SHA{2,3}-512, and so they fulfill what I had in mind by “very conservative”.

One could easily claim that SHA2-256 would be sufficient; however one additional criteria that should be considered is cost – if SHA2-512 is no more costly than SHA2-256, my opinion is that SHA2-512 should be preferred (it’s overkill, but there’s nothing wrong with cheap overkill). I would claim that this is even more true in the hybrid scenario – the attacker can forge if either they can break the hash function OR both of RSA and Dilithium.

In the scenario that’s immediately before us (signing with an HSM – that’s not the only relevant scenario, but would appear to be the most constrained), the obvious costs of prehashing are:

-The cost of performing the hash function on the full message by the main CPU

-The cost of transferring this hash to the HSM (where we transfer more for larger hash functions)

> -The cost of having the HSM sign the message (which increases in size with larger hash functions)
>
> As for the first one, I believe SHA-512 is actually more efficient than SHA-256 on 64 bit CPUs; on the other hand, if we consider smaller microcontrollers (32 bit), this is not as true, but it's not as clear if smaller microcontrollers would be signing/verifying very large messages. On the other hand, with SHA3, increasing the hash size does result in a less efficient hash function.
>
> As for the second and third costs, I don't believe that the additional 256 bits or so we get when moving from SHA{23}-256 to SHA{23}-512 would result in significantly higher costs (however hearing from some HSM vendors would be good)
>
> There are alternative approaches to this (e.g. randomizing the hash on the main CPU and including that randomization factor in the signature), however those go beyond the current hash-and-sign paradigm that's in common use.