# Lecture 14



**Disclaimer**. Draft note. No guarantee on completeness nor soundness. Read with caution, and shoot me an email at fsong@pdx.edu for corrections/comments (they are always welcome!)

**Logistics**. Last core lecture. Topics and review in next three lectures. Bring questions in class. Practice exam later this week. Remark RSA-OAEP in PKCS #1 V2.1.
**Last time**. Digital Signature, TDP-FDH
**Today**. DL based signature, DS from one-way functions (One-time signature and Merkle tree)

## 1 Hash-and-Sign paradigm

Full-Domain-Hash actually exemplifies a general approach of signing arbitrarily long messages based upon a digital signature scheme for fixed-length messages. This is often called the *hash-and-sign* paradigm. Another motivation is that (public-key) digital signature is not as efficient as MAC. So in practice, it is preferable to compress the messages before signing.

Hash-&-sign paradigm

$$m \longrightarrow H \longrightarrow S_{sk}(\cdot) \longrightarrow \sigma$$

**Theorem 1** (KL-Theorem 12.4)**.** *If* $\Pi$ *is a secure signature scheme for messages of length* $\ell$*, and* $H : \{0,1\}^* \to \{0,1\}^n$ *is collision resistant, then* $\Pi'$ *(hash-and-sign) is a secure signature for arb. length messages.*

Note that in practice this paradigm is susceptible to *offline* attacks that find a collision in $H$[1], which would break all signature schemes that uses $H$.

## 2 Signature based on DL

Again, TDPs based on Discrete logarithm are more complicated and instead we construct signatures directly instead of following the TDP-FDH framework. We introduce the *Digital signature algorithm* (DSA), which is included in NIST's *Digital Signature Standards* (DSS).

Correctness: $g^{H(m)\cdot s^{-1}} y^{r\cdot s^{-1}} = g^{(H(m)+xr)\cdot s^{-1}} = g^{(H(m)+xr)k(H(m)+xr)^{-1}}$.

**Theorem 2.** *Assuming DL is hard and* $H, F$ *are random oracles, then DSA is secure.*

In practice, $H$ is implemented by a cryptographic hash function, but $F$ is a specific simple function where RO may not be proper model. We have neither a security proof nor any attacks so far.

### 2.1 Optional Reading: Identification and Fiat-Shamir

Conceptually DSA can be interpreted by a general framework for designing signature schemes: the *Fiat-Shamir* heuristic [FS86][2]. Read more in [KL: Section 12.5] if interested.

[1] E.g., https://security.googleblog.com/2017/02/announcing-first-sha1-collision.html.
[2] The heuristic also has numerous applications beyond digital signatures

Let $\mathcal{G}$ be a group sampling algorithm. Let $H : \{0,1\}^* \to \mathbb{Z}_q$ and $F : G \to \mathbb{Z}_q$ be two functions. Construct $\Pi = (G, S, V)$:

- $G$: (similar to El Gamal) generate $(G, q, g) \leftarrow \mathcal{G}(1^n)$ with $q$ **prime**. Choose uniform $x \leftarrow \mathbb{Z}_q$ and compute $y = g^x$. Output $pk = (G, q, g, y)$ and $sk = (G, q, g, x)$.
- $S$: on $sk$ and $m \in \{0,1\}^*$,
  - choose uniform $k \in \mathbb{Z}_q^*$ and set $r := F(g^k)$.
  - compute $s := [k^{-1} \cdot (H(m) + xr) \bmod q]$. (If $r = 0$ or $s = 0$, start over with fresh $k$.)
  - Output $\sigma := (r, s)$.
- $V$: on input $pk$ and $(m, \sigma)$ with $\sigma = (r, s)$, accept iff.

$$r = F(g^{H(m)\cdot s^{-1}} \cdot y^{r\cdot s^{-1}})$$

Note: inverse is taken in multiplicative group $\mathbb{Z}_q^*$.

Figure 1: Abstract DSA

1. Construct an interactive *identification* protocol.
2. Collapse interaction and obtain a signature scheme in the random-oracle model.

A famous example following this approach is Schnorr's identification and signature schemes based on the DL problem [Sch89].

# 3 Signatures from hash functions

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).

This is obtained via two steps:

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

(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.

## 3.1 Lamport's one-time signature

**FS NOTE**: Draw signing diagram

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

- $G$:
  - for $i = 1, \ldots, \ell$ 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

$$pk = \begin{pmatrix} y_{1,0} & y_{2,0} & \ldots y_{\ell,0} \\ y_{1,1} & y_{2,1} & \ldots y_{\ell,1} \end{pmatrix}, \quad sk = \begin{pmatrix} x_{1,0} & x_{2,0} & \ldots x_{\ell,0} \\ x_{1,1} & x_{2,1} & \ldots x_{\ell,1} \end{pmatrix}.$$

- $S$: on message $m = m_1 \ldots m_\ell$, output $\sigma = (x_{1,m_1}, \ldots, x_{\ell,m_\ell})$.
- $V$: on input $(m = m_1 \ldots m_\ell, \sigma = (x_1, \ldots, x_\ell))$, accept iff. $H(x_i) = y_{i,m_i}$

Note: inverse is taken in multiplicative group $\mathbb{Z}_q^*$.

Figure 2: Abstract DSA

Lamport's scheme achieves a weak one-time security defined as follows. Consider the $\mathsf{Sig\text{-}forge}_{\mathcal{A},\Pi}(n)$ game, suppose we only allow the adversary asking one signing query $m$, and say that $\mathcal{A}$ succeeds if $\mathcal{A}$ produces a valid signature $\sigma^*$ on $m^* \neq m$. Call this game $\mathsf{Sig\text{-}forge}^{\mathsf{OTS}}_{\mathcal{A},\Pi}(n)$ and let $\mathsf{Sig\text{-}forge}^{\mathsf{OTS}}_{\mathcal{A},\Pi}(n) = 1$ iff. $\mathcal{A}$ succeeds.

**Definition 3** (KL-Def. 12.14)**.** $\Pi = (G, S, V)$ is $\mathsf{one\text{-}time\text{-}secure}$ if for any PPT $\mathcal{A}$,

$$\Pr[\mathsf{Sig\text{-}forge}^{\mathsf{OTS}}_{\mathcal{A},\Pi}(n) = 1] \leq \mathrm{negl}(n).$$

**Theorem 4.** *If $H$ is one-way, then $\Pi$ is a secure* $\mathsf{OTS}$.

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

## 3.2 Getting full signature using Merkle-tree

Let $\Sigma = (G, S, V)$ be a secure $\mathsf{OTS}$which can sign messages that are twice as long as its public key, i.e. $|m| = 2|pk|$. (Note that 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 $\mathsf{OWF}$.)

**FS NOTE**: Picture of a Merkle tree

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 $\varepsilon$. Each leaf (or rather path from root to leaf) corresponds to a message. For example the left-most leaf node corresponds to string $\underbrace{0\ldots0}_{h \text{ bits}}$.
- each node is associated with a OTS key-pair $(pk_p, sk_p)$ indexed by the path from the root to itself. Denote it $(pk_\varepsilon, sk_\varepsilon)$ 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 OTS signing algorithm and the leaf secret key.
  2) $\sigma_1 := (\mathsf{auth}^{(0)}, \ldots, \mathsf{auth}^{(h-1)})$, 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 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 $j = 0, \ldots, h-1$, 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-OTS and some optimized Merkle-tree constructions have gain popularity (see e.g., [BDH11] and its followup works).

# 4 TLS

Transport Layer Security (TLS, current version 1.2), replacing its precursor *secure socket layer* (SSL), is a standardized security protocol deployed on the Internet. You will learn the details in a Network Security course.

# References

[BDH11] Johannes Buchmann, Erik Dahmen, and Andreas Hülsing. Xmss-a practical forward secure signature scheme based on minimal security assumptions. In *Post-Quantum Cryptography*, pages 117–129. Springer, 2011.

[FS86] Amos Fiat and Adi Shamir. How to prove yourself: Practical solutions to identification and signature problems. In *Conference on the Theory and Application of Cryptographic Techniques*, pages 186–194. Springer, 1986.

[Sch89] Claus-Peter Schnorr. Efficient identification and signatures for smart cards. In *Conference on the Theory and Application of Cryptology*, pages 239–252. Springer, 1989.