Key Derivery Functions: first notes

Any hash function is by definition a KDF, and KDF is by definition a hash function (takes random string input and return bytesteam with exact size uncorrelated with input). Using basic hash function on key is vulnerable to standard dictionary and/or using massively parallel machine.

There however are other basic cryptographic operations that function similarly and have same weakpoint to cover. One of them is key stretching.

Other random ideas:

In case of iterative function, parametrizing number of iterations, is similar to changing salt,

Secure Applications of Low-Entropy Keys (old paper as a base to easly grasp basic concepts)

<https://www.schneier.com/academic/paperfiles/paper-low-entropy.pdf>

Valid points:

Use of salt to avoid dictionary attack.

We can use a hash function multiple times (as a cascade) to increase computation time required. There is no shortcut to be made. This mechanism only partially fights parallelism as you can still calculate different inputs on many small units using FPGA/GPU.

Using long int types, makes it harder to parrarel using a lot of very cheap hardware. Wasting ram also works, but there cannot be any workarounds, or they need to increase computation time significally.

Seems like a good starting idea is to use a strong hash function, use some sort of operation (that doesn’t need any cryptographic strength) to waste RAM and use this in a cascade to extend direct calculation time.

5.2) Since 32 bit processor as cheap as sand nowdays, it would be more suitable make use of 64bit integer (since 64 bit micro processors are not that common as 32bit and are more expensive). Increase in ram use would also be sufficient.

Another problem is that while we do want our function to be called on embedded system? Then fighting parallelism, also fight it’s purpose on such systems. Another idea is to simply extend the word size, which allows for cheaper parallelization but makes number of tries much greater.

Latest found competition in 2015 found Argon2 to be best KDF at this time:

<https://www.cryptolux.org/images/0/0d/Argon2.pdf>