RSA (2003)

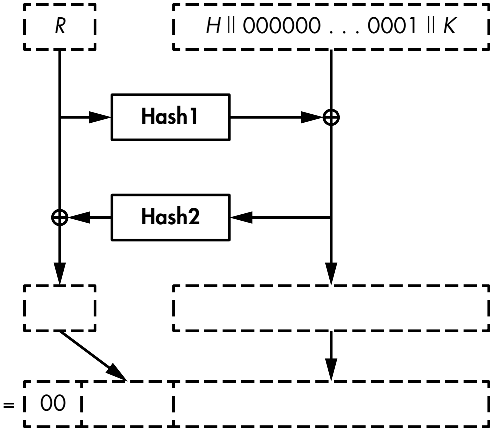
CS 553
CRYPTOGRAPHY

Lecture 25
More on RSA

Instructor
Dr. Dhiman Saha

Image Source: Google

Encrypting a symmetric key, K, with RSA-OAEP



H ← fixed parameter and R ← random bits

RSA Signatures

$$y = x^d \bmod n \qquad y^e = (x^d)^e \bmod n = x$$

- **No one** other than the private key holder knows the private exponent d,
- Only he/she can compute a signature $y = x^d \bmod n$ from some value x
- **Everyone** can verify $y^e \bmod n = x$ given the public exponent e
- Verified signature provides undeniability or **nonrepudiation**.

RSA-Enc Vs RSA-Sig

- Are RSA signatures as the converse of encryption?

Testbook RSA Signatures

Textbook RSA Signature

Signing a message, x, by directly computing $y = x^d \bmod n$, where x can be any number between 0 and $n - 1$.

- No Padding
- No Randomness

Aim

Get a (malicious) message M signed

- Find some value R such $R^e M \bmod n$ is a message that victim would sign.
- Get the signature $S = (R^e M)^d \bmod n$ from victim
- Derive signature of M i.e., M^d from S

$$\frac{S}{R} = \frac{(R^e M)^d}{R} = \frac{R M^d}{R} = M^d \bmod n$$

Aim

Get a (malicious) message M signed

- Find some value R such $R^e M \bmod n$ is a message that victim would sign.
- Get the signature $S = (R^e M)^d \bmod n$ from victim
- Derive signature of M i.e., M^d from S

$$\frac{S}{R} = \frac{(R^e M)^d}{R} = \frac{RM^d}{R} = M^d \bmod n$$

Aim

Get a (malicious) message M signed

- Find some value R such $R^e M \bmod n$ is a message that victim would sign.
- Get the signature $S = (R^e M)^d \bmod n$ from victim
- Derive signature of M i.e., M^d from S

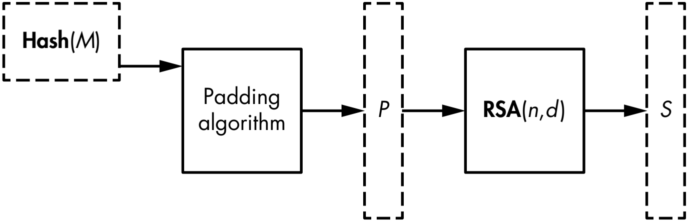
$$\frac{S}{R} = \frac{(R^e M)^d}{R} = \frac{R M^d}{R} = M^d \bmod n$$

Aim

Get a (malicious) message M signed

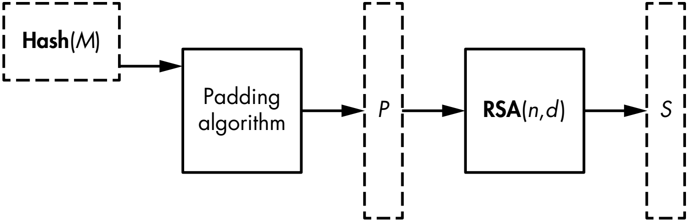
- Find some value R such $R^e M$ mod n is a message that victim would sign.
- Get the signature $S = (R^e M)^d$ mod n from victim
- Derive signature of M i.e., M^d from S

$$\frac{S}{R} = \frac{(R^e M)^d}{R} = \frac{R M^d}{R} = M^d \bmod n$$



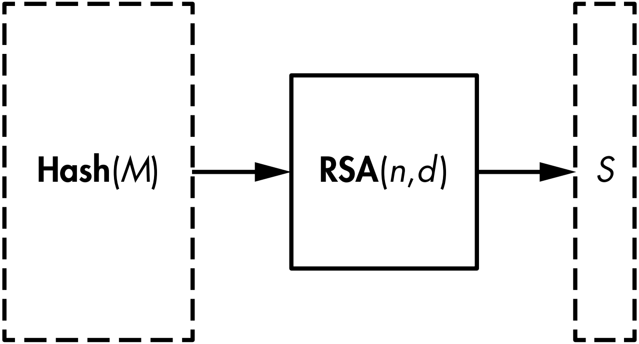
Signing a message, M, with RSA and with the PSS standard, where (n, d) is the private key

- Follows philosophy of RSA-OEAP
- **Note Hash-size less than what allowed by modulus**

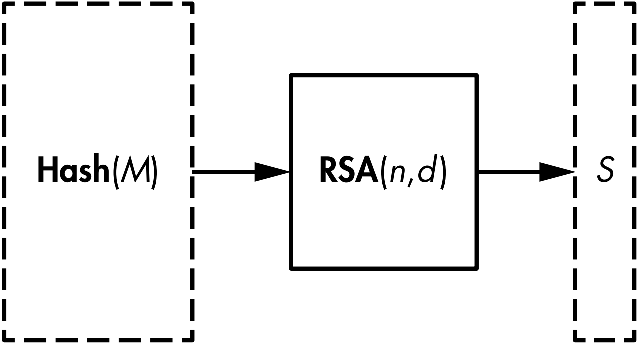


Signing a message, M, with RSA and with the PSS standard, where (n, d) is the private key

- Follows philosophy of RSA-OEAP
- **Note Hash-size less than what allowed by modulus**

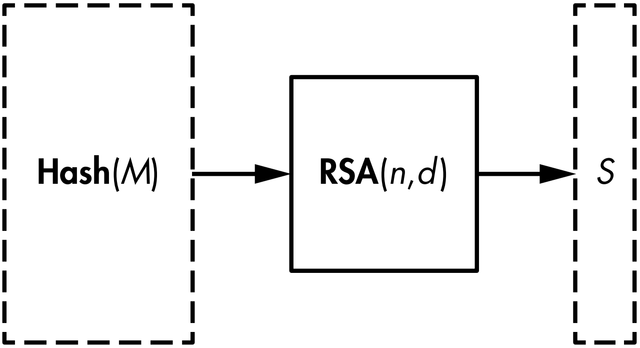


- Simplest signature scheme
- Uses the **maximal message space** allowed by RSA modulus
- Believed to be less secure than PSS due to lack of randomness



- Simplest signature scheme
- Uses the **maximal message space** allowed by RSA modulus
- Believed to be less secure than PSS due to lack of randomness

Full Domain Hash Signatures



- Simplest signature scheme
- Uses the **maximal message space** allowed by RSA modulus
- Believed to be less secure than PSS due to lack of randomness

```
expModNaive(x, e, n) {
    y = x
    for i = 1 to e - 1 {
        y = y * x  mod n
    }
    return y
}
```

Naive Exponentiation

The naive way to compute ($x^e \bmod n$) takes $e - 1$ multiplications

Exponent e consists of bits $e_{m-1}e_{m-2}\cdots e_1e_0$, where e_0 is the LSB

```
expMod(x, e, n) {
    y = x
    for i = m - 1 to 0 {
        y = y * y  mod n
        if e_i == 1 then
            y = y * x  mod n
    }
    return y
}
```

- Runs in time $O(m)$
- The naive algorithm runs in time $O(2^m)$

$$x^{26} = x^{11010_2} = x^{(h_4h_3h_2h_1h_0)_2}.$$

The algorithm scans the exponent bits, starting on the left with h_4 and ending with the rightmost bit h_0.

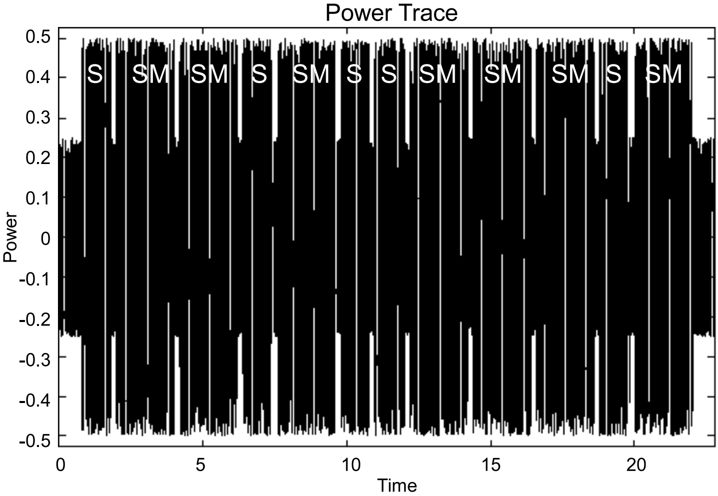
Step		
#0	$x = x^{\mathbf{1}_2}$	inital setting, bit processed: $h_4 = 1$
#1a	$(x^1)^2 = x^2 = x^{\mathbf{10}_2}$	SQ, bit processed: h_3
#1b	$x^2 \cdot x = x^3 = x^{10_2}x^{1_2} = x^{\mathbf{11}_2}$	MUL, since $h_3 = 1$
#2a	$(x^3)^2 = x^6 = (x^{11_2})^2 = x^{\mathbf{110_2}}$	SQ, bit processed: h_2
#2b		no MUL, since $h_2 = 0$
#3a	$(x^6)^2 = x^{12} = (x^{110_2})^2 = x^{\mathbf{1100_2}}$	SQ, bit processed: h_1
#3b	$x^{12} \cdot x = x^{13} = x^{1100_2}x^{1_2} = x^{\mathbf{1101}_2}$	MUL, since $h_1 = 1$
#4a	$(x^{13})^2 = x^{26} = (x^{1101_2})^2 = x^{\mathbf{11010_2}}$	SQ, bit processed: h_0
#4b		no MUL, since $h_0 = 0$

A Side Channel Attack on Fast Exponentiation

S&M Sequence Based on Private Exponent

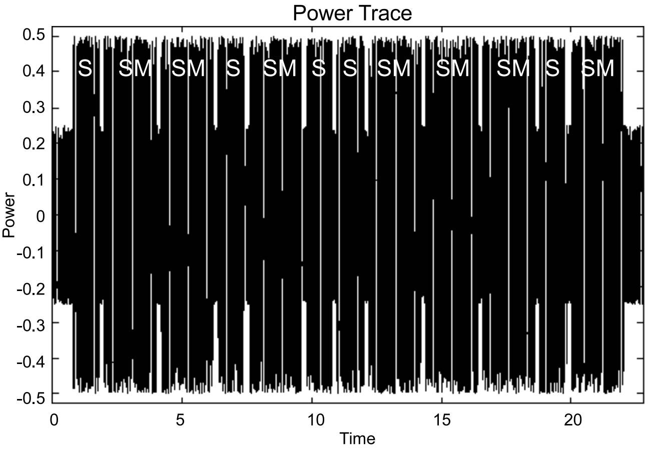
operations:	S	SM	SM	S	SM	S	S	SM	SM	SM	S	SM
private key:	0	1	1	0	1	0	0	1	1	1	0	1

Corresponding Power Trace



The power trace of an RSA implementation

operations:	$S\ SM\ SM\ S\ SM\ S\ S\ SM\ SM\ SM\ S\ SM$
private key:	0 1 1 0 1 0 0 1 1 1 0 1



Counter-measure

- What would be the simplest way to protect against this?