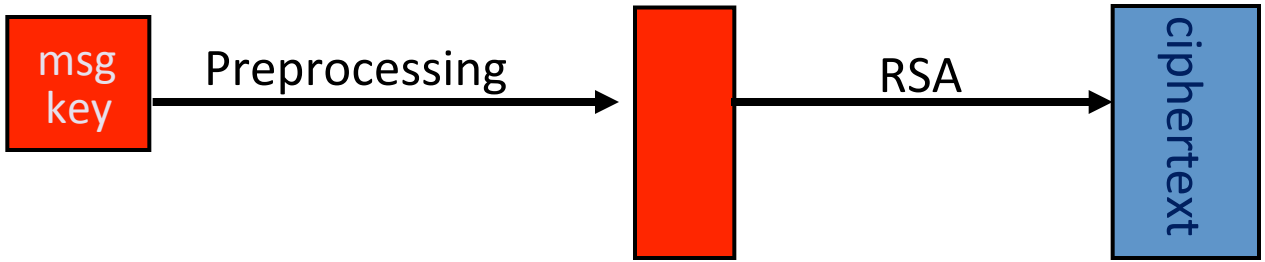
Public Key Encryption from trapdoor permutations

PKCS 1

RSA encryption in practice

Never use textbook RSA.

RSA in practice (since ISO standard is not often used) :

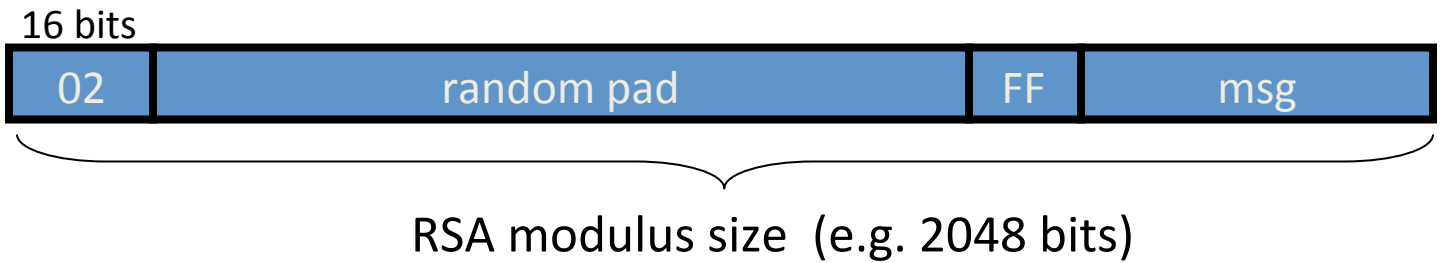


Main questions:

- How should the preprocessing be done?
- Can we argue about security of resulting system?

PKCS1 v1.5

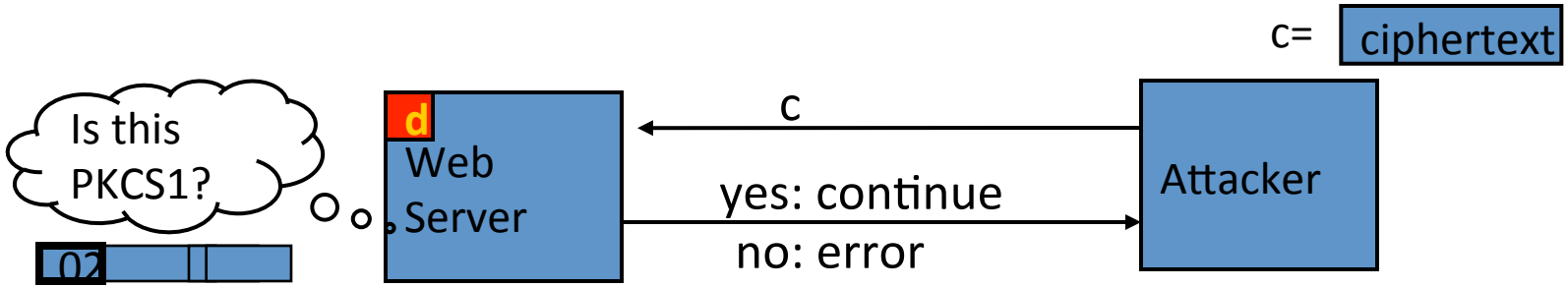
PKCS1 mode 2: (encryption)



- Resulting value is RSA encrypted
- Widely deployed, e.g. in HTTPS

Attack on PKCS1 v1.5 (Bleichenbacher 1998)

PKCS1 used in HTTPS:

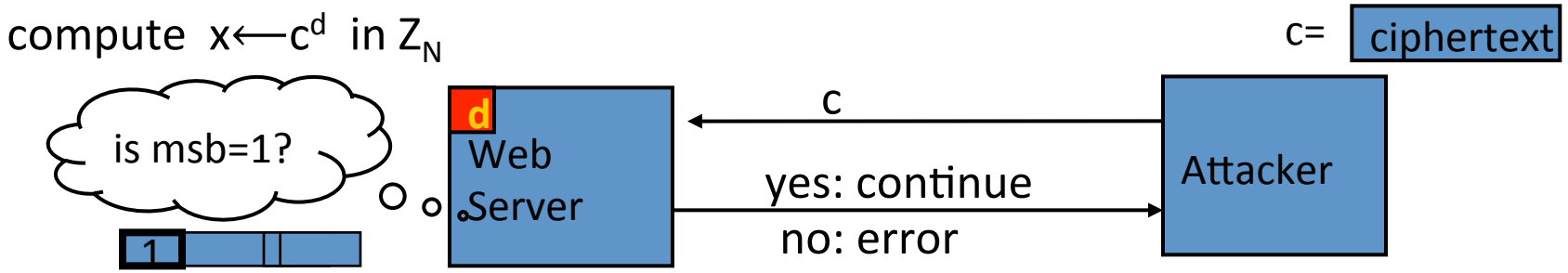


$\Rightarrow$ attacker can test if 16 MSBs of plaintext = '02'

Chosen-ciphertext attack: to decrypt a given ciphertext c do:

- Choose $r \in Z_N$. Compute $c' \leftarrow r^e \cdot c = (r \cdot \text{PKCS1}(m))^e$
- Send c' to web server and use response

Baby Bleichenbacher



Suppose N is $N = 2^n$ (an invalid RSA modulus). Then:

- Sending c reveals msb(x)
- Sending $\mathbf{2^e \cdot c = (2x)^e}$ in Z_N reveals msb(2x mod N) = $\mathbf{msb_2(x)}$
- Sending $\mathbf{4^e \cdot c = (4x)^e}$ in Z_N reveals msb(4x mod N) = $\mathbf{msb_3(x)}$
- … and so on to reveal all of x

HTTPS Defense (RFC 5246)

Attacks discovered by Bleichenbacher and Klima et al. … can be avoided by treating incorrectly formatted message blocks … in a manner indistinguishable from correctly formatted RSA blocks. In other words:

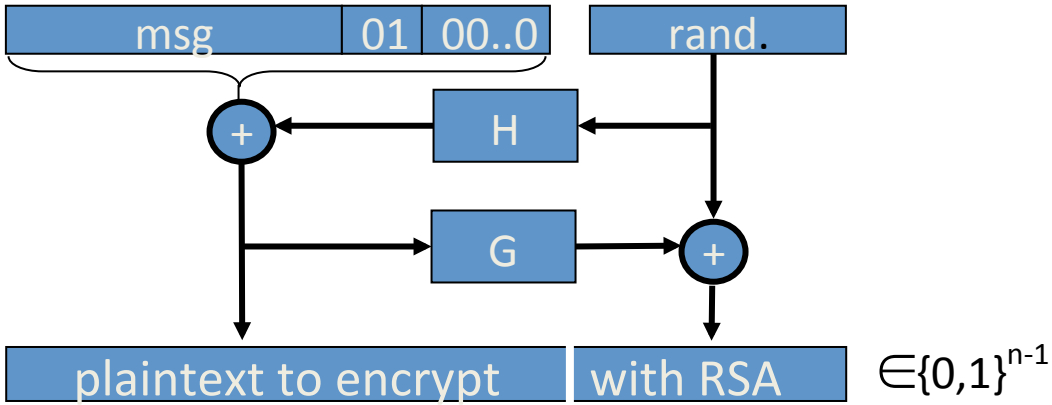
1. *Generate a string **R** of 46 random bytes*
2. *Decrypt the message to recover the plaintext M*
3. *If the PKCS#1 padding is not correct*

   *pre_master_secret = **R***

PKCS1 v2.0: OAEP

New preprocessing function: OAEP [BR94]

check pad
on decryption.
reject CT if invalid.



Thm [FOPS'01] : RSA is a trap-door permutation $\Rightarrow$
RSA-OAEP is CCA secure when H,G are *random oracles*

in practice: use SHA-256 for H and G

OAEP Improvements

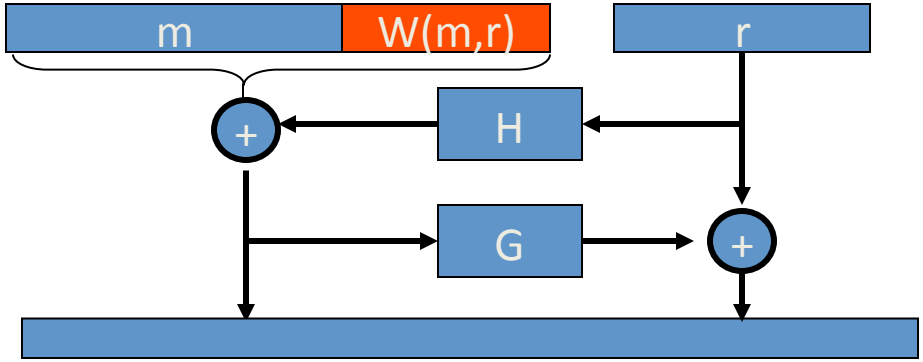
OAEP+: [Shoup'01]

∀ trap-door permutation F

F-OAEP+ is CCA secure when

H,G,W are *random oracles*.

During decryption validate W(m,r) field.

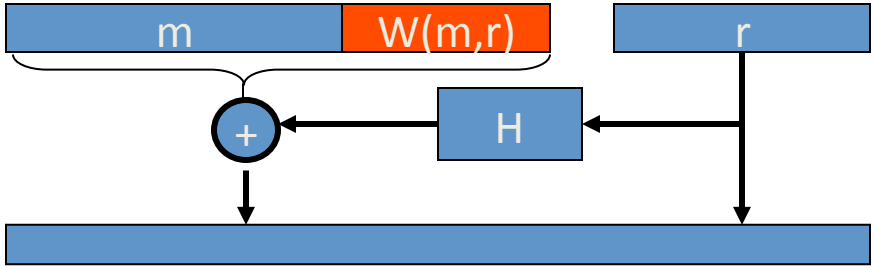


SAEP+: [B'01]

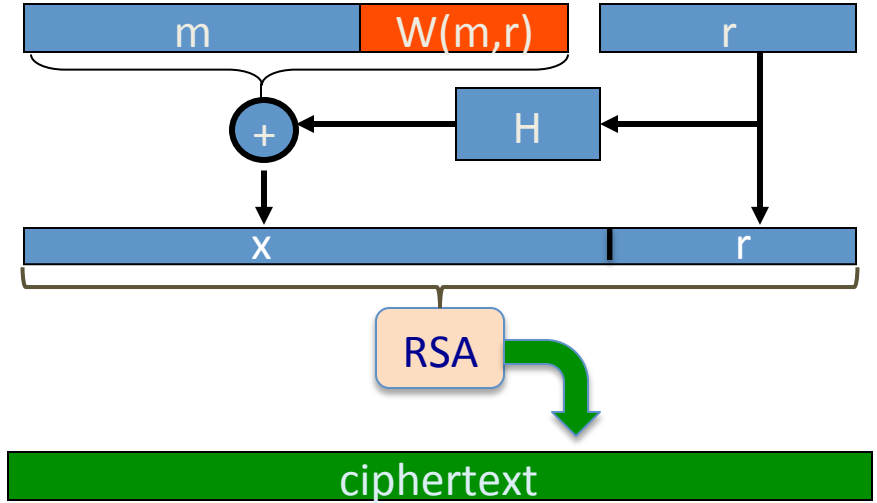
RSA (e=3) is a trap-door perm ⇒

RSA-SAEP+ is CCA secure when

H,W are *random oracle*.



How would you decrypt an SAEP ciphertext **ct** ?



- ○ $(x,r) \leftarrow RSA^{-1}(sk,ct)$, $(m,w) \leftarrow x \oplus H(r)$, output m if $w = W(m,r)$
- ○ $(x,r) \leftarrow RSA^{-1}(sk,ct)$, $(m,w) \leftarrow r \oplus H(x)$, output m if $w = W(m,r)$
- ○ $(x,r) \leftarrow RSA^{-1}(sk,ct)$, $(m,w) \leftarrow x \oplus H(r)$, output m if $r = W(m,x)$

Subtleties in implementing OAEP [M '00]

```
OAEP-decrypt(ct):
        error = 0;
        .........
        if ( RSA^-1(ct) > 2^(n-1) )
              { error =1;  goto exit; }
        .........
        if ( pad(OAEP^-1(RSA^-1(ct))) != "01000" )
              { error = 1;  goto exit; }
```

Problem: timing information leaks type of error

$\Rightarrow$ Attacker can decrypt any ciphertext

Lesson: Don't implement RSA-OAEP yourself !

End of Segment