Katz, Lindell
Introduction to Modern Cryptrography

Slides Chapter 11, part 2

Markus Bläser, Saarland University

Plain RSA

Algorithm 11.12: RSA key generation

Input: security parameter 1^n
Output: N, e, d with $ed = 1 \mod \phi(N)$

- $(N, p, q) \leftarrow \text{GenModulus}(1^n)$
  // p and q are primes with $N = pq$.
- $\phi(N) := (p-1)(q-1)$
- choose $e > 0$ such that $\gcd(e, \phi(N)) = 1$
- compute $d := e^{-1} \mod \phi(N)$
  // Using the extended Euclid algorithm
- return N, e, d.

Plain RSA (2)

Construction 11.16

- Gen: on input 1^n, run GenRSA(1^n) to obtain $\mathrm{N}, e, \mathrm{d}$. The public key is (N, e) and the private key is (N, d).
- Enc: on input a public key $pk = (\mathrm{N}, e)$ and a message $\mathrm{m} \in \mathbb{Z}_{\mathrm{N}}^*$, compute the ciphertext $\mathrm{c} := \mathrm{m}^e \mod \mathrm{N}$.
- Dec: on input a private key $sk = (\mathrm{N}, \mathrm{d})$ and a ciphertext $\mathrm{c} \in \mathbb{Z}_{\mathrm{N}}^*$, return the message $\mathrm{m} := \mathrm{c}^{\mathrm{d}} \mod \mathrm{N}$.

It works, since

$$\mathrm{c}^{\mathrm{d}} = (\mathrm{m}^e)^{\mathrm{d}} = \mathrm{m}^{e\mathrm{d}} = \mathrm{m} \mod \mathrm{N}.$$

Plain RSA is not CPA-secure!

Attacks on Plain RSA

Algorithm 11.28

Input: Public key (N, e), ciphertext c
Output: $m < 2^n$ such that $m^e = c \mod N$

- $T := 2^{\alpha n}$
- for $r := 1$ to T do
- $\quad x_r := c/r^e \mod N$
- sort the pairs (r, x_r), $r = 1, \ldots, T$, by their second component
- for $s = 1$ to T do
- $\quad$ if $x_r = s^e \mod N$ for some r
- $\quad\quad$ return $rs \mod N$

Padded RSA

Construction 11.30

- Gen: is the same as in plain RSA
- Enc: on input (N, e) and $m \in \{0,1\}^{\|N\|-\ell(n)-2}$, choose $r \in \{0,1\}^{\ell(n)}$ and let $\hat{m} = r \| m$ (as an element of $\mathbb{Z}_N^*$). Output $c := \hat{m}^e \mod N$.
- Dec: on input (N, d) and $c \in \mathbb{Z}_N^*$ compute $\hat{m} := c^d \mod N$ and output $\|N\| - \ell(n) - 2$ least significant bits of $\hat{m}$.

RSA PKCS #1 v1.5

- $pk = (\mathrm{N}, e)$
- k its length in bytes, that is, $2^{8(\mathrm{k}-1)} \leq \mathrm{N} < 2^{8\mathrm{k}}$.
- message length between 1 and $\mathrm{k} - 11$ bytes
- If m is D bytes long, then it is encrypted as

$$(0\mathrm{x}00||0\mathrm{x}02||\mathrm{r}||0\mathrm{x}00||\mathrm{m})^e \mod \mathrm{N}$$

  where r is a $\mathrm{k} - \mathrm{D} - 3$ bytes long random string not containing $0\mathrm{x}00$.
- Not CPA-secure, since the padding is too short.
- Not CCA-secure

Was replaced by newer versions.

Hard-core predicate for RSA

RSA hard-core predicate experiment $\text{RSA-lsb}_{\mathcal{A},\text{GenRSA}}(n)$**:**

1. Run $\text{GenRSA}(1^n)$ to obtain (N, e, d).
2. Choose a uniform $x \in \mathbb{Z}_N^*$ and compute $y := x^e \mod N$.
3. $\mathcal{A}$ is given N, e, y and outputs a bit b.
4. The output of the experiment is 1 if the least significant bit of x is b. Otherwise, it is 0.

Theorem (11.31)

If RSA is hard relative to GenRSA*, then for all ppt* $\mathcal{A}$ *,*

$$\Pr[\text{RSA-lsb}_{\mathcal{A},\text{GenRSA}}(n) = 1] \leq \frac{1}{2} + \text{negl}(n).$$

"Lsb is hardcore bit."

CPA-secure encryption

Construction 11.32

- Gen: on input 1^n, run GenRSA(1^n) to obtain (N, e, d). Output $pk = (N, e)$ and $sk = (N, d)$.
- Enc: On input pk and $m \in \{0,1\}$ choose a uniform $r \in \mathbb{Z}_N^*$ such that $\mathrm{lsb}(r) = m$. Output $c := r^e \mod N$.
- Dec: On input sk and ciphertext c, compute $r := c^d \mod N$ and output $\mathrm{lsb}(r)$.

Theorem

If RSA is hard relative to GenRSA*, then Construction 11.32 is CPA-secure.*

RSA-based KEM

Construction 11.34

- Gen: Run $\mathrm{GenRSA}(1^n)$ to obtain (N, e, d). Compute $d' := d^n \mod \phi(N)$. Output $pk = (N, e)$ and $sk = (N, d')$.
- Encaps: on input pk and 1^n, choose a uniform $c_1 \in \mathbb{Z}_N^*$. Then for $i = 1, \ldots, n$ do:
  1. $k_i := \mathrm{lsb}(c_i)$
  2. $c_{i+1} := c_i^e \mod N$

  Output the ciphertext c_{n+1} and the key $k = k_1 \ldots k_n$.
- Decaps: on input $sk = (N, d')$ and ciphertext c, compute $c_1 := c^{d'} \mod N$. For $i = 1, \ldots, n$ do:
  1. $k_i := \mathrm{lsb}(c_i)$
  2. $c_{i+1} := c_i^e \mod N$

  Output $k = k_1 \ldots k_n$.

RSA-based KEM (2)

Theorem (11.35)

If RSA is hard relative to GenRSA, *then Construction 11.34 is a CPA-secure KEM.*

RSA-OAEP

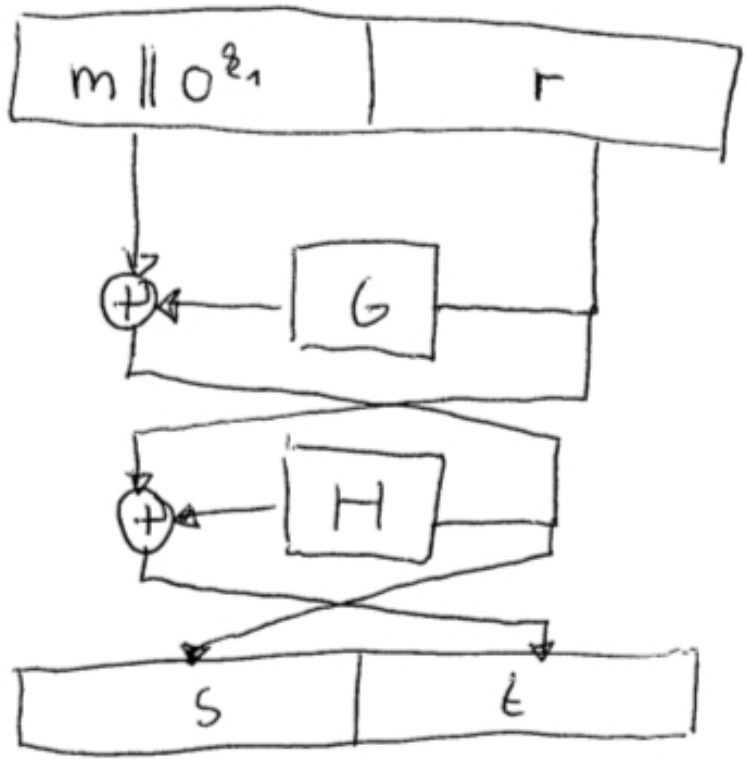
Construction 11.36

- Let ℓ, k_0, k_1 with $k_0, k_1 = \Theta(n)$ such that $\ell + k_0 + k_1$ is less than the minimum bit length of the moduli output by GenRSA(1^n). Let $G : \{0,1\}^{k_0} \to \{0,1\}^{\ell+k_1}$ and $H : \{0,1\}^{\ell+k_1} \to \{0,1\}^{k_0}$.
- Gen: run GenRSA(1^n) to obtain (N, e, d). pk is (N, e) and sk is (N, d).
- Enc: on input (N, e) and $m \in \{0,1\}^\ell$, let $m' := m||0^{k_1}$ and choose a uniform $r \in \{0,1\}^{k_0}$. Then compute

$$s := m' \oplus G(r), \quad t := r \oplus H(s)$$

  and let $\hat{m} := s||t$. Output $c := \hat{m}^e \mod N$:
- Dec: on input (N, d) and c, compute $\hat{m} := c^d \mod N$. If $||m|| > \ell + k_0 + k_1$, output $\perp$. Otherwise, parse $\hat{m} = s||t$ with $s \in \{0,1\}^{\ell+k_1}$ and $t \in \{0,1\}^{k_0}$. Compute $r := H(s) \oplus t$ and $m' := G(r) \oplus s$. If the least-significant k_1 bits of m' are not 0, then output $\perp$. Otherwise, output the ℓ most significant bits.

RSA-OAEP (2)



Theorem

If RSA is hard relative to GenRSA *and* G *and* H *are modelled as random oracles, then RSA-OAEP is CCA-secure.*

CCA-Secure KEM

Construction 11.37:

- Gen: run GenRSA(1^n) to obtain (N, e, d). pk is (N, e) and sk is (N, d). Specify a function $H : \mathbb{Z}_N^* \to \{0,1\}^n$.
- Encaps: Choose uniform $r \in \mathbb{Z}_N^*$. Output ciphertext $c := r^e \mod N$ and key $k := H(r)$.
- Decaps: Given a ciphertext $c \in \mathbb{Z}_N^*$, compute $r := c^d \mod N$ and output $k := H(r)$.

Theorem (11.38)

If RSA is hard relative to GenRSA *and* H *is modelled as a random oracle, then Construction 11.37 is CCA-secure.*

- part of ISO/IEC 18033-2 standard
- less efficient than RSA-OAEP, but proof is easier

Proof: $\mathrm{KEM}^{\mathrm{cca}}_{\mathcal{A},\Pi}(n)$

1. $\mathrm{GenRSA}(1^n)$ is run to obtain (N, e, d). A random $H : \mathbb{Z}^*_N \to \{0,1\}^n$ is chosen.
2. $r \in \mathbb{Z}^*_N$ is chosen uniformly at random. $c := r^e \mod N$ and $k := H(r)$ is computed.
3. $b \in \{0,1\}$ is chosen uniformly at random. If $b = 0$, set $\hat{k} := k$. Otherwise choose $\hat{k} \in \{0,1\}^n$ uniformly at random.
4. $\mathcal{A}$ is given pk, c and $\hat{k}$. $\mathcal{A}$ may query $H(.)$ on any input and $\mathrm{Decaps}_{sk}(.)$ on any input except c.
5. $\mathcal{A}$ outputs $b' \in \{0,1\}$. $\mathcal{A}$ wins if $b = b'$ and looses otherwise.

Proof: Adversary $\mathcal{A}'$

1. Initialize empty lists $\mathrm{L_H}$, $\mathrm{L_{Decaps}}$. Choose a uniform $\mathrm{k} \in \{0,1\}^n$ and store (c, k) in $\mathrm{L_{Decaps}}$.
2. Choose $\mathrm{b} \in \{0,1\}$ uniformly at random. If $\mathrm{b} = 0$, set $\hat{\mathrm{k}} = \mathrm{k}$. Otherwise choose $\hat{\mathrm{k}} \in \{0,1\}^n$ uniformly at random. Simulate $\mathcal{A}$ on pk, c, and $\hat{\mathrm{k}}$.

   When $\mathcal{A}$ queries $\mathrm{H}(\tilde{\mathrm{r}})$:

   - If there is an entry $(\tilde{\mathrm{r}}, \mathrm{k})$ in $\mathrm{L_H}$, then return k.
   - Otherwise, let $\tilde{\mathrm{c}} = \tilde{\mathrm{r}}^e \mod \mathrm{N}$. If there is an entry $(\tilde{\mathrm{c}}, \mathrm{k})$ in $\mathrm{L_{Decaps}}$, then return k and store $(\tilde{\mathrm{r}}, \mathrm{k})$ in $\mathrm{L_H}$.
   - Otherwise, choose $\mathrm{k} \in \{0,1\}^n$ uniformly at random, return k and store $(\tilde{\mathrm{r}}, \mathrm{k})$ in $\mathrm{L_H}$.

Proof: Adversary $\mathcal{A}'$

2. When $\mathcal{A}$ makes a query Decaps$(\tilde{c})$:
   - If there is an entry in L_{Decaps} of the form $(\tilde{c}, k)$, return k.
   - Otherwise, for each entry $(\tilde{r}, k) \in L_H$, check if $\tilde{r}^e = \tilde{c} \mod N$. If yes, output k.
   - Otherwise, choose a uniform $k \in \{0,1\}^n$, return k and store $(\tilde{c}, k)$ in L_{Decaps}.
3. At the end of $\mathcal{A}$'s execution, if there is an entry (r, k) in L_H for which $r^e = c \mod N$, return r.