# Winter 2018 CS 485/585 Introduction to Cryptography

## Lecture 17

Portland State University *Mar. 6, 2018*
Lecturer: Fang Song

Draft note. Version: March 8, 2018. Email fang.song@pdx.edu for comments and corrections.

*Agenda*

- (Last time) Diffie-Hellman, ElGamal, Hybrid Encryption
- Hash-based Signature
- Random-oracle
- PKC with RO: OAEP and FDH

## *Hash-based signature*

Three general approaches to digital signature:

1. **Trapdoor permutation + Full-Domain-Hash**. It can be viewed as an instantiation of the hash-and-sign paradigm. However, its provable security relies on an idealized model. Example: RSA-FDH. We will see later in class.
2. **Identification protocol + Fiat-Shamir transformation**. Some famous examples include: Digital signature algorithm (DSA) and Elliptic Curve Digital Signature Algorithm (ECDSA), which are part of the NIST Digital signature standard (DSS) and the security is based on the discret logarithm problem; Schnorr signature.
3. **Hash-based: One-time-signature+Merkle Tree**. Somewhat surprisingly, signature schemes can be constructed on cryptographic hash functions (actually one-way functions suffice). Note that public key encryption seems to need hard problems with algebraic structures (e.g., number-theoretical assumptions such as RSA or the abstract trapdoor permutations).

We discuss the hash-based approach now. This is obtained via two steps:

$$\begin{aligned} & \mathsf{OWF}(\text{One-way functions}) \\ \xrightarrow{a}\ & \mathsf{OTS}\ (\text{One-time signature}) \\ \xrightarrow{b}\ & \text{full-fledged signature} \end{aligned}$$

(a) Building a one-time signature scheme from one-way functions.

(b) Using the Merkle-tree technique to convert a one-time signature scheme to a full-fledged scheme that can sign unbounded many messages. The resulting scheme is stateful (some internal configuration needs to be recorded and updated dynamically with every signature). This can be removed using a pseudorandom function (which can be constructed based on OWF), getting a stateless signature scheme.

### *Lamport's One-Time-Signature (OTS)*

We don't define OTS formally. The only distinction from standard (EUCMA) secure signature is that the adversary only gets to ask a signing oracle once.

Let $H : \{0,1\}^* \to \{0,1\}^*$ be a function. Construct $\Pi = (G, S, V)$ for messages of length $\ell = \ell(n)$

- $G$:
  - for $i = 1, \dots, \ell$ choose random $x_{i,0}, x_{i,1} \leftarrow \{0,1\}^n$.
  - compute $y_{i,0} = H(x_{i,0})$ and $y_{i,1} = H(x_{i,1})$.
  - Output

$$pk = \begin{pmatrix} y_{1,0} & y_{2,0} & \dots y_{\ell,0} \\ y_{1,1} & y_{2,1} & \dots y_{\ell,1} \end{pmatrix};$$

$$sk = \begin{pmatrix} x_{1,0} & x_{2,0} & \dots x_{\ell,0} \\ x_{1,1} & x_{2,1} & \dots x_{\ell,1} \end{pmatrix}.$$

- $S$: on message $m = m_1 \dots m_\ell$, output

$$\sigma = (x_{1,m_1}, \dots, x_{\ell,m_\ell}).$$

- $V$: on input $(m = m_1 \dots m_\ell, \sigma = (x_1, \dots, x_\ell))$, accept iff. $H(x_i) = y_{i,m_i}$.

Figure 1: One-time signature by Lamport

**Theorem 1.** *If $H$ is one-way, then $\Pi$ is a secure* OTS.

*Proof idea.* To forge, $\mathcal{A}$ has to invert one of $y_{i,b}$, which breaks the one-way property. □

### *Getting full signature using Merkle-tree*

Let $\Sigma = (G, S, V)$ be a secure OTS that can sign messages that are twice as long as its public key, i.e. $|m| = 2|pk|$. [1]

[1] Lamport's scheme does not immediately provide this feature. But this can be achieved by compressing a long message with a universal-one-way hash function. UOWHF in turn can be constructed from a OWF.

How about start with $\{0,1\}$ again? Write down root $(pk, sk)$ and two nodes representing two messages to be signed. Generate fresh keys for each node, and sign them. But to verify, need to include the public keys as well. How to prevent arbitrary person doing so? We haven't used $pk, sk$ yet, how about sign both? OK! Here is the signature we've obtained. How to verify? use $pk$ to. How about $\{0,1\}^2$? General form $(\sigma_m, \{pk_i, pk_i^{sibling}, \sigma(sk_i^{parent}, pk_i, pk_i^{sibling})\}_{i \in \text{path } (root \to leaf_m)})$

Merkle-tree diagram goes here.

We construct a new signature scheme $\Sigma' = (G', S', V')$ that will be secure for signing multiple messages from $\Sigma$. Basically we maintain a tree of height $h$ to sign all $h$-bit messages:

- we lable every left edge 0 and every right edge 1, and each node of the tree is labeled with the prefix of the path from the root. The root is denoted by $\epsilon$. Each leaf (or rather path from root to leaf) corresponds to a message. For example the left-most leaf node corresponds to string $\underbrace{0 \dots 0}_{h \text{ bits}}$.
- each node is associated with a $\mathsf{OTS}$ key-pair $(pk_p, sk_p)$ indexed by the path from the root to itself. Denote it $(pk_\epsilon, sk_\epsilon)$ at the root. They are generated independently and adaptively, which is part of the *state* that the signing algorithm maintains and keeps updating whenever producing a new signature.
- Signing a message $m$ consists of
  1) $\sigma_0 := S(sk_m, m)$, signing $mf$ using the $\mathsf{OTS}$ signing algorithm and the leaf secret key.
  2) $\sigma_1 := (\mathsf{auth}^{(0)}, \dots, \mathsf{auth}^{(h-1)})$, an "authentication" list that signs the two public keys of the children of each node on the path from root to leaf $m$. Specifically, each $\mathsf{auth}^j$ is associated with the node of $m_j$ ($j$th prefix of the message $m$) and contains the public keys at $m_j$ and its two children (i.e., $pk_{m_j}$ and $(pk_{m_j0}, pk_{m_j1})$) as well as the signature $S(sk_{m_j}, (pk_{m_j0}, pk_{m_j1}))$. The Signer generates new key pairs of $\mathsf{OTS}$ $\Sigma$ when necessary, and they are appended in the state that the Signer maintains.
- To verify $(m, (\sigma_0, \sigma_1))$, we first verify the authentication path specified by $\sigma_1$. Namely for every $j = 0, \dots, h-1$, we check if $V(pk_{m_j}, (pk_{m_j0}, pk_{m_j1}), \sigma_1^j = \mathsf{auth}^j)$. If this passes, we accept if $V(pk_m, m, \sigma_0) = 1$.

Intuitively, $\Sigma'$ is secure because at any time a secret key at any node signs at most one message, which is either an actual message at the leaf node or a pair of public keys at two child nodes. In recent years many variants have been developed that are more efficient in terms of time complexity and sizes of verification key and signatures. For instance, Winternitz-$\mathsf{OTS}$ and some optimized Merkle-tree constructions have gain popularity[2].

[2] See e.g., "XMSS – A Practical Forward Secure Signature Scheme based on Minimal Security Assumption" `https://huelsing.files.wordpress.com/2013/05/mssgesamt.pdf`, and its followup works).

## *The Random Oracle Heuristic*

An idealized model to design and analyze crypto constructions involving hash functions. Hash functions are so efficient, but often we cannot prove security based solely on Collision resistance or other reasonable assumptions. Instead, people settle with a "middle-land".

Random oracle $\mathcal{O}(\cdot)$: *a publicly available black-box that implements a random function.* Everyone, including an adversary, can only query $\mathcal{O}(\cdot)$ for evaluating $y = \mathcal{O}(x)$. Note that a PRF is not a RO.

The random oracle *heuristic*

> If a "natural" scheme is designed and proven secure in the random-oracle model, then we can *instantiate* the RO with a "nice" hash function and the resulting real-world scheme remains secure.

This is only a heuristic, since we do not know what "natural" and "nice" should mean. Unfortunately, there are exammples that violates this heuristic, but you may argue these examples are "unnatural". RO heuristic is still under active debate and research in the crypto community.

Matt Green has a nice exposition on the random oracle model `https://blog.cryptographyengineering.com/2011/09/29/what-is-random-oracle-model-and-why-3/`.

### *Tips about Random oracle model*

Why are people still so reliant on RO despite of a lot of controversy? Efficiency is a big advantage, and proving security is often easier (or otherwise impossible). There are several extremely useful properties of RO.

1. If $x$ has not been queried to $\mathcal{O}$, then the value of $\mathcal{O}(x)$ is uniform.

The next two are specifically relevant to security proofs by *reduction.* Suppose there is an attacker $A$ breaking a scheme constructed in the RO model, then when we use $A$ in a reduction to solve some hard problem or breaking some underlying assumption, we are responsible to answer $A$'s RO queries. This gives us room for some "magic".

2. (Extractability) If $\mathcal{A}$ queries $x$ to $\mathcal{O}$, the reduction can **see this query** and learn $x$.
3. (Programmability) The reduction can **set** the value of $\mathcal{O}(x)$ to a value of its choice, as long as this value is correctly di stributed, i.e., uniform.

Note that there is no counterpart to extractability or programmability once we instantiate the RO with any concrete function ($H$ is determined, and the evaluation of $H$ will be private to the adversary).

*Constructing a PRF in RO.* $F_k(x) := \mathcal{O}(k\|x)$. Can you prove it is a PRF?

## *PKC in RO*

In the Random Oracle Model, we can get efficient Public-key encryption and digital signature schemes with any trapdoor one-way permutations. In practice, we use efficient hash functions to instantiate these schemes.

## *RSA-OAEP*

We know plain-RSA encryption is not CPA-secure, since it is deterministic. A natural idea is to randomly pad the message before applying the RSA encryption function. An early scheme included in the standard PKCS #1 v1.5 is conjectured to be CPA-security when the randomness is sufficiently long (without proof). However, it is abandoned due to a serious chosen-ciphertext-attack by Bleichenbacher. This is replaced by OAEP: *optimal asymmetric encryption padding*, that is standardized in PKCS #1 V2.0. We describe it with an arbitrary trapdoor one-way permutation.

The basic idea is applying random padding and a two-round Feistel network to "mix" the plaintext before encrypting.[3]

[3] OAEP diagram

Our building blocks are:

- $(G, F)$: a trapdoor one-way permutation on $\{0,1\}^{n+k_0+k_1}$.
- $\mathcal{O}_1 : \{0,1\}^{k_0} \to \{0,1\}^{n+k_1}$: random oracle 1.
- $\mathcal{O}_2 : \{0,1\}^{n+k_1} \to \{0,1\}^{k_0}$: random oracle 2.

**KeyGen** $G$:

$$(pk, sk) \leftarrow G(1^n) \,.$$

**Encrypt** $E_{pk}$: on $m \in \{0,1\}^n$

- $m' := m \| \bar{0}$ denotes $m$ appended with $k_1$ bits of 0.
- Sample $r \leftarrow \{0,1\}^{k_0}$. Compute $s := \mathcal{O}_1(r) \oplus m'$, and $t := \mathcal{O}_2(s) \oplus r$.
- Output $c := f(s\|t)$.

**Decrypt** $D_{sk}(c)$:

- Compute $F_{sk}(c)$ and parse it as $s\|t$.
- Compute $r := \mathcal{O}_2(s) \oplus t$ and $m' := \mathcal{O}_1(r) \oplus s$. Output the first $n$ bits of $m'$ as $m$.

Figure 2: CPA from OAEP

**Theorem 2.** *TDP-OAEP is CPA secure for any TDP. Moreover, RSA-OAEP is* ***CCA****-secure (based on the RSA assumption).*

## *RSA-FDH*

Given a TDP $(G, F)$ and random oracle $\mathcal{O} : \{0,1\}^* \to \mathbb{Z}_n^*$.

A variant of RSA-FDH has been standardized in PKCS #1 V2.1. [4]

[4] Exercise: how does the hash make the attacks on plainRSA signature no longer applicable?

**Theorem 3.** *TDP-FDH is a secure signature.*

*Proof Idea.* Programming RO. □

| **KeyGen** $G$: | **Sign** $S_{sk}$: on $m \in \{0,1\}^*$ | **Verify** $V_{pk}(m, \sigma)$: |
|---|---|---|
| $(pk, sk) \leftarrow G(1^n)\,.$ | • Compute $\hat{m} := \mathcal{O}(m)$. | • Compute $\hat{m} = \mathcal{O}(m)$. |
| | • Output $\sigma := F_{sk}(\hat{m})$. | • Accept iff. $F_{pk}(\sigma) = \hat{m}$. |

Figure 3: TDP-FDH