Notes on the Salsa20 key size

Daniel J. Bernstein *

Department of Mathematics, Statistics, and Computer Science (M/C 249)
The University of Illinois at Chicago
Chicago, IL 60607–7045
snuffle@box.cr.yp.to

Some ciphers aim for high security levels; other ciphers don't. Some ciphers aim for high speed in software; other ciphers don't. Some ciphers aim for high speed in hardware; other ciphers don't. Salsa20 simultaneously aims for a high security level, high speed in software, and high speed in hardware.

Some cipher designers sacrifice security level in an attempt to obtain the highest speed. Specifically, some ciphers use an 80-bit key, exposing themselves to 80-bit brute-force searches. More ciphers use a 128-bit key. For comparison, I recommend using 256-bit keys.

Theoreticians often propose foolish "mathematical" ciphers in which attacks are provably equivalent to integer factorization. A typical cipher of this type has key size, and time, growing as roughly the cube of the number of bits of conjectured security. Better-designed ciphers don't allow factorization attacks and have time growing much more slowly with conjectured security level; but one still expects a considerable performance gap between 80-bit security and 256-bit security.

The administrators of eSTREAM, the ECRYPT Stream Cipher Project, have imposed the following formal submission requirements: every cipher aiming for high speed in software must explain how to expose itself to a 128-bit brute-force search, and every cipher aiming for high speed in hardware must explain how to expose itself to an 80-bit brute-force search.

These are strange requirements. I can understand simplifying comparisons by requiring that key sizes be limited to the set $\{80, 96, 128, 160, 192, 256\}$; I can also understand demanding high speed; but why should we demand a low security level? Furthermore, why should this low-security-level demand be tied to the choice of platform?

The AES competition evaluated both software and hardware performance at every security level. Failing to consider the hardware performance of 128-bit ciphers, and the hardware and software performance of 256-bit ciphers, would be a surprising step backwards.

Anyway, here is how Salsa20 complies with the formal requirements. A 128-bit key is repeated to form a 256-bit key; and "32" is replaced with "16" in the Salsa20 diagonal constants, to eliminate any concerns regarding overlapping keys. An 80-bit key is zero-padded to form a 128-bit key; and "16" is replaced with "10" in the Salsa20 diagonal constants, again to eliminate any concerns

* The author was supported by the Alfred P. Sloan Foundation. Date of this document: 2005.10.02.

regarding overlapping keys. The exponents in the Salsa20 security conjecture—see the Salsa20 security document—scale linearly to 128 bits and 80 bits.

I designed Salsa20 for 256-bit keys. I'm comfortable with the 20 rounds of Salsa20 as being far beyond what I'm able to break for 256-bit keys. I'm sure that a smaller number of rounds would be safe for 128-bit keys, and that an even smaller number would be safe for 80-bit keys. However, as I wrote in the Salsa20 design document, varying the number of rounds with the key size creates two real-world problems: first, it adds cost to an implementation that supports both key sizes, especially if the implementation is fully unrolled; second, it seems to tempt users to reduce key sizes even in situations where the reduced number of rounds produces an insignificant reduction in total costs. Consequently, I am *not* specifying a smaller number of rounds for shorter keys.

The reader might be wondering about the huge discrepancy between (1) my recommendation of 256-bit keys and (2) many eSTREAM submissions that characterize 80-bit keys as safe. There are two explanations for this discrepancy. First, many cryptographers wildly overestimate the cost of a key-searching unit, estimating it as the cost of an entire PC rather than as a tiny fraction of the cost of a single chip. Second, many cryptographers ignore the fact that the attacker can further reduce costs by attacking many keys simultaneously. One can address the second problem by inserting randomness into the nonce, but this is more expensive than inserting the same randomness into the key. All of this is discussed in much more detail in my paper "Understanding brute force."

I predict that future cryptographers will settle on 256-bit keys as providing a comfortable security level. They will regard 80-bit keys as a silly historical mistake, and 128-bit keys as uncomfortably risky.