Course Business

- **Homework 4 Due Thursday in Class**
- Bonus Problem (10 Points)
  - Second bonus problem (5 pts) is easiest to solve with Mathematica
  - **https://sandbox.open.wolframcloud.com**

Cryptography
CS 555

Week 13:

- El Gamal
- RSA Attacks and Fixes
- Digital Signatures

Readings: Katz and Lindell Chapter 10 & Chapter 11.1-11.2, 11.4

Week 13 Topic 1: El-Gamal Encryption

El-Gamal Encryption

- **Public Key:** g, h
- **Secret Key:** $x = \mathrm{dlog}_g(h)$
- $\mathrm{Enc}_{\mathrm{pk}}(m) = \langle g^y, m \cdot h^y \rangle$ for a random y $\in \mathbb{Z}_q$
- $\mathrm{Dec}_{\mathrm{sk}}(c = (c_1, c_2)) = c_2 c_1^{-x}$

$$\begin{aligned}\mathrm{Dec}_{\mathrm{sk}}(g^y, m \cdot h^y) &= m \cdot h^y (g^y)^{-x} \\ &= m \cdot h^y (g^y)^{-x} \\ &= m \cdot (g^x)^y (g^y)^{-x} \\ &= m \cdot g^{xy} g^{-xy} \\ &= m\end{aligned}$$

El-Gamal Encryption

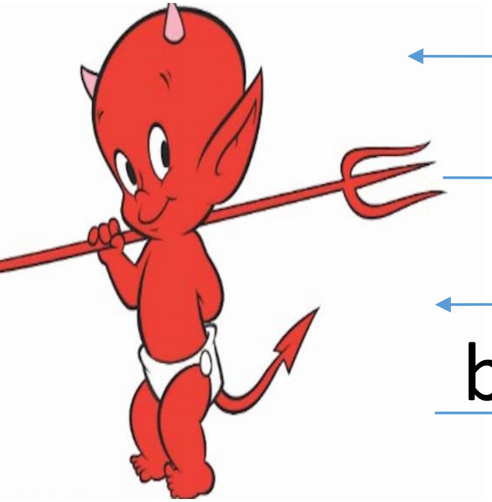
- $\mathrm{Enc}_{\mathrm{pk}}(m) = \langle g^y, m \cdot h^y \rangle$ for a random y $\in \mathbb{Z}_q$
- $\mathrm{Dec}_{\mathrm{sk}}(c = (c_1, c_2)) = c_2 c_1^{-x}$

Theorem 11.18: Let $\Pi = (Gen, Enc, Dec)$ be the El-Gamal Encryption scheme (above) then if DDH is hard relative to $\mathcal{G}$ then Π is CPA-Secure.

Proof: Recall that CPA-security and eavesdropping security are equivalent for public key crypto. Therefore, it suffices to show that for all PPT A there is a negligible function **negl** such that

$$\Pr\left[\mathrm{PubK}_{A,\Pi}^{\mathrm{eav}}(n) = 1\right] \leq \frac{1}{2} + \mathbf{negl(n)}$$

Eavesdropping Security ($\mathrm{PubK}_{A,\Pi}^{\mathrm{eav}}(n)$)



Public Key: pk

$m_0\ , m_1$

$\boldsymbol{c_1 = \mathrm{Enc}_{pk}(m_b\ )}$

b’

Random bit b

(pk,sk) = Gen(.)

$$\forall PPT\ A\ \exists \mu\ (\mathrm{negligible})\ \mathrm{s.t}$$

$$\Pr\left[\mathrm{PubK}_{A,\Pi}^{\mathrm{eav}}(\mathrm{n}) = 1\right] \leq \frac{1}{2} + \mu(n)$$

El-Gamal Encryption

Theorem 11.18: Let $\Pi = (Gen, Enc, Dec)$ be the El-Gamal Encryption scheme (above) then if DDH is hard relative to $\mathcal{G}$ then Π is CPA-Secure.

Proof: First introduce an `encryption scheme' $\widetilde{\Pi}$ in which $\widetilde{\mathrm{Enc}_{\mathrm{pk}}}(m) = \langle g^y, m \cdot g^z \rangle$ for random $\mathrm{y}, \mathrm{z} \in \mathbb{Z}_q$ (there is actually no way to do decryption, but the experiment $\mathrm{PubK}^{\mathrm{eav}}_{\mathrm{A},\widetilde{\Pi}}(\mathrm{n})$ is still well defined).

Claim: $\Pr\left[\mathrm{PubK}^{\mathrm{eav}}_{\mathrm{A},\widetilde{\Pi}}(\mathrm{n}) = 1\right] = \frac{1}{2}$

El-Gamal Encryption

Claim: $\Pr\left[\mathrm{PubK}^{\mathrm{eav}}_{\mathrm{A},\widetilde{\Pi}}(\mathrm{n}) = 1\right] = \frac{1}{2}$

Proof: (using Lemma 11.15)

$$\Pr\left[\mathrm{PubK}^{\mathrm{eav}}_{\mathrm{A},\widetilde{\Pi}}(\mathrm{n}) = 1\right]$$

$$= \frac{1}{2}\Pr\left[\mathrm{PubK}^{\mathrm{eav}}_{\mathrm{A},\widetilde{\Pi}}(\mathrm{n}) = 1 | b = 1\right] + \frac{1}{2}\left(1 - \Pr\left[\mathrm{PubK}^{\mathrm{eav}}_{\mathrm{A},\widetilde{\Pi}}(\mathrm{n}) = 0 | b = 0\right]\right)$$

$$= \frac{1}{2} + \frac{1}{2}\left(\Pr_{\mathrm{y,z}\leftarrow\mathbb{Z}_q}[A(\langle g^y, m_1 \cdot g^z\rangle) = 1] - \Pr_{\mathrm{y,z}\leftarrow\mathbb{Z}_q}[A(\langle g^y, m_0 \cdot g^z\rangle) = 1]\right)$$

$$= \frac{1}{2}$$

El-Gamal Encryption

Theorem 11.18: Let $\Pi = (Gen, Enc, Dec)$ be the El-Gamal Encryption scheme (above) then if DDH is hard relative to $\mathcal{G}$ then Π is CPA-Secure.

Proof: We just showed that

$$\Pr\left[\mathrm{PubK}^{\mathrm{eav}}_{\mathrm{A},\widetilde{\Pi}}(\mathrm{n}) = 1\right] = \frac{1}{2}$$

Therefore, it suffices to show that

$$\left|\Pr\left[\mathrm{PubK}^{\mathrm{eav}}_{\mathrm{A},\Pi}(\mathrm{n}) = 1\right] - \Pr\left[\mathrm{PubK}^{\mathrm{eav}}_{\mathrm{A},\widetilde{\Pi}}(\mathrm{n}) = 1\right]\right| \leq \mathbf{negl}(n)$$

This, will follow from DDH assumption.

El-Gamal Encryption

Theorem 11.18: Let $\Pi = (Gen, Enc, Dec)$ be the El-Gamal Encryption scheme (above) then if DDH is hard relative to $\mathcal{G}$ then Π is CPA-Secure.

Proof: We can build $B(g^x, g^y, Z)$ to break DDH assumption if Π is not CPA-Secure. Simulate eavesdropping attacker A

1. Send attacker public key $\mathrm{pk} = \langle \mathbb{G}, q, g, h = g^x \rangle$
2. Receive m_0,m_1 from A.
3. Send A the ciphertext $\langle g^y, m_b \cdot Z \rangle$.
4. Output 1 if and only if attacker outputs b'=b; otherwise output 0.

$$\left|\Pr\left[B(g^x, g^y, Z) = 1 \middle| Z = g^{xy}\right] - \Pr\left[B(g^x, g^y, Z) = 1 \middle| Z = g^z\right]\right|$$
$$= \left|\Pr\left[\mathrm{PubK}_{\mathrm{A},\Pi}^{\mathrm{eav}}(\mathrm{n}) = 1\right] - \Pr\left[\mathrm{PubK}_{\mathrm{A},\tilde{\Pi}}^{\mathrm{eav}}(\mathrm{n}) = 1\right]\right|$$
$$= \left|\Pr\left[\mathrm{PubK}_{\mathrm{A},\Pi}^{\mathrm{eav}}(\mathrm{n}) = 1\right] - {}^{1}\!/_{2}\right|$$

El-Gamal Encryption

- $\mathrm{Enc}_{\mathrm{pk}}(m) = \langle g^y, m \cdot h^y \rangle$ for a random y ∈ $\mathbb{Z}_q$ and $h = g^x$,
- $\mathrm{Dec}_{\mathrm{sk}}(c = (c_1, c_2)) = c_2 c_1^{-x}$

Fact: El-Gamal Encryption is malleable.

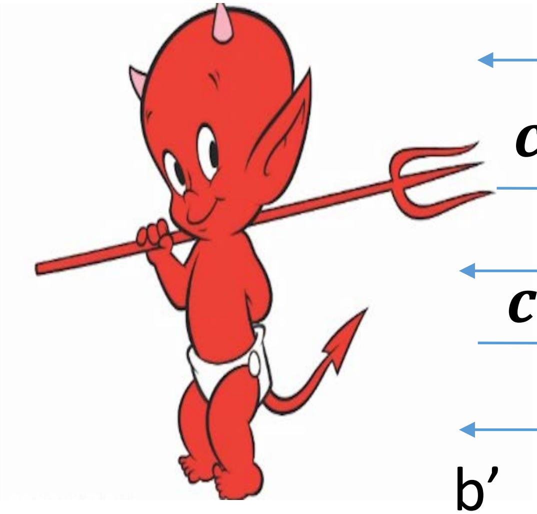
$$\mathrm{c} = \mathrm{Enc}_{\mathrm{pk}}(m) = \langle g^y, m \cdot h^y \rangle$$
$$c' = \langle g^y, 2 \cdot m \cdot h^y \rangle$$
$$\mathrm{Dec}_{\mathrm{sk}}(c') = 2 \cdot m \cdot h^y \cdot g^{-xy} = 2m$$

Hint: This observation may be relevant for homework 4.

Key Encapsulation Mechanism (KEM)

- Three Algorithms
  - $\mathrm{Gen}(1^n, R)$ (Key-generation algorithm)
    - Input: Random Bits R
    - Output: $(\boldsymbol{pk}, \boldsymbol{sk}) \in \boldsymbol{\mathcal{K}}$
  - $\mathrm{Encaps}_{\mathrm{pk}}(1^n, R)$
    - Input: security parameter, random bits R
    - Output: Symmetric key $\mathrm{k} \in \{0,1\}^{\ell(n)}$ and a ciphertext c
  - $\mathrm{Decaps}_{\mathrm{sk}}(c)$ (Deterministic algorithm)
    - Input: Secret key $\mathrm{sk} \in \mathcal{K}$ and a ciphertex c
    - Output: a symmetric key$\{0,1\}^{\ell(n)}$ or $\perp$ (fail)
- **Invariant**: $\mathrm{Decaps}_{\mathrm{sk}}$(c)=k whenever (c,k) = $\mathrm{Encaps}_{\mathrm{pk}}(1^n, R)$

KEM CCA-Security ($\mathrm{KEM}^{\mathrm{cca}}_{\mathrm{A},\Pi}(\mathrm{n})$)



$(\boldsymbol{pk}, \boldsymbol{c}, \boldsymbol{kb})$

⋮

$\boldsymbol{c_1} \neq \boldsymbol{c}$

$\mathbf{Decaps}_{\boldsymbol{sk}}(\boldsymbol{c_1})$

$\boldsymbol{c_2} \neq \boldsymbol{c}$

$\mathbf{Decaps}_{\boldsymbol{sk}}(\boldsymbol{c_2})$

b'

⋮

Random bit b

(pk,sk) = Gen(.)

$(\boldsymbol{c}, \boldsymbol{k_0}) = \mathbf{Encaps}_{\boldsymbol{pk}}(.)$

$\boldsymbol{k_1} \leftarrow \{\mathbf{0}, \mathbf{1}\}^{\boldsymbol{n}}$

$$\forall PPT\ A\ \exists \mu\ (\text{negligible})\ \text{s.t}$$

$$\Pr\left[\mathrm{KEM}^{\mathrm{cca}}_{\mathrm{A},\Pi} = 1\right] \leq \frac{1}{2} + \mu(n)$$

KEM from RSA and El-Gamal

- Recap: CCA-Secure KEM from RSA in Random Oracle Model

- El-Gamal also yields CCA-Secure KEM in Random Oracle Model
  - $\left(\boldsymbol{g^y}, \boldsymbol{H}(\boldsymbol{h^y})\right) \leftarrow \mathbf{Encaps_{pk}}(\mathbf{1}^{\boldsymbol{n}}; \boldsymbol{R})$ and $\mathbf{Decaps_{sk}}(\boldsymbol{g^y}) = \boldsymbol{H}(\boldsymbol{g^{xy}})$
  - **CDH** assumption must hold.
- Above construction is also a CPA-Secure KEM in standard model
  - As long as $Pr_{x \in \mathbb{G}}[H(x) = k] \approx 2^{-\ell}$ for each key $k \in \{0,1\}^{\ell}$ and ***DDH*** holds
  - **Disadvantage:** weaker security notion for KEM, stronger DDH assumption
  - **Advantage:** Proof in standard model

CCA-Secure Variant in Random Oracle Model

- Key Generation ($\mathrm{Gen}(1^n)$):
  1. Run $\mathcal{G}(1^n)$ to obtain a cyclic group $\mathbb{G}$ of order q (with $\|q\| = n$) and a generator g such that $< \mathrm{g} >= \mathbb{G}$.
  2. Choose a random $\mathrm{x} \in \mathbb{Z}_q$ and set $h = g^x$
  3. Public Key: $\mathrm{pk} = \langle \mathbb{G}, q, g, h \rangle$
  4. Private Key: $\mathrm{sk} = \langle \mathbb{G}, q, g, x \rangle$
- $\mathrm{Enc}_{\mathrm{pk}}(m) = \left\langle g^y, c', Mac_{K_M}(c') \right\rangle$ for a random $\mathrm{y} \in \mathbb{Z}_q$ where

$$K_E \| K_M = H(h^y) \quad \text{(KEM)}$$

and

$$c' = \mathrm{Enc}'_{\mathrm{K_E}}(m) \quad \text{(Encrypt then MAC)}$$

CCA-Secure Variant in Random Oracle Model

Public Key: $\mathrm{pk} = \langle \mathbb{G}, q, g, h \rangle$
Private Key: $\mathrm{sk} = \langle \mathbb{G}, q, g, x \rangle$

- $\mathrm{Enc}_{\mathrm{pk}}(m) = \left\langle g^y, c', Mac_{K_M}(c') \right\rangle$ for a random y $\in \mathbb{Z}_q$ and $K_E \| K_M = H(h^y)$ and $c' = \mathrm{Enc}'_{\mathrm{K_E}}(m)$
- $\mathrm{Dec}_{\mathrm{sk}}(\langle c, c', t \rangle)$

1. $K_E \| K_M = H(c^x)$
2. If $\mathrm{Vrfy}_{\mathrm{K_M}}(c', t) \neq 1$ or $c \notin \mathbb{G}$ output $\perp$; otherwise output $\mathrm{Dec}'_{\mathrm{K_E}}(c')$

CCA-Secure Variant in Random Oracle Model

Theorem: If $\mathrm{Enc}'_{\mathrm{K_E}}$ is CPA-secure, $\mathrm{Mac}_{\mathrm{K_M}}$ is a strong MAC and a problem called gap-CDH is hard then this a CCA-secure public key encryption scheme in the random oracle model.

- $\mathrm{Enc}_{\mathrm{pk}}(m) = \langle g^y, c', \mathrm{Mac}_{\mathrm{K_M}}(c') \rangle$ for a random y $\in \mathbb{Z}_q$ and $K_E \| K_M = H(h^y)$ and $c' = \mathrm{Enc}'_{\mathrm{K_E}}(m)$
- $\mathrm{Dec}_{\mathrm{sk}}(\langle c, c', t \rangle)$

1. $K_E \| K_M = H(c^x)$
2. If $\mathrm{Vrfy}_{\mathrm{K_M}}(c', t) \neq 1$ or $c \notin \mathbb{G}$ output $\perp$; otherwise output $\mathrm{Dec}'_{\mathrm{K_E}}(c')$

CCA-Secure Variant in Random Oracle Model

Remark: The CCA-Secure variant is used in practice in the ISO/IEC 18033-2 standard for public-key encryption.

- Diffie-Hellman Integrated Encryption Scheme (DHIES)
- Elliptic Curve Integrated Encryption Scheme (ECIES)
- $\mathrm{Enc}_{\mathrm{pk}}(m) = \langle g^y, c', \mathrm{Mac}_{\mathrm{K_M}}(c') \rangle$ for a random y ∈ $\mathbb{Z}_q$ and $K_E \| K_M = H(h^y)$ and $c' = \mathrm{Enc}'_{\mathrm{K_E}}(m)$
- $\mathrm{Dec}_{\mathrm{sk}}(\langle c, c', t \rangle)$

1. $K_E \| K_M = H(c^x)$
2. If $\mathrm{Vrfy}_{\mathrm{K_M}}(c', t) \neq 1$ or $c \notin \mathbb{G}$ output $\perp$; otherwise output $\mathrm{Dec}'_{\mathrm{K_E}}(c')$

Week 13: Topic 2: More RSA Attacks + Fixes

Recap

- CPA/CCA Security for Public Key Crypto
- Key Encapsulation Mechanism
- El-Gamal

Recap

- Plain RSA
- Public Key (pk): $\mathrm{N} = \mathrm{pq}$, 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 $\phi(N)$

$$\mathbf{Enc}_{\boldsymbol{pk}}(\boldsymbol{m}) = \boldsymbol{m}^{\boldsymbol{e}}\ \boldsymbol{mod}\ \boldsymbol{N}$$
$$\mathbf{Dec}_{\boldsymbol{sk}}(\boldsymbol{c}) = \boldsymbol{c}^{\boldsymbol{d}}\ \boldsymbol{mod}\ \boldsymbol{N}$$

- Decryption Works because
  $[c^d \mathrm{mod\ N}] = [m^{ed} \mathrm{mod\ N}] = \left[m^{[ed\ mod\ \boldsymbol{\phi}(N)]} \mathrm{mod\ N}\right] = [m\ \mathrm{mod\ N}]$

(Review) Attacks on Plain RSA

- We have not introduced security models like CPA-Security or CCA-security for Public Key Cryptosystems

- However, notice that (Plain) RSA Encryption is stateless and deterministic.

→Plain RSA is not secure against chosen-plaintext attacks

- Plain RSA is also highly vulnerable to chosen-ciphertext attacks
  - Attacker intercepts ciphertext c of secret message m
  - Attacker generates ciphertext c’ for secret message 2m
  - Attacker asks for decryption of c’ to obtain 2m
  - Divide by 2 to recover original message m

(Review) More Plain RSA Attacks

- Encrypted messages with low entropy are vulnerable to a brute-force attack.
  - If m < B then attacker can recover m after at most B queries to encryption oracle (using public key)
  - In fact, there is an attack that runs in time $\mathrm{T} = B^{\frac{1}{2}+\varepsilon}$
- Coppersmith Attacks
  - Recover partially known message m from ciphertext (when e is small)
  - Factor N=pq when we have good estimate $\tilde{p} \approx p$

More Attacks: Encrypting Related Messages

- Sender encrypts m and $m + \delta$, where offset δ is known to attacker

- Attacker intercepts

$$c_1 = \mathrm{Enc}_{pk}(m) = m^e \; mod \; N$$

and

$$c_2 = \mathrm{Enc}_{pk}(m + \delta) = (m + \delta)^e \; mod \; N$$

- Attacker defines polynomials

$$f_1(x) = x^e - c_1 \, mod \; N$$

and

$$f_2(x) = (x + \delta)^e - c_2 \, mod \; N$$

More Attacks: Encrypting Related Messages

$$c_1 = \mathrm{Enc}_{pk}(m) = m^e \; mod\; N$$
$$c_2 = \mathrm{Enc}_{pk}(m+\delta) = (m+\delta)^e \; mod\; N$$

- Attacker defines polynomials

$$f_1(x) = x^e - c_1 \, mod\; N$$

and

$$f_2(x) = (x+\delta)^e - c_2 \, mod\; N$$

- Both polynomials have a root at x=m, thus (x-m) is a factor of both polynomials
- The GCD operation can be extended to operate over polynomials ☺
- GCD($f_1(x)$, $f_2(x)$) reveals the factor (x-m), and hence the message m

Sending the Same Message to Multiple Receivers

- Homework 4 Bonus Question
  - e=3
  - $c_1 = [m^3 \ mod \ N_1]$
  - $c_2 = [m^3 \ mod \ N_2]$
  - $c_2 = [m^3 \ mod \ N_3]$

- Attacker receives all (e=3) ciphexts (sent to Alice, Bob and Jane) and can recover m.

- **Homework 4 Hint:** The solution involves the Chinese Remainder Theorem

Apply GCD to Pairs of RSA Moduli?

- **Fact**: If we pick two random RSA moduli N_1 and N_2 then except with negligible probability $\gcd(N_1, N_2) = 1$
- In theory the attack shouldn't work, but…
- In practice, many people generated RSA moduli using weak pseudorandom number generators.
  - .5% of TLS hosts
  - .03% of SSH hosts
- See https://factorable.net

Dependent Keys Part 1

- Suppose an organization generates N=pq and a pair (e_i,d_i) for each employee i subject to the constraints $e_i d_i = 1 \bmod \phi(N)$.

- **Question**: Is this secure?

- **Answer**: No, given $e_i d_i$ employee i can factor N (and then recover everyone else's secret key).
- See Theorem 8.50 in the textbook

Dependent Keys Part 2

- Suppose an organization generates N=pq and a pair (e_i,d_i) for each employee i subject to the constraints $e_i d_i = 1 \bmod \phi(N)$.

- Suppose that each employee is trusted (so it is ok if employee i factors N)

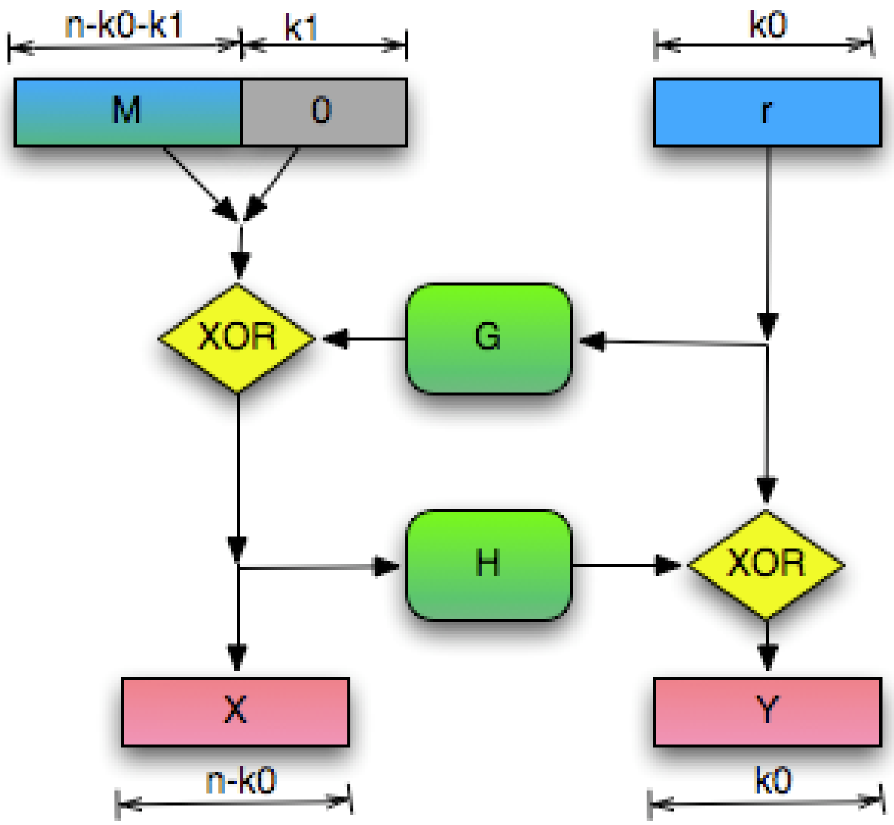
- Suppose that a message m is encrypted and sent to employee 1 and 2.
- Attacker intercepts $c_1 = [m^{e_1} \ mod\ N]$ and $c_2 = [m^{e_2}\ mod\ N]$

Dependent Keys Part 2

- Suppose an organization generates N=pq and a pair (e_i,d_i) for each employee i subject to the constraints $e_i d_i = 1 \bmod \phi(N)$.
- Suppose that a message m is encrypted and sent to employee 1 and 2.
- Attacker intercepts $c_1 = [m^{e_1} \bmod N]$ and $c_2 = [m^{e_2} \bmod N]$
- If **gcd**$(e_1,e_2)=1$ (which is reasonably likely) then attacker can run extended GCD algorithm to find X,Y such that $Xe_1+Ye_2=1$.
  $$[c_1^X c_2^Y \bmod N] = [m^{Xe_1} m^{Ye_2} \bmod N] = [m^{Xe_1+Ye_2} \bmod N] = m$$

RSA-OAEP
(Optimal Asymmetric Encryption Padding)

- $\mathbf{Enc}_{\boldsymbol{pk}}(m; r) = [(x \parallel y)^e \ mod\ N]$
- Where $x \parallel y \leftarrow \mathrm{OAEP}(m \parallel 0^{k_1} \parallel r)$
- $\mathbf{Dec}_{\boldsymbol{sk}}(c) =$
- $\tilde{m} \leftarrow [(c)^d \, mod\ N]$
- If $\|\tilde{m}\| > n$ return fail
- $m \parallel z \parallel r \leftarrow \mathrm{OAEP}^{-1}(\tilde{m})$
- If $z \neq 0^{k_1}$ then output fail
- Otherwise output m



Recap RSA-Assumption

RSA-Experiment: RSA-INV$_{A,n}$

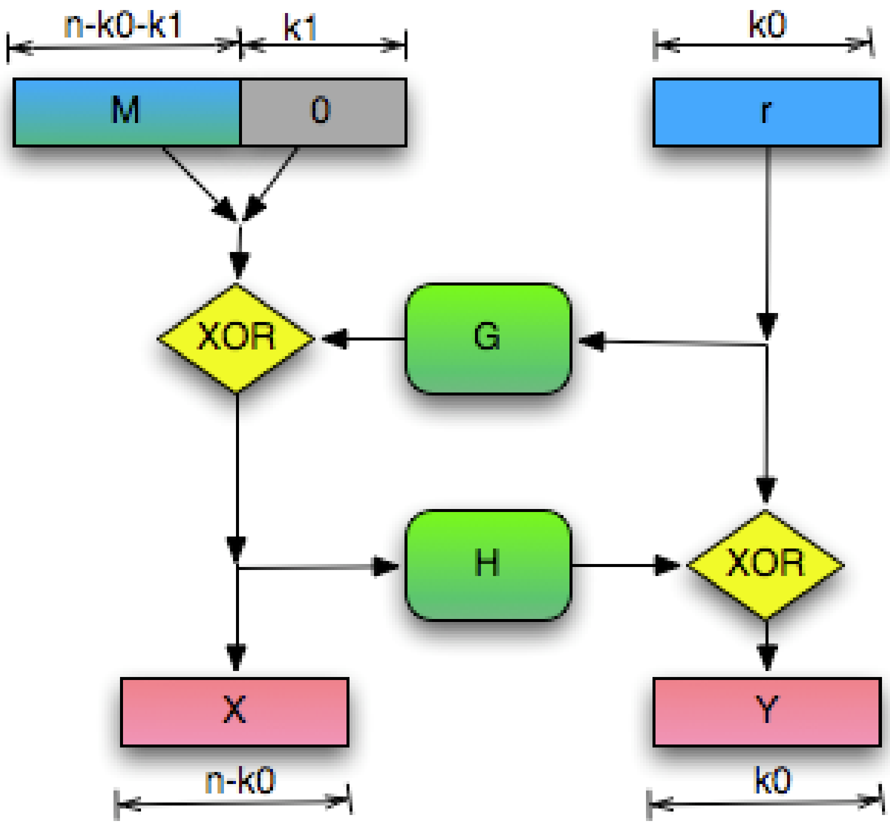
1. **Run KeyGeneration**(1^n) **to obtain (N,e,d)**
2. **Pick uniform** $\mathrm{y} \in \mathbb{Z}_{\mathrm{N}}^*$
3. Attacker A is given N, e, y and outputs $\mathrm{x} \in \mathbb{Z}_{\mathrm{N}}^*$
4. Attacker wins (RSA−INV$_{A,n}$=1) if $x^e = \mathrm{y} \bmod \mathrm{N}$

$$\forall PPT\ A\ \exists\mu\ (\text{negligible})\ \text{s.t}\ \Pr\left[\text{RSA−INV}_{A,n} = 1\right] \leq \mu(n)$$

RSA-OAEP
(Optimal Asymmetric Encryption Padding)

Theorem: If we model G and H as Random oracles then RSA-OAEP is a CCA-Secure public key encryption scheme (given RSA-Inversion assumption).

Bonus: One of the fastest in practice!



PKCS #1 v2.0

- Implementation of RSA-OAEP

- James Manger found a chosen-ciphertext attack.

- What gives?

PKCS #1 v2.0 (Bad Implementation)

- $\mathbf{Enc}_{\boldsymbol{pk}}\ (m; r) = [(x \parallel y)^e \ \ mod\ N]$
- Where $x \parallel y\ \leftarrow \mathrm{OAEP}(m \parallel 0^{k_1} \parallel r)$
- $\mathbf{Dec}_{\boldsymbol{sk}}\ (c) =$
- $\widetilde{m} \leftarrow [(c)^d\, mod\ N]$
- **If $\|\widetilde{\boldsymbol{m}}\| > \boldsymbol{n}$ return Error Message 1**
- $m \parallel z \parallel r \leftarrow \mathrm{OAEP}^{-1}(\widetilde{m})$
- **If $\boldsymbol{z} \neq \mathbf{0}^{\boldsymbol{k}_1}$ then output Error Message 2**
- Otherwise output m

- $\mathbf{Enc}_{\boldsymbol{pk}}\ (m; r) = [(x \parallel y)^e \ \ mod\ N]$
- Where $x \parallel y\ \leftarrow \mathrm{OAEP}(m \parallel 0^{k_1} \parallel r)$
- $\mathbf{Dec}_{\boldsymbol{sk}}\ (c) =$
- $\widetilde{m} \leftarrow [(c)^d\, mod\ N]$
- **If $\|\widetilde{\boldsymbol{m}}\| > \boldsymbol{n}$ return Error Message 1**
- $m \parallel z \parallel r \leftarrow \mathrm{OAEP}^{-1}(\widetilde{m})$
- **If $\boldsymbol{z} \neq \mathbf{0}^{\boldsymbol{k}_1}$ then output Error Message 2**
- Otherwise output m

PKCS #1 v2.0 (Attack)

- Manger's CCA-Attack recovers secret message
  - **Step 1:** Use decryption oracle to check if $2\widetilde{m} \geq \ 2^n$
  - $c = [(\widetilde{m})^e \, mod \, N]$ →$2^e c = [(2\widetilde{m})^e \, mod \, N]$
- Requires $||\boldsymbol{N}||$ $queries\ to\ decryption\ oracle$.
- Attack also works as a side channel attack
  - Even if error messages are the same the time to respond could be different in each case.
- **Fix:** Implementation should return same error message and should make sure that the time to return each error is the same.

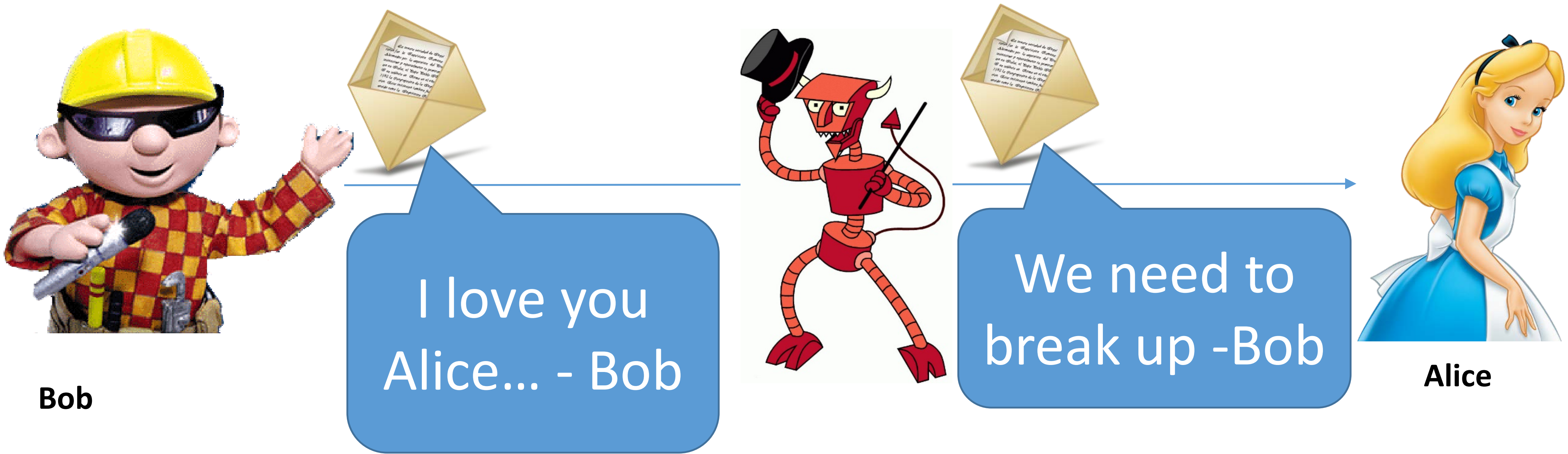
Week 13: Topic 3: Digital Signatures (Part 1)

Recap

- CPA/CCA Security for Public Key Crypto
- Key Encapsulation Mechanism
- El-Gamal/RSA-OAEP

What Does It Mean to “Secure Information”

- Confidentiality (Security/Privacy)
  - Only intended recipient can see the communication
- Integrity (Authenticity)
  - The message was actually sent by the alleged sender



Encryption/MACs/Signatures

- (Public/Private Key) Encryption: Focus on Secrecy
  - But does not promise integrity
- MACs/Digital Signatures: Focus on Integrity
  - But does not promise secrecy
- Digital Signatures
  - Public key analogue of MAC

Digital Signature: Application

- Verify updates to software package

- Vendor generates (**sk,pk**) for Digital Signature scheme and packages **pk** in the original software bundle

- An update **m** should be signed by vendor using secret key **sk**

- **Security**: Malicious party should not be able to generate signature for new update **m’**

Digital Signature vs MACs

- Application: Validate updates to software
- Problem can be addressed by MACs, but there are several problems
- Key Explosion: Vendor must sign update using every individual key
  - Thought Question: Why not use a shared Private key?
- Non-Transferable: If Alice validates an update from vendor she can not convince Bob that the update is valid
  - Bob needs to receive MAC directly from vendor

Digital Signatures vs MACs

- Publicly Verifiable
- Transferable
  - Alice can forward digital signature to Bob, who is convinced (both Alice and Bob have the public key of the vendor)
- Non-repudiation
  - Can “certify” a particular message came from sender
- MACs do not satisfy non-repudiation
  - Suppose Alice reveals a shared key KAB along with a valid tag for a message m to a judge.
  - The judge should not be convinced the message was MACed by Bob. Why not?

Digital Signature Scheme

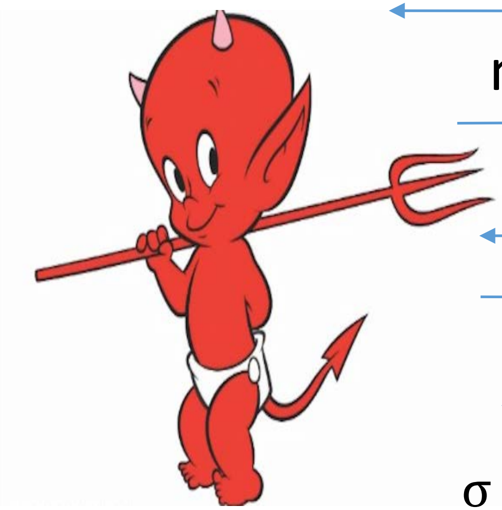
- Three Algorithms
  - $\mathrm{Gen}(1^n, R)$ (Key-generation algorithm)
    - Input: Random Bits R
    - Output: $(\boldsymbol{pk}, \boldsymbol{sk}) \in \mathcal{K}$
  - $\sigma \leftarrow \mathrm{Sign}_{\mathrm{sk}}(m, R)$ (Signing algorithm)
    - Input: Secret key sk message m, random bits R
    - 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

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

⋮

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

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

(pk,sk) = Gen(.)

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

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

Formally, let $\Pi = (\mathrm{Gen}, \mathrm{Sign}, \mathrm{Vrfy})$ denote the signature scheme, call the experiment $\mathrm{Sig-forge}_{A,\Pi}(\mathrm{n})$

We say that Π *is existentially unforgeable under an adaptive chosen message attack* (or just *secure*) if for all PPT adversaries A, there *is a* negligible function μ such that

$$\Pr[\mathrm{Sig-forge}_{A,\Pi}(n) = 1] \leq \mu(n)$$

Existential Unforgeability

- **Limitation**: Does not prevent replay attacks
  - $\sigma \leftarrow \text{Sign}_{sk}(\text{"}Pay\ Bob\ \$50\text{"}, R)$
  - If this is a problem then you can include timestamp in signature
- Unforgeability: does rule out the possibility attacker modifies a signature
- Plain RSA signatures are malleable (does not satisfy our security notion)
- **Remark:** By design signatures cannot hide all information about message *m*
  - Public Verification → Attacker can easily distinguish between a signature for m_1 and m_2

Plain RSA Signatures

- Plain RSA
- Public Key (pk): N = pq, e such that $\text{GCD}\big(\text{e}, \phi(N)\big) = 1$
  - $\phi(N) = (p-1)(q-1)$ 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)

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

Hash and Sign Paradigm

- Public-Key vs Private Key Encryption
  - Private Key Encryption is much more efficient (computationally)
- Similarly, natural signature schemes (e.g., RSA signatures) are much less efficient than MACs

- For long messages we can achieve same (amortized) efficiency

Hash and Sign Paradigm

- Suppose we have a Digital Signature Scheme for messages of length $\ell(n)$ and we want to sign a longer message $m \in \{0,1\}^*$.
- **Attempt 1:**

$$\mathrm{Sign}^*_{\mathrm{sk}}\,(m_1, m_2, \dots, m_k, R_1, \dots, R_k) = \mathrm{Sign}_{\mathrm{sk}}(m_1, R_1), \dots, \mathrm{Sign}_{\mathrm{sk}}(m_k, R_k)$$

- Problem?

Hash and Sign Paradigm

- Suppose we have a Digital Signature Scheme for messages of length $\ell(n)$ and we want to sign a longer message $m \in \{0,1\}^*$.

$$\mathrm{Sign}^*_{\langle \mathrm{sk}, s\rangle}(m_1, m_2, \dots, m_k, R) = \mathrm{Sign}_{\mathrm{sk}}(H(m_1, m_2, \dots, m_k), R)$$

$$\mathrm{Vrfy}^*_{\langle \mathrm{pk}, s\rangle}(m_1, m_2, \dots, m_k, \sigma) = \mathrm{Vrfy}_{\mathrm{pk}}(H^s(m_1, m_2, \dots, m_k), \sigma)$$

- Secure?

Hash and Sign Paradigm

- Suppose we have a Digital Signature Scheme for messages of length $\ell(n)$ and we want to sign a longer message $m \in \{0,1\}^*$.

$$\mathrm{Sign}^*_{\langle \mathrm{sk}, s \rangle}(m_1, m_2, \ldots, m_k, R) = \mathrm{Sign}_{\mathrm{sk}}(H^s(m_1, m_2, \ldots, m_k), R)$$

$$\mathrm{Vrfy}^*_{\langle \mathrm{pk}, s \rangle}(m_1, m_2, \ldots, m_k, \sigma) = \mathrm{Vrfy}_{\mathrm{pk}}(H^s(m_1, m_2, \ldots, m_k), \sigma)$$

- Secure?

Theorem 12.4. If $\Pi = (\mathrm{Gen}, \mathrm{Sign}, \mathrm{Vrfy})$ is a secure signature scheme for messages of length $\ell(n)$ and Π_H is collision resistant then the above construction is a secure signature scheme for arbitrary length messages.

Hash and Sign Paradigm

- Suppose we have a Digital Signature Scheme for messages of length $\ell(n)$ and we want to sign a longer message $m \in \{0,1\}^*$.

$$\mathrm{Sign}^*_{\langle \mathrm{sk}, s\rangle}(m_1, m_2, \ldots, m_k, R) = \mathrm{Sign}_{\mathrm{sk}}(H^s(m_1, m_2, \ldots, mk), R)$$

$$\mathrm{Vrfy}^*_{\langle \mathrm{pk}, s\rangle}(m_1, m_2, \ldots, mk, \sigma) = \mathrm{Vrfy}_{\mathrm{pk}}(H^s(m_1, m_2, \ldots, mk), \sigma)$$

Theorem 12.4. If $\Pi = (\mathrm{Gen}, \mathrm{Sign}, \mathrm{Vrfy})$ is a secure signature scheme for messages of length $\ell(n)$ and Π_H is collision resistant then the above construction is a secure signature scheme for arbitrary length messages.

Proof Sketch: If attacker wins security game with $\mathrm{Sign}^*_{\langle sk, s\rangle}$ then he outputs message $m \notin \mathfrak{Q}$ such that $\mathrm{Vrfy}^*_{\langle pk, s\rangle}(m, \sigma)$

Hash and Sign Paradigm

- Suppose we have a Digital Signature Scheme for messages of length $\ell(n)$ and we want to sign a longer message $m \in \{0,1\}^*$.

$$\mathrm{Sign}^*_{\langle sk,s\rangle}(m_1, m_2, \dots, m_k, R) = \mathrm{Sign}_{\mathrm{sk}}(H^s(m_1, m_2, \dots, mk), R)$$
$$\mathrm{Vrfy}^*_{\langle \mathrm{pk},s\rangle}(m_1, m_2, \dots, mk, \sigma) = \mathrm{Vrfy}_{\mathrm{pk}}(H^s(m_1, m_2, \dots, mk), \sigma)$$

Theorem 12.4. If $\Pi = (\mathrm{Gen}, \mathrm{Sign}, \mathrm{Vrfy})$ is a secure signature scheme for messages of length $\ell(n)$ and Π_H is collision resistant then the above construction is a secure signature scheme for arbitrary length messages.

Proof Sketch: If attacker wins security game with $\mathrm{Sign}^*_{\langle sk,s\rangle}$ then he outputs message $m \notin \mathcal{Q}$ such that $\mathrm{Vrfy}^*_{\langle pk,s\rangle}(m, \sigma)$

- Case 1: H(m)=H(m') for some $m' \notin \mathcal{Q}$

→break collision-resistance

- Case 2: H(m)≠ H(m') for all $m' \notin \mathcal{Q}$

→(break security of underlying signature scheme Π)

RSA-FDH

- **Full Domain Hash:** $H: \{0,1\}^* \to \mathbb{Z}_N^*$
- Given a message $m \in \{0,1\}^*$

$$\sigma = \mathrm{Sign}_{sk}\,(m) = H(m)^d \; mod \; N$$

Theorem 12.7: RSA-FDH is a secure signature scheme assuming that the RSA-Inversion problem is hard and H is modeled as a random oracle.

Remark: The domain of H (e.g.,SHA3) may be shorter than $\mathbb{Z}_N^*$.

Solution: Repeated application of H.

RSA-FDH

- **Full Domain Hash:** $H: \{0,1\}^* \rightarrow \mathbb{Z}_N^*$
- Given a message $m \in \{0,1\}^*$

$$\sigma = \mathrm{Sign}_{sk}\,(m) = H(m)^d \; mod \; N$$

Theorem 12.7: RSA-FDH is a secure signature scheme assuming that the RSA-Inversion problem is hard and H is modeled as a random oracle.

Proof Sketch: Given an RSA-Inversion challenge $\mathrm{c} = \; r^e \; mod \; N$ we will program the value $\mathrm{H(m)} = c \in \mathbb{Z}_N^*$ into the random oracle to trick the signature attacker into revealing $\mathrm{Sign}_{sk}(m) = r^{ed} = r \; mod \; N$.

One-Time Signature Scheme

- Weak notion of one-time secure signature schemes
  - Attacker makes one query to oracle $\mathrm{Sign}_{sk}(.)$ and then attempts to output forged signature for m’
  - If attacker sees two different signatures then guarantees break down
- Achievable from Hash Functions
  - No number theory!
  - No Random Oracles!

Lamport's Signature Scheme (from OWFs)

$$sk = \begin{bmatrix} x_{1,0} & x_{2,0} & x_{3,0} \\ x_{1,1} & x_{2,1} & x_{3,1} \end{bmatrix}$$

$$pk = \begin{bmatrix} y_{1,0} & y_{2,0} & y_{3,0} \\ y_{1,1} & y_{2,1} & y_{3,1} \end{bmatrix}$$

$$x_{i,j} \in \{0,1\}^n \ (uniform)$$
$$y_{i,j} = f\left(x_{i,j}\right)$$

Assumption: f is a One-Way Function

Lamport's Signature Scheme (from OWFs)

$$sk = \begin{bmatrix} x_{1,0} & x_{2,0} & x_{3,0} \\ x_{1,1} & x_{2,1} & x_{3,1} \end{bmatrix}$$

$$pk = \begin{bmatrix} y_{1,0} & y_{2,0} & y_{3,0} \\ y_{1,1} & y_{2,1} & y_{3,1} \end{bmatrix}$$

$$Sign_{sk}(011) = \left(x_{1,0}, x_{2,1}, x_{3,1}\right)$$

Lamport's Signature Scheme (from OWFs)

$$sk = \begin{bmatrix} x_{1,0} & x_{2,0} & x_{3,0} \\ x_{1,1} & x_{2,1} & x_{3,1} \end{bmatrix}$$

$$pk = \begin{bmatrix} y_{1,0} & y_{2,0} & y_{3,0} \\ y_{1,1} & y_{2,1} & y_{3,1} \end{bmatrix}$$

$$Sign_{sk}(011) = \left(x_{1,0}, x_{2,1}, x_{3,1}\right)$$

$$\mathrm{Vrfy}_{pk}\left(011, (x_1, x_2, x_3)\right) = \begin{cases} 1 & \text{if } f(x_1) = y_{1,0} \wedge f(x_2) = y_{2,1} \wedge f(x_3) = y_{3,1} \\ 0 & \text{otherwise} \end{cases}$$

Lamport's Signature Scheme

Theorem 12.16: Lamport's Signature Scheme is a secure one-time signature scheme (assuming f is a one-way function).

Proof Sketch: Signing a fresh message requires inverting $f\left(x_{i,j}\right)$ for random $x_{i,j}$.

Remark: Attacker can break scheme if he can request two signatures.

How?

Request signatures of both 0^n and 1^n.

Lamport's Signature Scheme

Remark: Attacker can break scheme if he can request two signatures.

How?

Request signatures of both 0^n and 1^n.

$$sk = \begin{bmatrix} x_{1,0} & x_{2,0} & x_{3,0} \\ x_{1,1} & x_{2,1} & x_{3,1} \end{bmatrix}$$

$$Sign_{sk}(000) = (x_{1,0}, x_{2,0}, x_{3,0})$$

$$Sign_{sk}(111) = (x_{1,1}, x_{2,1}, x_{3,1})$$

Secure Signature Scheme from OWFs

Theorem 12.22: secure/stateless signature scheme from collision-resistant hash functions.

- Collision Resistant Hash Functions do imply OWFs exist

Remark: Possible to construct signature scheme Π which is existentially unforgeable under an adaptive chosen message attacks using the minimal assumption that one-way functions exist.