Problem Set 7 – Due Wed, 6 Mar 2019 at 12pm

Problem 20. Consider using the CBC MAC of a 256-bit blockcipher as a cryptographic hash function: fix a permutation $\pi = E_K$ on 256 bits for some fixed, known key K; append to M some $10^*|M|_{64}$ padding; then hash M by applying the raw CBC MAC, over π, to the padded M. Is this method collision resistant? Explain.

Problem 21. Read the specification document for an AEAD scheme. Choose one of the six schemes in the "Final portfolio" from the CAESAR contest:

`http://competitions.cr.yp.to/caesar-submissions.html`

After understanding how the scheme works, write a coherent essay that describes the high-level idea of the scheme. Your essay should be at most one page (not including any drawings, which you may lift, with credit, from any source you like). For writing this, imagine you're speaking to someone who knows cryptography well, including authenticated encryption, but he or she just never saw the particular scheme you're describing. Omit unimportant details and explain what's interesting. Your writing will be graded on how clearly/grammatically/insightfully it gets across the high-level ideas.

Problem 22. Suppose you'd like to realize an *associative array* (also called a *dictionary*) whose contents will live in the cloud. You will use operations $\text{Insert}(K, X)$ and $\text{Lookup}(K)$. The $\text{Insert}(K, X)$, also written $A[K] \leftarrow X$ where A is the associative array, associates to the *key* K the *value* X. If some other value X' had already been associated to the key K, then the new association will overwrite the old one. The $\text{Lookup}(K)$ operation, also written $A[K]$, returns the value associated to K. If there is no value associated to K, it returns a special symbol $\perp$.

Your dictionary might be huge, holding terabytes of data. You want to access it from a client, perhaps an old smartphone, with very limited memory. Thing is, you don't trust the answers you back from the cloud. After all, the *cloud* is just a feel-good word for some computer you have no control over. So your client—despite not storing the dictionary itself—should detect if the server gives you a bogus answer. The client must have a small memory footprint no matter how big the dictionary grows.

Describe a solution to this problem, explaining what Insert and Lookup do on both the client and server side. You don't have to actually implement anything, but clearly describe how a solution could work. The anticipated running time for operations, both client-side and server-side, should be $O(\lg n)$, where n is the number of items in your dictionary. In stating that bound, I am assuming that keys and values are strings whose length is bounded by some constant, say 1 MB.

Problem 23*. Let E be an n-bit blockcipher. Find a string whose CBC MAC over E you can forge without asking any queries whatsoever. Explain.

This problem is hard enough that I will consider it extra-credit. Don't feel obliged to think about it. No questions will be answered about it. The only hint I am going to give is that the string you forge is super long.