PKCS1, OAEP, OAEP+, SAEP+

Josh Spieth

Background

- Given a trapdoor function defined over (X, Y), PKCS1 is an attempt to simplify our PKE schemes to have the ciphertext solely be one element in Y
- E_{RSA}, or SRSA as we studied in class, outputs (y,c) where y is in $\mathbb{Z}^*_n$ and c is a symmetric ciphertext.
- Let’s look at how we can design a scheme that gives us just one element as our ciphertext

Base Function (Textbook RSA with padding)

Algorithm KG
$(N, p, q, e, d) \xleftarrow{\$} K_{RSA}$
$pk \leftarrow (N, e)$
$sk \leftarrow (N, d)$
Return (pk,sk)

Algorithm $\mathcal{E}_{pk}(M)$
$S \leftarrow Pad(M)$
$C \leftarrow S^e \bmod N$
Return C

Algorithm $\mathcal{D}_{sk}(C)$
$T \leftarrow C^d \bmod N$
$M \leftarrow Unpad(T)$
Return M

- Our goal is to define the functions Pad and Unpad such that S and T are elements of X, and the encryption process is at least CPA, and ideally CCA secure

First attempts

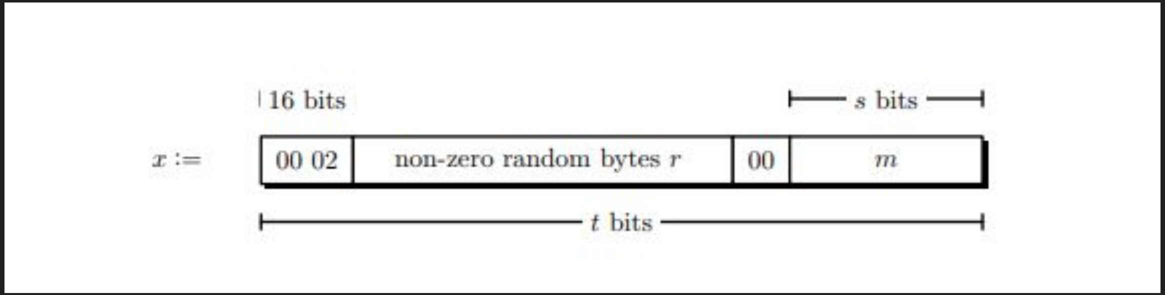
Algorithm KG	Algorithm $\mathcal{E}_{pk}(M)$	Algorithm $\mathcal{D}_{sk}(C)$
$(N, p, q, e, d) \xleftarrow{\$} K_{RSA}$ $pk \leftarrow (N, e)$ $sk \leftarrow (N, d)$ Return (pk,sk)	$S \leftarrow Pad(M)$ $C \leftarrow S^e$ mod N Return C	$T \leftarrow C^d$ mod N $M \leftarrow Unpad(T)$ Return M

(Assume size of strings in domain space X = t, size of strings in message space M = s)

Algorithm $BadPadding_{pk}(M)$	Algorithm $BadUnpadding_{sk}(M')$
$M' \leftarrow 0^n \| M$ such that $\lvert M' \rvert =$ t Return M'	$M \leftarrow M'[0 : s]$ Return M

- This is obviously a bad scheme, it is deterministic and if textbook RSA is insecure (which it is), then this is insecure

PKCS1 Padding scheme:



- Note that 00, 02 and 00 are listed in hex, each 2 digits are representing 8 bits
- PKCS1 is designed with bytes in mind, everything will function in multiples of 8 bits

PKCS1 Padding scheme

Algorithm KG
$(N, p, q, e, d) \xleftarrow{\$} K_{RSA}$
$pk \leftarrow (N, e)$
$sk \leftarrow (N, d)$
Return (pk,sk)

Algorithm $\mathcal{E}_{pk}(M)$
$S \leftarrow Pad(M)$
$C \leftarrow S^e$ mod N
Return C

Algorithm $\mathcal{D}_{sk}(C)$
$T \leftarrow C^d$ mod N
$M \leftarrow Unpad(T)$
Return M

(Assume size of strings in domain space X $= 0^8||0,1^{t-8}$, size of strings in message space M = s, s.t. $s \leq t - 88$.)

(r' is random bits of length t-88 given to the padding algorithm)

Algorithm $PKCS1Pad_{pk}(M, r')$
$M' \leftarrow (0x00||0x02||r||0x00||M)$
with r $= r'[: x]$ such that $|M'| = t$
Return M'

Algorithm $PKCS1Unpad_{sk}(M')$
parse M' $(0x00||0x02||non - zero - bytes - r||0x00||M)$
if parse fails, return reject, else return M

- Note domain space is defined to make all messages smaller than the RSA modular, and message space is limited in size to allow for padding scheme

PKCS1 Security

- PKCS1 is not fully secure, especially against chosen ciphertext attacks.
- First famous attack, Bleichenbacher's attack, is based on a SSL 3.0 protocol that used PKCS1 to set up secure communication between client and server
- The setup worked by the client choosing a 48 byte key, encrypting it with RSA-PKCS1 under the servers public key, and sending the ciphertext to the server
- The server then attempts to decrypt the ciphertext. If the PKCS1 decoding rejects, the server returns an abort message, otherwise it continues as normal

Bleichenbacher's attack

$$P_x(r) := \begin{cases} 1 & \text{if } x \cdot r \text{ in } \mathbb{Z}_n \text{ is a valid PKCS1 encoding;} \\ 0 & \text{otherwise.} \end{cases}$$

- Assume the attacker has some c intercepted from a previous communication to the server it wishes to decrypt.
- The attacker knows that $c = M^e \bmod n$
- The attacker can multiply c by any number r, and find out whether $rM^e \bmod n$ is a valid PKCS1 padded message
- Approximately every 3000-7000 attempts at s will lead to a valid padded message, and by figuring out which values for s give you a valid message, you can deduce M

Defense attempts

1. generate a string r of 48 random bytes,
2. decrypt the RSA-PKCS1 ciphertext to recover the plaintext m,
3. if the PKCS1 padding is invalid, or the length of m is not exactly 48 bytes:
4. set $m \leftarrow r$
5. return m

- TLS was a new server side protocol developed to defend against this attack
- The attacker now can't see initially when the server rejects a message or not, but through timing attacks later in the servers handling of the message, the same attack is still possible

OAEP

- After the failure of PKCS1, cryptographers wanted a padding scheme that was CCA-secure, and the first attempt was Optimal Asymmetric Encryption Padding (1994, Bellare and Rogaway)
- OAEP uses hash functions to gain extra security

OAEP

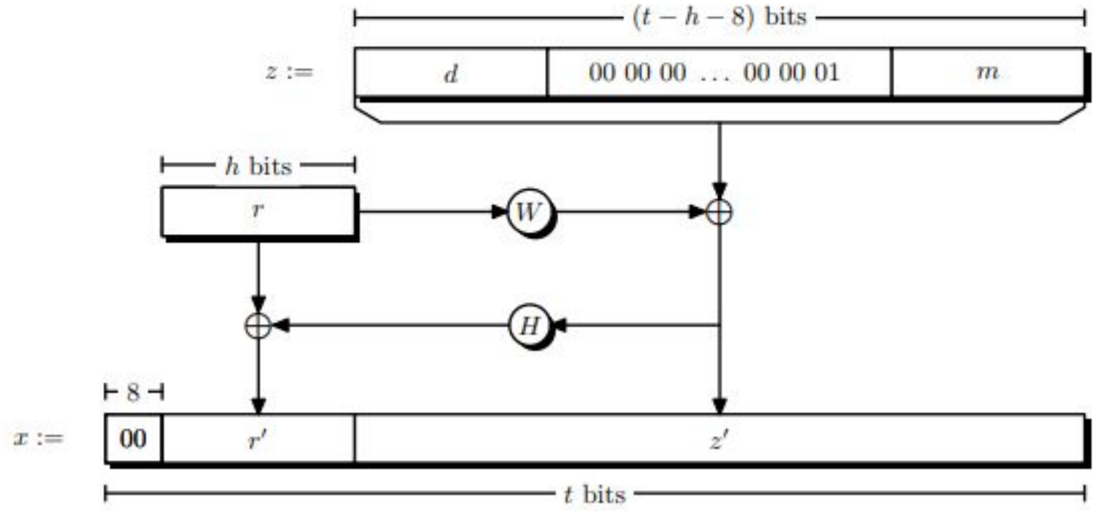


Figure 12.7: OAEP padding using hash functions H and W, and optional associated data d

- Note that the hash functions go back and forth between $\{0,1\}^{t-h-8}$ and $\{0,1\}^h$, h must be sufficiently large to be collision resistant

One wayness vs Partial one wayness

- Recall our definition of one wayness for trapdoor functions:

Definition

Let K_{rsa} be an RSA generator with security parameter k.
Let A be an algorithm.

Experiment $\mathbf{Exp}^{\text{ow-kea}}_{K_{rsa}}(A)$
$((N,e),(N,p,q,d)) \xleftarrow{\$} K_{rsa}$
$x \xleftarrow{\$} \mathbf{Z}^*_N ; y \leftarrow x^e \bmod N$
$x' \xleftarrow{\$} A(N,e,y)$
If $x = x'$ then 1 else 0

$$\mathbf{Adv}^{\text{ow-kea}}_{K_{rsa}}(A) = \Pr\left[\mathbf{Exp}^{\text{ow-kea}}_{K_{rsa}}(A) = 1\right].$$

- Partial one wayness works identically, but instead of guessing all of x, the adversary simply has to guess some of the bits of x

OAEP security

- OAEP is better than PKCS1, but there are plausible secure trapdoor functions T for which OAEP is not CCA secure
- OAEP is secure if and only if the trapdoor function which it is based on is secure under the partial one way definition, not just one way secure
- It turns out that any one way secure RSA scheme is also partial one way secure, and thus OAEP-RSA can be done securely (though it can be difficult to implement)

OAEP+

- OAEP+ is a modification of OAEP to make it secure with all one way trapdoor functions, not just partial one way trapdoors
- Instead of the block of zeros put into z in OAEP, we take H'(m,r) for some hash function H'

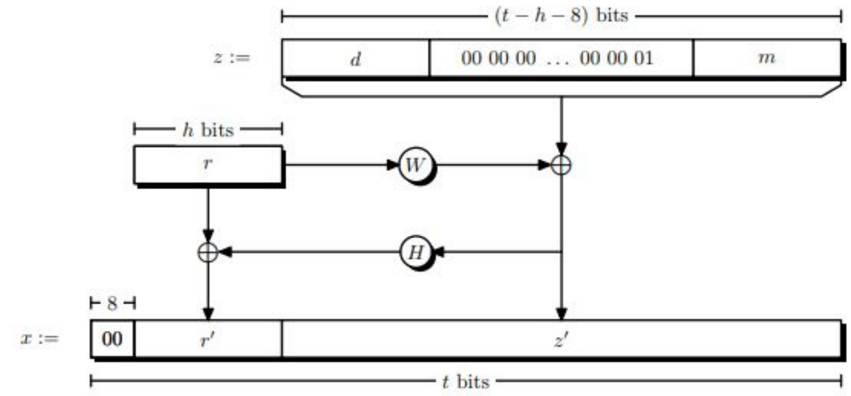


Figure 12.7: OAEP padding using hash functions H and W, and optional associated data d

SAEP+

- SAEP+ is a simpler CCA secure padding scheme for RSA specifically
- It uses a longer randomizer than OAEP, but only one hash function
- Simply hash r and xor with z

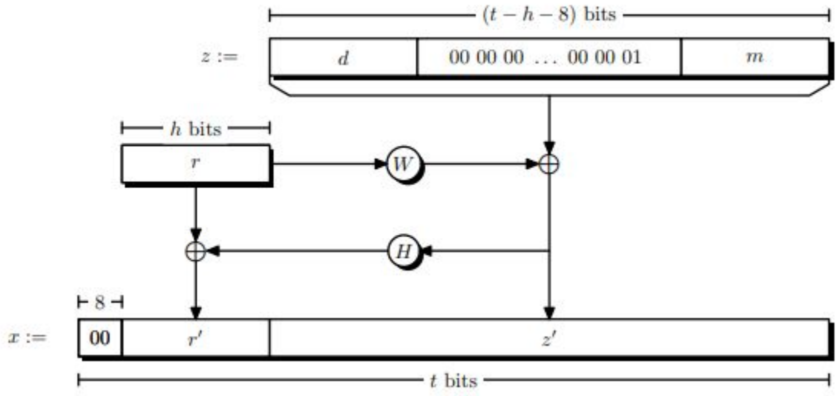


Figure 12.7: OAEP padding using hash functions H and W, and optional associated data d

Sources

- A Graduate Course in Applied Cryptography, Dan Boneh and Victor Shoup