CS 359C – Classics of Cryptography April 12, 2017

Lecture 2: Even-Mansour Cipher

Instructors: Henry Corrigan-Gibbs, David Wu *Scribe: Dylan Liu*

1 Review

We covered negligible functions, efficient algorithms, and security parameters last week. A pseudo-random generator (PRG) is an efficiently computable function $G : \{0,1\}^\lambda \to \{0,1\}^{\ell(\lambda)}$ that takes a λ bit string and returns a longer string of length $\ell(\lambda)$ with $\ell(\lambda) \gg \lambda$. The resulting string should be computationally indistinguishable from a string that is sampled from a random distribution. That is, $\{s \xleftarrow{R} \{0,1\}^\lambda : G(s)\} \approx \{z \xleftarrow{R} \{0,1\}^{l(\lambda)}\}$. With the Blum-Micali construction, a PRG that extends a string by one bit can be used to extend it by polynomial bits. A proof was done last week using a hybrid argument (triangle inequality).

A pseudo-random function (PRF) is an efficiently computable function $F : \mathcal{K} \times X \to Y$ that, given oracle access, is computationally indistinguishable from a truly random function. That is, for any efficient adversary $\mathcal{A}$, we have that

$$|\Pr[k \xleftarrow{R} \mathcal{K} : \mathcal{A}^{F(k,\cdot)}(1^\lambda) = 1] - \Pr[f \xleftarrow{R} \text{Funs}[X,Y] : \mathcal{A}^{f(\cdot)}(1^\lambda) = 1]| = \text{negl}(\lambda).$$

A pseudorandom permutation (PRP) is defined similarly as a permutation.

$$|\Pr[k \xleftarrow{R} \mathcal{K} : \mathcal{A}^{F(k,\cdot)}(1^\lambda) = 1] - \Pr[f \xleftarrow{R} \text{Perms}[X] : \mathcal{A}^{f(\cdot)}(1^\lambda) = 1]| = \text{negl}(\lambda).$$

A PRG can be generated from a PRF F by evaluating F on a number of points. For instance, defining the PRG to be $G(k) = F(k,0)|F(k,1)|\ldots|F(k,n)$ is a secure PRG. This is also known as counter mode. A PRP can be generated from a PRF by using a 3-round Feistel Network. A PRP can also be used as a PRF by the PRF switching lemma, which roughly states that if insufficiently many queries are made to detect a collision a PRF is indistinguishable from a PRF. Lastly, a PRF can be constructed from a PRG by using a GGM tree. Putting everything together, we have the following implications

$$\text{PRG} \Longleftrightarrow \text{PRF} \Longleftrightarrow \text{PRP}.$$

2 Format-preserving encryption

Historically, PRPs was considered one of the most important notions for applications. Encryption and decryption, for instance, is (at an intuitive level) a process of applying a function (encrypting) and then computing its inverse (decrypting). However, today, we know that almost all applications that can be achieved using PRPs can actually be achieved using any PRFs.

One exception to this is format-preserving encryption, which can be constructed from PRPs but does not seem to follow from PRFs. This is an encryption scheme that "preserves" the underlying structure of the message to be encrypted such as mapping a credit card number to another credit card number. This is interesting since there are many legacy systems that cannot use n-bit strings, but can use other credit card numbers.

So how do we construct format-preserving encryption from PRPs? It's not actually obvious that a PRP on n-bit strings can actually be used for credit card numbers since credit card numbers have certain structures, such as the last digit somehow checking the preceding digits.

To encrypt credit card numbers, we proceed by a concept known as cycle walking (Shroeppel, Orman). Suppose we have an efficient algorithm $\mathsf{Valid} : \{0,1\}^n\{0,1\}\{0,1\}$ that checks whether a given string encodes a valid credit number or not (it outputs 1 if the input is a credit number and 0 otherwise). Then, the space of credit card numbers is defined as $\mathcal{C} = \{x : \mathsf{Valid}(x) = 1\} \subset \{0,1\}^n$. Now, suppose we have a PRP $F : \mathcal{K} \times \{0,1\}^n \to \{0,1\}^n$ (a permutation on n-bit strings to n-bit strings), and we wish to define a PRP $F_{cc} : \mathcal{K} \times \mathcal{C} \to \mathcal{C}$ (a permutation on $\mathcal{C}$ to $\mathcal{C}$).

Lemma 1 *Given F, $\mathsf{Valid}(\cdot)$, we can construct a PRP F_{cc} as long as $\frac{|\mathcal{C}|}{2^n} > \frac{1}{poly(n)}$.*

Proof. The idea is simple. Given a credit number c, we repeatedly apply F to until we get a valid credit card number.

```
• $c' \leftarrow c$
• While $\mathsf{Valid}(c') = 0$:
•     $c' \leftarrow F(k, c')$
• Return $c'$.
```

The PRF F will map c' to a seemingly random point string in $\{0,1\}^n$. By the condition $\frac{|\mathcal{C}|}{2^n} > \frac{1}{\text{poly}(n)}$, the algorithm will terminate in expected polynomial time. Security can be proven in a straightforward way by replacing $F(k, \cdot)$ by a truly random permutation, and is left as an exercise.

3 Even-Mansour Cipher

Cryptography is the study of using computationally hard problems to construct secure systems. More precisely, in cryptography, we construct systems and show that they are secure by giving reductions to computationally hard problems. However, since we cannot actually prove that a certain computational problem is actually hard (for instance, we cannot show $\mathsf{P} \neq \mathsf{NP}$), this does not unconditionally prove that our system is secure. So how do we actually show that our cryptographic constructions are secure? One way is to work with a set of "standard" computational assumptions that seem to hold and have withstood the test of time (have not yet been shown to be false for some time) such as "DES is a secure PRP", "factoring is hard", etc. Another way to show that our scheme is secure is to actually change the model. In the standard model, we assume that the challenger and an adversary are in the form of standard Turing machines. However, today, we examine a model where the challenger and the adversary are Turing machines that are augmented with a peripheral device that implements a random permutation. At any point in the computation, the Turing machine can query an input x, and the peripheral device will return $\pi(x)$ and $\pi^{-1}(x)$. This model is called the random permutation model (RPM) and is strictly stronger than the standard model. We will see that in this model, PRPs can be constructed unconditionally (without any computational assumption). The construction is known as the Even-Mansour Cipher.

The Even-Mansour Cipher has a public random permutation $\pi : \{0,1\}^n \to \{0,1\}^n$, and an encryption function $E : \{0,1\}^n \times \{0,1\}^n \to \{0,1\}^n$. The encryption scheme is $E(k,m) = k \oplus \pi(k \oplus m)$, and the decryption scheme proceeds similarly as $D(k,c) = k \oplus \pi^{-1}(k \oplus c)$. This construction is interesting because it's as simple as possible, and it's also the basis for AES. AES has a fixed permutation, and proceeds by 10 rounds. In each round, the input is xor-ed with a key k_i, then the permutation π is applied to that result to produce the input for the next round.

Theorem 2 *(Kilian-Rogaway, 4.14 in Boneh-Shoup) Let $A^{\pi,\pi^{-1}}$ be an adversary with oracle access to π, π^{-1} making at most Q_{enc} encryption queries and Q_π permutation queries. Then*

$$PRPAdv[EM^{\pi,\pi^{-1}}, A^{\pi,\pi^{-1}}] \leq \frac{2Q_{enc}Q_\pi}{2^n} \leq \frac{poly(n)}{2^n} = negl(n). \tag{1}$$

This holds even for exponential-time adversaries, as long as $Q_{enc}Q_\pi$ is polynomial in n.

Proof. In this security game, the adversary can submit queries of the form (type, x_i), and the challenger responds with a y_i. The type is of the form type $\in \{\pi, \pi^{-1}, E\}$. We define three possible games. In game 0, the challenger uses the EM cipher, so this is the real construction. Game 1 is a variant of game 0 with two permutations π, π_E with disjoint domains and disjoint ranges. Think of π as the public permutation and π_E as the private permutation. Game 2 is a relaxation of game 1 in which π, π_E are no longer are required to have disjoint domains and disjoint ranges. The challenger uses an independent permutation π_E in response to E queries, but she still uses the permutation π in response to π, π^{-1} queries. Let $p_i = \mathbb{P}[\text{Adv outputs 1 in game } i]$. Note that the adversary does not get direct access to π in the RPM, but rather can access it only through the challenger. We claim that $|p_0 - p_1| = 0$ and $|p_1 - p_2| \leq \frac{2Q_{\text{enc}}Q_\pi}{2^n}$.

Algorithm 1: `Game 0`

```
$\pi \leftarrow \varnothing$
$k \xleftarrow{R} \{0,1\}^n$
**if** *the query is* $(E, m)$ **then**
    $\alpha \leftarrow m \oplus k$
    **if** $\pi(\alpha)$ *is undefined* **then**
        $\pi(\alpha) \xleftarrow{R} \{0,1\}^n \setminus \text{Range}(\pi)$
    **return** $k \oplus \pi(\alpha)$
**if** *the query is* $(\pi, \alpha)$ **then**
    **if** $\pi(\alpha)$ *is undefined* **then**
        $\pi(\alpha) \xleftarrow{R} \{0,1\}^n \setminus \text{Range}(\pi)$
    **return** $\pi(\alpha)$
**if** *the query is* $(\pi^{-1}, \beta)$ **then**
    **if** $\pi^{-1}(\beta)$ *is undefined* **then**
        $\pi^{-1}(\beta) \xleftarrow{R} \{0,1\}^n \setminus \text{Domain}(\pi)$
    **return** $\pi(\beta)$
```

Algorithm 2: `Game 1`

```
$\pi, \pi_E \leftarrow \varnothing$
$k \xleftarrow{R} \{0,1\}^n$
**if** *the query is* $(E, m)$ **then**
    $\alpha \leftarrow m \oplus k$
    **if** $\pi_E(\alpha)$ *and* $\pi(\alpha)$ *are undefined* **then**
        $\pi_E(\alpha) \xleftarrow{R} \{0,1\}^n \setminus (\mathrm{Range}(\pi_E) \cup \mathrm{Range}(\pi))$
        **return** $k \oplus \pi_E(\alpha)$
    **else**
        **return** $k \oplus \pi_\mu(\alpha)$, where $\pi_\mu \in \{\pi, \pi_E\}$ is the unique choice such that $\pi_\mu(\alpha)$ is defined
**if** *the query is* $(\pi, \alpha)$ **then**
    **if** $\pi_E(\alpha)$ *and* $\pi(\alpha)$ *are undefined* **then**
        $\pi(\alpha) \xleftarrow{R} \{0,1\}^n \setminus (\mathrm{Range}(\pi) \cup \mathrm{Range}(\pi_E))$
        **return** $\pi(\alpha)$
    **else**
        **return** $\pi_\mu(\alpha)$, where $\pi_\mu \in \{\pi, \pi_E\}$ is the unique choice such that $\pi_\mu(\alpha)$ is defined
**if** *the query is* $(\pi^{-1}, \beta)$ **then**
    **if** $\pi_E^{-1}(\beta)$ *and* $\pi^{-1}(\beta)$ *are undefined* **then**
        $\pi^{-1}(\beta) \xleftarrow{R} \{0,1\}^n \setminus (\mathrm{Domain}(\pi) \cup \mathrm{Domain}(\pi_E))$**return** $\pi^{-1}(\beta)$
    **else**
        **return** $\pi_\mu^{-1}(\beta)$, where $\pi_\mu \in \{\pi, \pi_E\}$ is the unique choice such that $\pi_\mu^{-1}(\beta)$ is defined
```

π and π_E have very few collisions, which are bad when they are of the form $\alpha_j = m_i \oplus k$ or $\beta_j = c_i \oplus k$. For a fixed key k, $\mathbb{P}$[one E query and one π/π^{-1} query are bad] $\leq \frac{2}{2^n}$. By the union bound, the probability that any pair is bad is at most $\frac{2Q_{\text{enc}}Q_\pi}{2^n}$. Then $|p_1 - p_2| \leq \frac{2Q_{\text{enc}}Q_\pi}{2^n}$, so $\text{PRPAdv}[A^\pi, EM^\pi] \leq \frac{2Q_{\text{enc}}Q_\pi}{2^n}$.

Algorithm 3: `Game 2`

```
$\pi, \pi_E \leftarrow \varnothing$
$k \xleftarrow{R} \{0,1\}^n$
if the query is $(E, m)$ then
    $\alpha \leftarrow m \oplus k$
    if $\pi_E(\alpha)$ is undefined then
        $\pi_E(\alpha) \xleftarrow{R} \{0,1\}^n \setminus \text{Range}(\pi_E)$
    return $k \oplus \pi_E(\alpha)$
if the query is $(\pi, \alpha)$ then
    if $\pi(\alpha)$ is undefined then
        $\pi(\alpha) \xleftarrow{R} \{0,1\}^n \setminus \text{Range}(\pi)$
    return $\pi(\alpha)$
if the query is $(\pi^{-1}, \beta)$ then
    if $\pi^{-1}(\beta)$ is undefined then
        $\pi^{-1}(\beta) \xleftarrow{R} \{0,1\}^n \setminus \text{Domain}(\pi)$
    return $\pi^{-1}(\beta)$
```