Reminder: Homework 4

Due: Friday at the beginning of class

Cryptography
CS 555

Topic 33: Digital Signatures Part 2

Recap

- El-Gamal/RSA-OAEP
- Digital Signatures
  - Similarities and differences with MACs
  - Security
  - Hash then MAC
  - One-Time-Signatures

Digital Signature Scheme

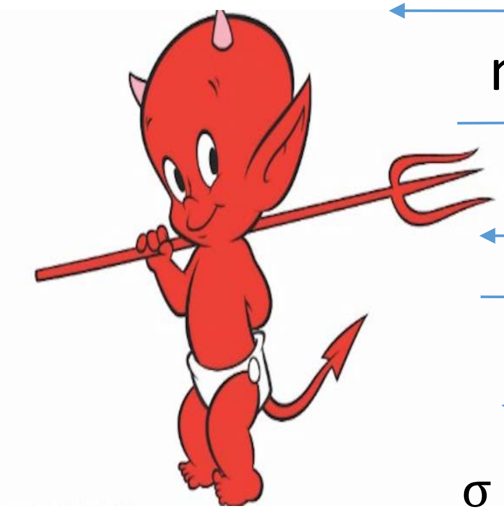
- Three Algorithms
  - $\mathrm{Gen}(1^n, R)$ (Key-generation algorithm)
    - Input: Random Bits R
    - Output: $(\boldsymbol{pk}, \boldsymbol{sk}) \in \mathcal{K}$
  - $\sigma \leftarrow \mathrm{Sign}_{\mathrm{sk}}(m, R)$ (Signing algorithm)
    - Input: Secret key sk message m, random bits R
    - Output: signature σ
  - $\mathrm{b} \coloneqq \mathrm{Vrfy}_{\mathrm{pk}}(m, \sigma)$ (Verification algorithm --- Deterministic)
    - Input: Public key pk, message m and a signature σ
    - Output: 1 (Valid) or 0 (Invalid)

Alice must run key generation algorithm in advance an publishes the public key: pk

Assumption: Adversary only gets to see pk (not sk)

- **Correctness**: $\mathrm{Vrfy}_{\mathrm{pk}}(\mathrm{m}, \mathrm{Sign}_{\mathrm{sk}}(m, R))=1$ (except with negligible probability)

Signature Experiment ($\mathrm{Sig-forge}_{A,\Pi}(n)$)



Public Key: pk

m_1

$\sigma_1 = \mathbf{Sign}_{sk}(\boldsymbol{m_1})$

m_2

$\sigma_2 = \mathbf{Sign}_{sk}(\boldsymbol{m_2})$

$\vdots$

$\sigma, m \notin \mathfrak{Q} = \{m_1, m_2 \dots\}$

$\mathrm{Sig-forge}_{A,\Pi}(n) = \mathrm{Vrfy}_{pk}(m, \sigma)$

Random bit b

(pk,sk) = Gen(.)

$$\forall PPT\ A\ \exists \mu\ (\mathrm{negligible})\ \mathrm{s.t}$$
$$\Pr\left[\mathrm{Sig-forge}_{A,\Pi}(n) = 1\right] \le \mu(n)$$

Plain RSA Signatures

- Plain RSA
- Public Key (pk): N = pq, e such that $\mathrm{GCD}\big(\mathrm{e}, \phi(N)\big) = 1$
  - $\phi(N) = (p-1)(q-1)$ for distinct primes p and q
- Secret Key (sk): N, d such that ed=1 mod $\phi(N)$

$$\mathbf{Sign}_{sk}(\boldsymbol{m}) = \boldsymbol{m^d \ mod\ N}$$

$$\mathbf{Vrfy}_{pk}(\boldsymbol{m}, \sigma) = \begin{cases} \mathbf{1} & \boldsymbol{if\ m} = [\sigma^{\boldsymbol{e}}\ \boldsymbol{mod\ N}] \\ \mathbf{0} & \boldsymbol{otherwise} \end{cases}$$

- Verification Works because

$$\left[\mathbf{Sign}_{sk}(\boldsymbol{m})^e \text{ mod N}\right] = [m^{ed} \text{mod N}] = \left[m^{[ed\ mod\ \boldsymbol{\phi}(\boldsymbol{N})]} \text{mod N}\right] = m$$

No Message Attack

- **Goal:** Generate a forgery using only the public key
  - No intercepted signatures required
- Public Key (pk): $\mathrm{N} = \mathrm{pq}, \mathrm{e}$ such that $\mathrm{GCD}\big(\mathrm{e}, \phi(N)\big) = 1$
  - $\phi(N) = (p-1)(q-1)$ for distinct primes p and q
- Pick random $\sigma \in \mathbb{Z}^*_{\mathrm{N}}$
- Set $\boldsymbol{m} = [\sigma^{\boldsymbol{e}}\ \boldsymbol{mod}\ \boldsymbol{N}]$.
- Output (m, σ)

$$\mathbf{Vrfy}_{\boldsymbol{pk}}(\boldsymbol{m}, \sigma) = \begin{cases} \mathbf{1} & \boldsymbol{if}\ \boldsymbol{m} = [\sigma^{\boldsymbol{e}}\ \boldsymbol{mod}\ \boldsymbol{N}] \\ \mathbf{0} & \boldsymbol{otherwise} \end{cases}$$

Forging a Signature on Arbitrary Message

- (Last Attack): Attacker does not control message m in forgery
- What if we can convince honest party to sign random messages?
  - Authentication by signing random nonces

- Attacker selects message $m \in \mathbb{Z}_{\mathrm{N}}^*$

- Attacker selects $r_1 \in \mathbb{Z}_{\mathrm{N}}^*$ at random and sets $r_2 = m(r_1)^{-1}$

- Attacker requests signatures σ_1 and σ_2 for r_1 and r_2 (respectively)

Forging a Signature on Arbitrary Message

- Attacker selects message $m \in \mathbb{Z}_N^*$
- Attacker selects $r_1 \in \mathbb{Z}_N^*$ at random and sets $r_2 = m(r_1)^{-1}$
- Attacker requests signatures σ_1 and σ_2 for r_1 and r_2 (respectively)

- Attacker outputs signature $\sigma = [\sigma_1\ \sigma_2 \bmod N]$ for m

$$
\begin{aligned}
\sigma^e &= [(\sigma_1)^e(\sigma_2)^e \bmod N] \\
&= [r_1\ r_2 \bmod N] \\
&= \left[r_1 m(r_1)^{-1} \bmod N\right] \\
&= m
\end{aligned}
$$

RSA-FDH (Full Domain Hash)

- Public Key (pk): $\mathrm{N} = \mathrm{pq}$, e and **hash function** $\boldsymbol{H}: \{\mathbf{0}, \mathbf{1}\}^* \rightarrow \mathbb{Z}_{\mathbf{N}}^*$
- Secret Key (sk): N, d such that ed=1 mod $\phi(N)$

$$\mathbf{Sign}_{\boldsymbol{sk}}(\boldsymbol{m}) = \boldsymbol{H}(\boldsymbol{m})^{\boldsymbol{d}}\ \boldsymbol{mod}\ \boldsymbol{N}$$

$$\mathbf{Vrfy}_{\boldsymbol{pk}}(\boldsymbol{m}, \sigma) = \begin{cases} \mathbf{1} & \boldsymbol{if}\, \boldsymbol{H}(\boldsymbol{m}) = [\sigma^{\boldsymbol{e}}\ \boldsymbol{mod}\ \boldsymbol{N}] \\ \mathbf{0} & \boldsymbol{otherwise} \end{cases}$$

- Verification Works because

$$\left[\mathbf{Sign}_{\boldsymbol{sk}}(\boldsymbol{m})^{e} \bmod \mathrm{N}\right] = [\boldsymbol{H}(\boldsymbol{m})^{ed} \mathrm{mod}\ \mathrm{N}]$$
$$= \left[\boldsymbol{H}(\boldsymbol{m})^{[ed\ mod\ \boldsymbol{\phi}(\boldsymbol{N})]} \mathrm{mod}\ \mathrm{N}\right] = [\boldsymbol{H}(\boldsymbol{m}) \mathrm{mod}\ \mathrm{N}]$$

RSA-FDH (Full Domain Hash)

- What properties does H are required for security of RSA-FDH?
- Collision Resistance is necessary
- If attacker finds m and m' such that $H(m) = H(m')$ then he can win Sig-Forge game.
- How?

RSA-FDH (Full Domain Hash)

- What properties does H are required for security of RSA-FDH?

- Collision Resistance is necessary
- Should be infeasible to find m, σ such that

$$\boldsymbol{H}(\boldsymbol{m}) = \sigma^{\boldsymbol{e}}\ \boldsymbol{mod}\ \boldsymbol{N}$$

- Why?
  - No-message attack
  - σ is a valid signature for m

RSA-FDH (Full Domain Hash)

- What properties does H are required for security of RSA-FDH?

- Collision Resistance is necessary
- Should be infeasible to find m, σ such that $\boldsymbol{H(m) = \sigma^e \; mod \; N}$
- Should be infeasible to find m, m_1, m_2 such that
$$\boldsymbol{H(m) = H(m_1)\, H(m_2)\; mod\; N}$$
- Why?
  - $\sigma = [\sigma_1\ \sigma_2 \text{ mod N}]$ is a valid signature for m

RSA-FDH (Full Domain Hash)

- What properties does H are required for security of RSA-FDH?

- Collision Resistance is necessary
- Should be infeasible to find m, σ such that $\boldsymbol{H(m) = \sigma^e \; mod \; N}$
- Should be infeasible to find m, m_1, m_2 such that

$$\boldsymbol{H(m) = H(m_1)\, H(m_2) \; mod \; N}$$

- Random Oracle H satisfies all three properties

RSA-FDH (Full Domain Hash)

$$\mathbf{Sign}_{sk}(\boldsymbol{m}) = \boldsymbol{H}(\boldsymbol{m})^{\boldsymbol{d}}\ \boldsymbol{mod}\ \boldsymbol{N}$$

$$\mathbf{Vrfy}_{pk}(\boldsymbol{m}, \sigma) = \begin{cases} \mathbf{1} & \boldsymbol{if}\, \boldsymbol{H}(\boldsymbol{m}) = [\sigma^{\boldsymbol{e}}\ \boldsymbol{mod}\ \boldsymbol{N}] \\ \mathbf{0} & \boldsymbol{otherwise} \end{cases}$$

Theorem 12.7: If the RSA problem is hard relative to GenRSA and if H is modeled as a random oracle then RSA-FDH is secure.

Proof Sketch: Use Sig-Forge attacker A to build RSA-INV attacker A'

RSA-FDH (Full Domain Hash)

$$\mathbf{Sign}_{sk}(\boldsymbol{m}) = \boldsymbol{H}(\boldsymbol{m})^{\boldsymbol{d}}\ \boldsymbol{mod}\ \boldsymbol{N}$$

$$\mathbf{Vrfy}_{pk}(\boldsymbol{m}, \sigma) = \begin{cases} \mathbf{1} & \boldsymbol{if} \boldsymbol{H}(\boldsymbol{m}) = [\sigma^{\boldsymbol{e}}\ \boldsymbol{mod}\ \boldsymbol{N}] \\ \mathbf{0} & \boldsymbol{otherwise} \end{cases}$$

Theorem 12.7: If the RSA problem is hard relative to GenRSA and if H is modeled as a random oracle then RSA-FDH is secure.

Proof Sketch:

Observation 1: If the attacker A outputs (m, σ) and never queries H(m) then the odds of A winning are negligible.

Observation 2: We can guess that attacker A will output attempted forgery for message m_i, where m_i is the i'th query to random oracle H(.)

RSA-FDH (Full Domain Hash)

$$\mathbf{Sign}_{sk}(\boldsymbol{m}) = \boldsymbol{H}(\boldsymbol{m})^{\boldsymbol{d}}\ \boldsymbol{mod}\ \boldsymbol{N}$$

$$\mathbf{Vrfy}_{pk}(\boldsymbol{m}, \sigma) = \begin{cases} \mathbf{1} & \boldsymbol{if}\, \boldsymbol{H}(\boldsymbol{m}) = [\sigma^{\boldsymbol{e}}\ \boldsymbol{mod}\ \boldsymbol{N}] \\ \mathbf{0} & \boldsymbol{otherwise} \end{cases}$$

Theorem 12.7: If the RSA problem is hard relative to GenRSA and if H is modeled as a random oracle then RSA-FDH is secure.

Proof Sketch: Suppose that we guess that attacker A will output attempted forgery for message m_i, where m_i is the i'th query to random oracle H(.).

- We are right with probability 1/q(n).
- Abort if the attacker A ever requests a signature for m_i (i.e., guess is wrong)

RSA-FDH (Full Domain Hash)

$$\mathbf{Sign}_{sk}(\boldsymbol{m}) = \boldsymbol{H}(\boldsymbol{m})^{\boldsymbol{d}}\ \boldsymbol{mod}\ \boldsymbol{N}$$

$$\mathbf{Vrfy}_{pk}(\boldsymbol{m}, \sigma) = \begin{cases} \mathbf{1} & \boldsymbol{if}\, \boldsymbol{H}(\boldsymbol{m}) = [\sigma^{\boldsymbol{e}}\ \boldsymbol{mod}\ \boldsymbol{N}] \\ \mathbf{0} & \boldsymbol{otherwise} \end{cases}$$

Theorem 12.7: If the RSA problem is hard relative to GenRSA and if H is modeled as a random oracle then RSA-FDH is secure.

Proof Sketch: will simulate A

- **RSA-Inv attacker B** starts with (N,e,y).
- **Goal of B:** Decrypt y using the signature forging adversary.
- **Programmability of Random Oracle:** When signature attacker makes its i^{th} random oracle query $H(m_i)$ respond with y instead of $H(m_i)$
  - Signature attacker cannot tell the difference since y is random!

RSA-FDH (Full Domain Hash)

$$\mathbf{Sign}_{sk}(\boldsymbol{m}) = \boldsymbol{H}(\boldsymbol{m})^{\boldsymbol{d}}\ \boldsymbol{mod}\ \boldsymbol{N}$$

$$\mathbf{Vrfy}_{pk}(\boldsymbol{m}, \sigma) = \begin{cases} \mathbf{1} & \boldsymbol{if}\,\boldsymbol{H}(\boldsymbol{m}) = [\sigma^{\boldsymbol{e}}\ \boldsymbol{mod}\ \boldsymbol{N}] \\ \mathbf{0} & \boldsymbol{otherwise} \end{cases}$$

Theorem 12.7: If the RSA problem is hard relative to GenRSA and if H is modeled as a random oracle then RSA-FDH is secure.

Proof Sketch: Start with (N,e,y) our goal is to decrypt y using the signature forging adversary.

- **Programmability of Random Oracle:** When signature attacker makes its ith random oracle query $H(m_i)$ respond with y instead of $H(m_i)$
  - Signature attacker cannot tell the difference!
- **Forgery:** A valid forgery for message m_i is now $\boldsymbol{y}^{\boldsymbol{d}}\ \boldsymbol{mod}\ \boldsymbol{N}$ (the decryption of y)

$$\Pr[\text{RSA–INV}_B(n) = 1] = \frac{1}{q(n)} \Pr[\text{Sig–Forge}_A(n) = 1] - \text{negl}(n)$$

RSA-FDH (Full Domain Hash)

$$\mathbf{Sign}_{sk}(\boldsymbol{m}) = \boldsymbol{H}(\boldsymbol{m})^{\boldsymbol{d}}\ \boldsymbol{mod}\ \boldsymbol{N}$$

$$\mathbf{Vrfy}_{pk}(\boldsymbol{m}, \sigma) = \begin{cases} \mathbf{1} & \boldsymbol{if} \boldsymbol{H}(\boldsymbol{m}) = [\sigma^{\boldsymbol{e}}\ \boldsymbol{mod}\ \boldsymbol{N}] \\ \mathbf{0} & \boldsymbol{otherwise} \end{cases}$$

Remark: In practice output of H needs to be close to all of $\mathbb{Z}_{\mathbf{N}}^{*}$ (otherwise known attacks exist)

H = SHA-1 doesn't work for two reasons

1. The output is too short
2. SHA-1 is no longer collision resistant ☺

Identification Scheme

- Interactive protocol that allows one party to prove its identify (authenticate itself) to another
- Two Parties: Prover and Verifier
  - Prover has secret key sk and Verifier has public key pk

1. Prover runs P_1(sk) to obtain (I,st) ---- initial message I, state st
   - Sends I to Verifier
2. Verifier picks random message r from distribution Ω_{pk} and sends r to Prover
3. Prover runs P_2(sk,st,r) to obtain s and sends s to verifier
4. Verifier checks if V(pk,r,s)=I

Identification Scheme

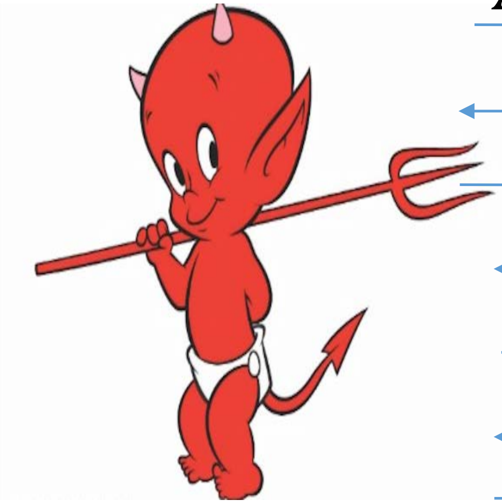
1. Prover runs $P_1(sk)$ to obtain (I,st)   ---- initial message I, state st
   1. Sends I to Verifier
2. Verifier picks random message r from distribution Ω_{pk} and sends r to Prover
3. Prover runs $P_2(sk,st,r)$ to obtain s and sends s to verifier
4. Verifier checks if V(pk,r,s)=I

An eavesdropping attacker obtains a transcript (I,r,s) of all the message sent.

Transcript Oracle: $Trans_{sk}(.)$ runs honest execution and outputs transcript.

Identification Game ($\mathrm{Ident}_{\mathrm{A},\Pi}(\mathrm{n})$)

Public Key: pk



$Again$

$\mathbf{Trans}_{sk}(.)$

$Again$

⋮

$\mathbf{Trans}_{sk}(.)$

I

$r \leftarrow \Omega_{pk}$

$Again$

$\mathbf{Trans}_{sk}(.)$

s

$$Ident_{\mathrm{A},\Pi}(\mathrm{n}) = \begin{cases} 1 \text{ if } V(pk, r, s) = I \\ 0 \text{ otherwise} \end{cases}$$

(pk,sk) = Gen(.)

$$\forall PPT\ A\ \exists \mu \text{ (negligible) s.t } \Pr[\mathrm{Ident}_{\mathrm{A},\Pi}(\mathrm{n}) = 1] \leq \mu(n)$$

Fiat-Shamir Transform

- Identification Schemes can be transformed into signatures
- $\text{Sign}_{sk}(m)$
  - First compute (I,st)= P_1(sk)   (as prover)
  - Next compute the challenge $\boldsymbol{r} = \boldsymbol{H}(\boldsymbol{I}, \boldsymbol{m})$ (as verifier)
  - Compute the response s = P_2(sk,st,r)
  - Output signature (r,s)
- $\text{Vrfy}_{pk}(m,(r,s))$
  - Compute I := V(pk,r,s)
  - Check that H(I,m)=r

Theorem 12.10: If the identification scheme is secure and H is a random oracle then the above signature scheme is secure.

Schnorr Identification Scheme

- Verifier knows $h=g^x$
- Prover knows x such that $h=g^x$

1. Prover runs $P_1(x)$ to obtain $\left(k \in \mathbb{Z}_q, I = g^k\right)$ and sends initial message I to verifier
2. Verifier picks random $r \in \mathbb{Z}_q$ (q is order of the group) and sends r to prover
3. Prover runs $P_2(x,k,r)$ to obtain $s := [rx + k \ mod \ q]$ and sends s to Verifier
4. Verifier checks if $g^s * (h^{-1})^r = I = g^k$

Schnorr Identification Scheme

- Verifier knows $h=g^x$
- Prover knows x such that $h=g^x$

1. Prover runs $P_1(x)$ to obtain $\left(k \in \mathbb{Z}_q, I = g^k\right)$ and sends initial message I to verifier
2. Verifier picks random $r \in \mathbb{Z}_q$ (q is order of the group) and sends r to prover
3. Prover runs $P_2(x,k,r)$ to obtain $s \coloneqq [rx + k \ mod \ q]$ and sends s to Verifier
4. Verifier checks if $g^s * (h^{-1})^r = I = g^k$

$$g^s * (h^{-1})^r = g^{rx+k \ mod \ q} * g^{-xr} = g^k$$

Schnorr Identification Scheme

- Verifier knows $h=g^x$
- Prover knows x such that $h=g^x$
- Prover runs $P_1(x)$ to obtain $\left(k \in \mathbb{Z}_q, I = g^k\right)$and sends initial message I to verifier
- Verifier picks random $r \in \mathbb{Z}_q$ (q is order of the group) and sends r to prover
- Prover runs P1(x,k,r) to obtain $s \coloneqq [rx + k \; mod \; q]$ and sends s to Verifier
- Verifier checks if $g^s * (h^{-1})^r = I = g^k$

Theorem 12.11: If the discrete-logarithm problem is hard (relative to group generator) then Schnorr identification scheme is secure.

Digital Signature Algorithm (DSA)

- Secret key is x, public key is $h=g^x$
- $\text{Sign}_{sk}(m)$
  - Pick random $(k \in \mathbb{Z}_q)$ and set $r = F(g^k) = [g^k \; mod \; q]$
  - Compute $s \coloneqq [k^{-1}(xr + H(m)) \; mod \; q]$
  - Output signature (r,s)
- $\text{Vrfy}_{pk}(m,(r,s))$ check to make sure that

$$r = F\left(g^{H(m)s^{-1}} h^{rs^{-1}}\right)$$

Theorem: If H and F are modeled as random oracles then DSA is secure.

Weird Assumption?

- **Theory:** DSA Still lack compelling proof of security from standard crypto assumptions
- **Practice:** DSA has been used/studied for decades without attacks

Digital Signature Algorithm (DSA)

- Secret key is x, public key is $h=g^x$
- $\text{Sign}_{sk}(m)$
  - Pick random $\left(k \in \mathbb{Z}_q\right)$ and set $r = F\left(g^k\right) = [g^k \bmod q]$
  - Compute $s := [k^{-1}(xr + H(m)) \bmod q]$
  - Output signature (r,s)
- $\text{Vrfy}_{pk}(m,(r,s))$ check to make sure that

$$r = F\left(g^{H(m)s^{-1}} h^{rs^{-1}}\right)$$

Remark: If signer signs two messages with same random $k \in \mathbb{Z}_q$ then attacker can find secret key sk!

- **Theory:** Shouldn't happen
- **Practice:** Will happen if a weak PRG is used
- Sony PlayStation (PS3) hack in 2010.

Next Class: Digital Signatures Part 2

- Read Katz and Lindell: 12.8