University of Maryland
CMSC858K — Cryptography
Professor Jonathan Katz

Problem Set 4

Due at *beginning* of class on April 10 (due to the midterm)
Note: do not wait until April 3 to work on this!

1. **Basing identification on RSA.** In class we discussed public-key identification schemes based on the discrete logarithm problem. Here, we develop a public-key scheme based on the hardness of the RSA problem.

   The scheme proceeds as follows: A prover $\mathcal{P}$ generates his public key by choosing a modulus $N = pq$ (where p, q are distinct, k-bit primes) and a *prime* exponent e for which $\gcd(e, \varphi(N)) = 1$. They also choose $x \leftarrow \mathbb{Z}_N^*$ and compute $y = x^e \bmod N$. The public key is $\langle N, e, y \rangle$ and the private key is x.

   In an execution of the protocol, the prover begins by choosing random $r \leftarrow \mathbb{Z}_N^*$ and sending $A = r^e \bmod N$ to the verifier. The verifier chooses and sends a random challenge $b \leftarrow \mathbb{Z}_e$. Finally, the prover responds with $C = x^b r \bmod N$ and the verifier accepts iff $C^e \stackrel{?}{=} y^b A \bmod N$.

   - Show that the scheme is *correct*; that is, if the prover and verifier both act honestly then the verifier always accepts.
   - Prove the following lemma:

     Given $\tilde{C}, e, y, N$ and $\tilde{b} > 0$ such that (1) $\tilde{C}^e = y^{\tilde{b}} \bmod N$ and (2) $\gcd(e, \tilde{b}) = 1$, it is possible to efficiently compute x such that $x^e = y \bmod N$.

     Hint: use the fact that if $\gcd(e, \tilde{b}) = 1$ then it is possible to efficiently compute integers α, β such that $\alpha \cdot e + \beta \cdot \tilde{b} = 1$.
   - Prove that any PPT adversary attacking this identification scheme via a weak attack cannot succeed with probability noticeably better than $1/e$, assuming the RSA problem is hard. You may use the lemma from the previous part.
   - Prove that any PPT adversary attacking this identification scheme via a passive attack cannot succeed with probability noticeably better than $1/e$, assuming the RSA problem is hard. You may use results from any previous part of the problem.
   - In practice, for reasons of efficiency $e = 3$ is often chosen. Do you recommend that choice of parameters here?

2. **A variant of the Lamport scheme.** We improve (slightly) on the Lamport one-time signature scheme we gave in class. Recall that the Lamport scheme requires a public key consisting of 2ℓ elements in order to sign messages ℓ bits long. Since signing ℓ-bit messages can also be viewed as signing one message out of 2^ℓ possible messages,

we can view the efficiency of the Lamport scheme in the following equivalent way: if there are n elements in the public key, we can sign one message out of $2^{n/2}$ possible messages.

We now show one way to improve this. Consider the following scheme which allows signing one message out of 6 possible messages: the public key consists of four elements (y_1, y_2, y_3, y_4). The secret key consists of their inverses $(x_1 = f^{-1}(y_1), \ldots)$. We assume the 6 possible messages are ordered in advance in some publicly known way (i.e., lexicographically). To sign message 1, send the pair (x_1, x_2); to sign message 2, send the pair (x_1, x_3); $\ldots$; to sign message 6, send the pair (x_3, x_4). Each signature consists of a pair of elements. Verification is done in the obvious way.

- Prove the security of the above scheme for signing one of a possible 6 messages. How does the security reduction you obtain here compare to what was obtained in class for the Lamport scheme?
- Sketch the generalization of the above scheme for when you have n elements in the public key (no proof of security is necessary).
- What is the complexity of this generalization? In other words, given a public key containing n elements, how large is the space of possible messages you can sign? Try to generalize the scheme so as to obtain the best possible result.

3. **Representations of group elements, and applications.** Let $\mathbb{G}$ be a cyclic group of order q, where q is a prime. Assume also that the discrete logarithm problem is hard in $\mathbb{G}$. Let $g_1, g_2, g_3 \in \mathbb{G}$ be generators. For any element $h \in \mathbb{G}$, we say that (x, y, z) is a *representation* of h with respect to g_1, g_2, g_3 iff $h = g_1^x g_2^y g_3^z$.

   - For a given element $h \in \mathbb{G}$, how many distinct representations (x, y, z) of h are there with respect to g_1, g_2, g_3? How many of these satisfy $x = \tilde{x}$, for some fixed $\tilde{x} \in \mathbb{Z}_q$? How many satisfy $x = \tilde{x}, y = \tilde{y}$ for fixed $\tilde{x}, \tilde{y} \in \mathbb{Z}_q$?
   - Show that, assuming the discrete logarithm problem is hard in $\mathbb{G}$, no PPT algorithm can take g_1, g_2, g_3 as input and output $h \in \mathbb{G}$ along with two distinct representations of h (with respect to g_1, g_2, g_3).
   - Assume that we can represent elements in $\mathbb{G}$ by strings of length $|q| + 1$. Define the function $H_{g_1,g_2,g_3} : \mathbb{Z}_q^3 \to \mathbb{G}$ by $H_{g_1,g_2,g_3}(x, y, z) \stackrel{\text{def}}{=} g_1^x g_2^y g_3^z$. Argue that this is a *collision-resistant hash function* (when g_1, g_2, g_3 are randomly chosen).