6.857
NETWORK AND COMPUTER SECURITY
PROBLEM SET 1, PROBLEM 3

JOHN WANG

1. Problem 3.a

The Keccak function (selected to be SHA-3 by the NIST competition) had a very well defined set of design goals. In particular, since the NIST defined the requirements of SHA-3 explicitly, many of the goals that Keccak attains are set specifically for SHA-3. The NIST wanted a cryptographic hash function that would replace SHA-2 and use an entirely different algorithm. They set goals broadly in the following categories:

- Security. The algorithm should have a high expected safety margin and be resistant to known attacks, such as differential crypanalysis.
- Speed. The algorithm should have fast computation time. An algorithm that is highly secure but is infeasible in practice is useless for a cryptographic standard. The NIST asked for fast runtimes for both software and hardware implementions of the algorithm.
- Ease of use. The algorithm should not have too many parameters to tune and should also be easily implemented in either software or hardware. Widespread use of a cryptographic algorithm, such as SHA-3, would require it to be simple enough to be distributed to non-experts.
- Memory efficiency. The algorithm should not require an inordinately large amount of memory in order to run. Such an algorithm, even if effective, might hinder some users with older hardware or memory constraints.

These goals influenced a number of design decisions made by the Keccak team. For instance, simplicity and functionality greatly affect the choice of the sponge construction for Keccak. The sponge construction's simplicity meant that its security bounds could be expressed in a simple way, and also, that those wanting to implement Keccak would have an easier time understanding the algorithm. The iterated permutation of Keccak was chosen in part because of its memory efficiency. Instead of using a feed-forward loop, like in many other cryptographic algorithms, Keccak's iterated permutation only required the state of the last iteration, which meant that Keccak only required a small amount of memory.

Another goal of SHA-3 was ease of use, and this led to a number of design changes by the Keccak team. For instance, the Keccak function is actually a variable-input variable-output function. This means that the user has the ability to choose the length of the output from the Keccak function. The Keccak team, however, made a deliberate decision to choose particular input-output length pairs. The reasoning behind this decision stemmed from the fact that the input length b is composed of two parameters r and c such that $b = r + c$. The capacity c governs the tradeoff between speed and security. Larger values of c cause the function to be more secure, but also decrease the function's performance. The creators of Keccak said, "the choice of the capacity value determines a ceiling to the security level that the sponge function provides and one could argue that the user usually does not have the responsibility or the expertise to make that choice." Hence, the team not only defined predetermined pairs of input and output lengths, but they also created a rule of thumb, telling the users to set the capacity as $c = 2n$ for an output message of length n.

One other consequence of these particular design goals was a method for responding to a breach in Keccak's security. Rivest originally suggested that having a tunable security parameter, i.e. increasing the number of rounds of the permutation, could add extra security in event of a breach. However, the Keccak team decided against this suggestion because of extra burden on the user and the additional implementation complexity. Instead, they chose to use the same algorithm and insert a number of waste bits so that the actual bitrate r would become $r - \delta$, essentially increasing the capacity by some δ. Again, this solution was made in the view of user friendliness, in the expectation of widespread use of SHA-3.