Assignment #8

Hardware Crypto, Post Quantum Crypto

Due: December 6, 2016

1. Suppose there is a compliance requirement to use a Hardware Security Module as part of your online services security across your servers. Your HSM has one key embedded in the hardware for key encryption (*key wrapping*), and supports only one active additional key operation. Once the active cryptographic operation is completed, you can load (*key unwrap*) another key to perform another operation, then load another key, and so on. Every time you load a new key, the current key in the HSM is evicted. Using the HSM's embedded HW key, design a cryptographically secure key wrap and unwrap function to store an encrypted application key on the disk: write the function API (just the interface), describe the cryptographic algorithms of your choice for key wrap and unwrap, and why you chose those algorithms. *Hint: Think about both secrecy and integrity of keys. Ignore the file I/O, HSM I/O, and HSM memory/object management aspects, and focus on crypto.*
2. Suppose that smart cards are required for user authentication during logging on to a web site in a browser, such as mutual TLS. The server has its private key and certificate, and the client private key and its certificate are stored in the user's smart card. The client browser would pick an appropriate certificate and a private key when there are multiple private keys and certificates on the smart card. How would you pick an appropriate key and a certificate on the smart card? If you chose to ask the user to pick a certificate, what information would you present to the user to pick an appropriate key and a certificate? *Hint: the smart card must sign a message as part of the TLS protocol. Think about private keys with corresponding public key certificates on the smart card, and assume each certificate has validity and key usage fields populated.*
3. A Quantum Key Distribution (QKD) protocol includes a fiber channel between Alice and Bob to send quantum states of light, a public communication channel such as the Internet, and a key exchange protocol from the exchanged quantum states. Alice and Bob run this QKD protocol to compute the same AES secret key. However, QKD does not authenticate the peers as mentioned in the class, so Alice and Bob have no clue who they exchanged a key with; we want to fix that. Assume QKD as a black-box key exchange protocol, and build a user authentication protocol around it. *Hint: You may want to think about user signature keys and public key certificates like TLS.* Bonus (0 points): A very short argument about key exchange with QKD vs. traditional [EC]DH.
4. We are going to factor integer $N = 21$ (where we know $N = pq,\ p = 3, q = 7)$ using Miller's observation by finding periods (same as order). *Order: smallest integer r such that* $x^r = 1\ mod\ N$. As you might recall, this means $x^r - 1 = 0\ mod\ N$, or $(x^{r/2} - 1)(x^{r/2} + 1) = 0\ mod\ N$ for even r, or $x^r - 1$ is divisible by N. If $x|N$, then x is a factor, so pick numbers $x \in \{2, 5\}$, and compute $f_r(x) = x^r mod\ N$ for $r = 1, 2, \ldots N - 1$ till you find the period for each x, discard odd periods if any. Compute $gcd(x^{\frac{r}{2}} + 1, N)$ and $gcd(x^{\frac{r}{2}} - 1, N)$ to see if you find a factor. If you do, find the other factor by division.

5. Recall that the existence of a quantum computer is likely to render RSA algorithm insecure with today's key lengths. We'd like to know replacement RSA key length to match today's n=2048 bit RSA key in the presence of a quantum computer. Shor's and Dan's algorithms tell us about complexity: GNFS: $O(e^{cn^{\frac{1}{3}}(\log n)^{2/3}})$, Shor: $O(n^2(\log n)(\log\log n))$. We will approximate Shor's complexity as $O(n^3)$. Assume $c = 2$, and unit multiplier for the big-O notation. Compute the equivalent RSA modulus length to yield equivalent security under Shor's algorithm on a hypothetical quantum computer. *Hint: Solve for n in Shor's to match GNFS complexity for 2048 bits.*