Winter 2018 CS 485/585 Introduction to Cryptography

Lecture 7

Portland State University
Lecturer: Fang Song

Jan. 31, 2018

Draft note. Version: February 4, 2018. Email fang.song@pdx.edu for comments and corrections.

Agenda

- (Last time) PRF-OTP, Message authentication;
- PRF-MAC, Domain-extension
- Review HW1/Quiz1

Logistics

- Don't copy solutions from online forums.

MAC continued

Review the issue of data integrity and the definition of a secure MAC: existentially unforgeable under chosen-message-attacks.

- Replay attacks. The security definition itself does not prevent a simple attack in practice: copy a previous message-tag pair and resend it to an honest user at a later point. Again, consider the greedy ebay seller Mr. M, what if he keeps forwarding the money transfer request?

  Two common techniques for thwarting such attacks:

  1. *Sequence number (counter)*
  2. *time-stamps* $t = S_k(\mathsf{TIME}\|m)$ and verify $V_k(\mathsf{TIME}\|m, t) = 1$, and TIME is "recent".

  Both need synchronization to some extend.

- canonical verification.

A fixed-length eu-cma-secure MAC from PRFs

Theorem 1 ([KL: Thm. 4.6])**.** Π *is an eu-cma MAC (for messages of length n).*

Let $F_k : \{0,1\}^n \to \{0,1\}^n$ be a PRF, construct MAC scheme $\Pi = (G, S, V)$

- $G(1^n)$: $k \leftarrow \{0,1\}^n$ (random key for F_k).
- $S(m)$: $t := F_k(m)$.
- $V(m,t)$: compute $t' := F_k(m)$ and check $t' \stackrel{?}{=} t$. (Canonical verification)

Figure 1: A fixed-length MAC from any PRF. PRF-MAC

Intuitively forging in this scheme amounts to predict the output of a PRF on a new point, which should be infeasible, especially if we think about a truly random function. Consider a variant $\tilde{\Pi}$ where we use a truly random function $f \leftarrow \mathcal{F}$. Let $\mathcal{A}$ be any adversary trying to produce a forgery. Let $q(n)$ be an upper bound on its number of MAC-queries. For any candidate forgery (m^*, t^*), where m^* is a new message, $y := f(m^*)$ is sampled uniformly at random (by the "sample-on-the-fly" interpretation of a truly random function). Therefore y would differ from t^* except with probability 2^{-n}.

Lemma 2. *For any* $\mathcal{A}$*,* $\Pr[\textit{Mac-forge}_{\mathcal{A},\tilde{\Pi}}(n)] \leq 2^{-n}$*.*

Then we show that switching back to a PRF, does not make the adversary's life any easier based on the security of PRF. Therefore we conclude that PRF-MAC (Fig. 1) is eu-cma.

Lemma 3. $$\Big|\underbrace{\Pr[\textit{Mac-forge}_{\mathcal{A},\Pi}(n) = 1]}_{p_{\mathcal{A},\Pi}} - \underbrace{\Pr[\textit{Mac-forge}_{\mathcal{A},\tilde{\Pi}}(n) = 1]}_{p_{\mathcal{A},\tilde{\Pi}}}\Big| \leq \mathrm{negl}(n).$$

Proof. For any $\mathcal{A}$, we construct a distinguisher D and show that

$$\left|p_{\mathcal{A},\Pi} - p_{\mathcal{A},\tilde{\Pi}}\right| \leq \Big|\underbrace{\Pr[D^{F_k}(1^n) = 1]}_{p_{D,k}} - \underbrace{\Pr[D^{f}(1^n) = 1]}_{p_{D,f}}\Big| \leq \mathrm{negl}(n)\,.$$

Distinguisher D: given 1^n and oracle access $\mathcal{O} : \{0,1\}^n \to \{0,1\}^n$:

1. Run $\mathcal{A}(1^n)$. Whenever $\mathcal{A}$ makes a MAC-query on m, forward m to $\mathcal{O}$ and return $t := \mathcal{O}(m)$.
2. $\mathcal{A}$ outputs (m^*, t^*) in the end. Let $Q = \{m_i\}$ be the list of $\mathcal{A}$'s MAC-queries. D does the following
   a) Query $\mathcal{O}$ with m^* and obtain $\hat{t} := \mathcal{O}(m^*)$.
   b) Output 1 iff. both $\hat{t} = t$ and $m^* \notin Q$ hold.

Observe that

- if $\mathcal{O}$ is truly random: then $\mathcal{A}$ sees exactly as in the forgery game $\mathsf{Mac\text{-}forge}_{\mathcal{A},\tilde{\Pi}}(n)$. Therefore D outputs 1 iff. $\mathcal{A}$ produces a valid forgery (i.e. succeeds in $\mathsf{Mac\text{-}forge}(n)$). We have $p_{D,f} = p_{\mathcal{A},\tilde{\Pi}}$.
- similarly, if $\mathcal{O} = F_k$ is pseudorandom, we see that $p_{D,k} = p_{\mathcal{A},\Pi}$.

Thus $|p_{\mathcal{A},\Pi} - p_{\mathcal{A},\tilde{\Pi}}| = |p_{D,k} - p_{D,f}| \leq \text{negl}(n)$. □

MAC: domain-extension

In practice, our block ciphers work on a data block of small length, e.g. 128-bit, how do we MAC long messages? We will discuss two general approaches:

1. Hash-and-MAC paradigm. Apply a hash function to "compress" the input string to a shorter one that fits your MAC. [1]
2. Direct approach: domain extension. (Below)

[1] NEXT LECTURE

Given MAC on short inputs, construct a MAC on long inputs.

Natural ideas (that often fail).

1. block-by-block? *reordering attack*
2. including block index? $t_i := S(i\|m[i])$ *truncation attack* [2]
3. including message length in each block? $t_i = S(\ell\|i\|m[i])$.

   mix-&-match attack. Consider

$$m = m[1]\|m[2], t = t_1, t_2\,;$$
$$m' = m'[1]\|m'[2], t' = t'_1, t'_2\,.$$

   Then $t_1\|t'_2$ is a valid tag for $m[1]\|m'[2]$.
4. additional random identifier. $S'(m[i]) := S(r\|\ell\|i\|m[i])$. This works, but very inefficient! We will not discuss further about it. Read [KL: Thm 4.8] for details.

[2] message length is not included in the tag; how about authenticating message length in last block? it doesn't help.

Domain extension for PRF*s.* We ask a slightly different question:

Given PRF on short inputs, construct a PRF on long inputs. i.e., given $F : X \to Y$ how to get $F' : X^{\leq \ell} \to Y'$, for $\ell \geq 1$? (assume $X = Y = Y' = \{0,1\}^n$)

If this is possible, then we will just use the PRF on longer messages to achieve message authentication on long messages. We show this is indeed possible.

Cascade and encrypted cascade (NMAC)

Cascade construction.

$$t_1 = F_k(m[1]), t_i = F_{t_{i-1}}(m[i]), \quad \text{and only output } t_d\,.$$

Draw Cascade

It is a secure PRF if the input length is fixed. Unfortunately, you can break cascade by *extension attacks*. [3]

[3] knowing $m, t = \mathsf{CASCADE}_F(m)$, can compute $\mathsf{CASCADE}_F(m\|m')$ on any m'. [KL: Exercise 4.13]

Reading material. The extension attack can be cast into an distinguisher that tells apart cascade from truly random, since knowing $\mathsf{CASCADE}_F(x)$,$\mathsf{CASCADE}_F(x\|x')$, i.e., input strings that share x as their prefix, becomes predictable. The issue is that two messages could share the same prefix. If we exclude such attacks, cascade does becomes secure.

Definition 4 (Prefix-free set & algorithm)**.** A set of strings $P \subseteq (\{0,1\}^n)^*$ is *prefix-free* if it does not contain the empty string (i.e. $\epsilon \notin P$), and no string $x \in P$ is a prefix of any other string $x' \in P$. We call algorithm D with oracle access to f *prefix-free* if D only queries on a prefix-free set.

Theorem 5. *If F is a* PRF*, then* $\mathsf{CASCADE}_F$ *is a* PRF *against any prefix-free PPT distinguisher D.*

In particular if we fixed the message length to be $\{0,1\}^{n\cdot\ell}$ for any ℓ, then the prefix-free constraint is trivally true because no string can be a prefix of another string of the same length. As an immediate consequence, we have

Corollary 6. *If F is a* PRF*, then* $\mathsf{CASCADE}_{F,\ell}$ *is a eu-cma MACs for messages of length $\{0,1\}^{n\cdot\ell}$ for any fixed $\ell \geq 1$.*

To obtain a fully secure PRF, a natural idea would be to introduce an encoding mechanism that ensures prefix-freeness (*prefix-free encoding*). Some examples[4]

[4] Read more on Boneh-Shoup Sect. 6.6.

- prepending message-length. Not practical since it's not suitable for data streams.
- stop bits. $m[1]\|0, m[2]\|0, \ldots, m[d]\|\mathbf{1}$. Inefficient.
- randomized encoding. NIST standard: **CMAC**. CBC with randomized prefix-free encoding.

Encrypted Cascade a.k.a ***NMAC*** *(Nested MAC).* A variant of it (using a hash function instead of a PRF in the cascade construction) called HMAC is widely used in the Internet (rfc2104).

$$\mathsf{ECAS}_{k_1,k_2}(\cdot) := F_{k_2}(\mathsf{CASCADE}_{F_{k_1}}(\cdot))\,.$$

Theorem 7. *NMAC* ECAS_{k_1,k_2} *is a* PRF.

Draw NMAC diagram. Formal proofs are beyond the scope of this course. Read Boneh-Shoup Chapter 7 if interested. Note that both are **streaming** MACs, since we do not need to know the message length ahead of time.

CBC-MAC. Read CBC-MAC and do HW problem. Come back in a future lecture.