CS 578 – Cryptography

Prof. Michael Backes

More on the RSA Trapdoor Permutation, Related Encryption Schemes

June 16, 2006

Administrative Announcements

- Investigating your exam:
  - Investigating in today's office hour

Recall: One-way Functions (OWF)

- Definition (One-way function): An efficiently computable function F: {0,1}* → {0,1}* is one-way iff it is hard to invert: given F(x), hard to get x. More precisely:
  $\Pr[F(x) = F(x'); x \leftarrow_R \{0,1\}^n, y := F(x), x' \leftarrow A(n,y)]$
  is negligible.
- Remark: Sometimes families of functions with different domains/ranges and explicit generation algorithm for x

Recall: RSA Trapdoor Permutation

- RSA (Rivest, Shamir, Adleman 1977): Trapdoor Permutation (Permutation that is hard to compute unless one knows a secret trapdoor)
- Definition ((Keyed) trapdoor permutation): A keyed family of efficiently computable functions $F = (F(pk, \cdot))_{(pk,*) \in [Gen(n)]}$, $F(pk, \cdot): \mathcal{M}_{pk} \rightarrow \mathcal{T}_{pk}$ is a family of trapdoor permutations iff
  1. $Pr [F(pk,x) = F(pk,x'); (pk,sk) \leftarrow Gen(n), x \leftarrow_R \mathcal{M}_{pk}, y := F(pk,x), x' \leftarrow A(n,pk,y)]$ is negligible for all efficient A.
  2. There exists an efficient algorithm B such that $B(n, sk, pk, F(pk,x)) = x$ for all $x \in \mathcal{M}_{pk}$.

Recall: Arithmetic Modulo Composites

- RSA: requires arithmetic modulo composites
- $N = p \cdot q$, p and q large primes of same size ($N \approx 1024$ bits)
- For all $a,b \in Z$, one can efficiently compute $x,y \in Z$ such that $a \cdot x + b \cdot y = \gcd(a,b)$.
- Lemma: $x \in Z_N$ is invertible in Z_N if and only if $\gcd(x,N) = 1$.
- Note: If $0 \neq x \notin Z^*_N$, then we can factor N
  → gcd(x,N) is a non-trivial factor of N

Recall: Arithmetic Modulo Composites

- Size of Z^*_N denoted by $\varphi(N)$
- φ called Euler's totient function; easily computable if factorization of N is known:

$$\varphi(N) = N \cdot \prod_i (1 - \frac{1}{p_i}) \qquad if \;\; N = \prod_i p_i^{e_i}$$

- For RSA moduli: $N = p \cdot q$, then $\varphi(N) = (p-1)(q-1) = N - q - p + 1$
- Theorem (Euler, generalization of Fermat's theorem): $\forall\, a \in Z^*_N: a^{\varphi(N)} = 1 \pmod N$

Recall: Chinese Remainder Theorem

- Chinese Remainder Theorem (CRT) for N=pq:
  Let $p \neq q$ be primes and $N = pq$. Then for all $a,b \in Z$:
  $a = b \bmod N \Leftrightarrow a = b \bmod p \wedge a = b \bmod q$
- Works as well if p and q relatively prime
- Further, given $x_p \in Z_p$ and $x_q \in Z_q$ there exists a unique element $s \in Z_N$ such that
  $s = x_p \bmod p$ and $s = x_q \bmod q$.

RSA Trapdoor Permutation

- Setup:
  - Pick random n-bit primes p, q (e.g., n = 512 bits)
  - Set $N = p \cdot q$.
  - Choose arbitrary value e such that $\gcd(e, \varphi(N)) = 1$
- Function: $F: Z_N \rightarrow Z_N$
  - $F(x) = x^e \bmod N$
- Trapdoor for given pk = (e,N):
  $d := e^{-1} \bmod \varphi(N)$

Naïve Use of RSA as an Encryption Scheme

- Naively (wrong):
  - Generate pk = (N,e), sk = d
  - Enc(pk,m) := m^e mod N, Dec(sk,c) := c^d mod N
- Not even secure against passive attacks:
  - Deterministic, Jacobi symbol leaks
- Various successful active attacks:
  - Simple attack in CCA game using two ciphertexts
  - Slightly more complicated attack based on one ciphertext (“blind decryption”)

Improving RSA's performance

- Speeding up RSA encryption by small values of e
  (fast function evaluation → fast encryption)
- Minimal possible value: e = 3
  (2 does not work since: gcd(2, $\varphi(N)$) = 2)
  → be careful: sending the same message encrypted with 3 different public keys then breaks naïve RSA!
- Recommended value: $e=65537=2^{16}+1$
  Runtime for encryption: 17 modular multiplications.
- Runtime of RSA: fast encrypt/slow decrypt
  (for ElGamal: approx. same time for both)

Improving RSA's performance

- Chinese Remainder Theorem to compute separately modulo p and q, i.e., for computing c^d mod N, do:
  - Compute u, v such that up + vq = 1
  - Compute $m_p := c^{d1} \bmod p$ and $m_q := c^{d2} \bmod q$ where $d_1 = d \bmod p-1$ and $d_2 = d \bmod q-1$
  - Set $m := upm_q + vqm_p \bmod N$
- Exponents and modulo shorter by half the size
- Additionally: Two multiplications, one addition
- Gain in speed: roughly factor 4

Improving RSA's performance

- (Bad) Idea: Use small values of d to speed up decryption
- Fact: Cannot make d small and still stay secure !!
  - Wiener '89: Suppose $d < 1/3\ N^{0.25}$. Then given (N,e), it's easy to recover d.
  - Boneh, Durfee, Frankel '98: Suppose $d < N^{0.292}$. Then given (N,e), it's easy to recover d.
  - Open problem for $d < N^{0.5}$

Wiener's Attack (Sketch)

- Recall: $e \cdot d = 1 \pmod{\varphi(N)}$
- $\rightarrow \exists\, k \in Z: e \cdot d = k \cdot \varphi(N) + 1$
- Thus $$\left| \frac{e}{\varphi(N)} - \frac{k}{d} \right| \leq \frac{1}{d\varphi(N)}$$
- $\varphi(N) = N-p-q+1$, hence $|N- \varphi(N)| \leq p+q \leq 3N^{0.5}$
- If $d < 1/3\, N^{0.25}$, the following holds:

$$\left| \frac{e}{N} - \frac{k}{d} \right| \leq \frac{1}{3d^2}$$

Continued Fraction Expansion

- Continued fraction expansion of a fraction of a/b is a tuple $[q_1, \ldots q_m]$ of non-negative integers such that

$$\frac{a}{b} = q_1 + \cfrac{1}{q_2 + \cfrac{1}{q_3 + \cfrac{1}{q_4 + \cfrac{1}{q_5 + \ldots + \cfrac{1}{q_m}}}}}$$

- Continued fraction expansion q of a/b with gcd(a,b)=1 can be efficiently computed using the extended Euclidian algorithm: $q := [q_1, \ldots q_m]$ where the q_i's are the quotients of the extended Euclidian algorithm.

Extended Euclidian Algorithm

- An example for a=53, b=30 :

53		30	(1,0)		(0,1)
23	←-1-	30	(1,-1)	←-1-	(0,1)
23	-1→	7	(1,-1)	-1→	(-1,2)
2	←-3-	7	(4,-7)	←-3-	(-1,2)
2	-3→	1	(4,-7)	-3→	(-13,23)
0	←-2-	1	(30,-53)	←-2-	(-13,23)

- Thus 30/53 = [0,1,1,3,3,2]

Continued Fraction Expansion

- Let $q = [q_1, \ldots q_m]$ be a continued fraction expansion. Then the j-th prefix $[q_1, \ldots q_j]$ of q is called the j-th convergent of q.
- j'th convergent easily computable from j-th convergents for j < i.
- Theorem (Continued Fractions): If gcd(a,b) = gcd(c,d) = 1, and

$$\left| \frac{a}{b} - \frac{c}{d} \right| \leq \frac{1}{2d^2}$$

then c/d is one of the convergents of the continued fraction expansion of a/b.

Wiener's Attack

- Back to Wiener's attack. We have

$$\left| \frac{e}{N} - \frac{k}{d} \right| \leq \frac{1}{3d^2}$$

- Continued Fraction theorem implies that k/d is one of the convergents of e/N (which is publicly known)
- Remark: Continued fraction expansion polynomial in size of N → efficient to test all convergents.
- Final question: Which one to take?

Wiener's Attack

- If x/y is a convergent of e/N, we get two equations if x/y = k/d:
  1. $e \cdot y = x \cdot \varphi(N) + 1$
  2. $N = p \cdot q$
- Two equations in two variables (p,q). Check if solutions over the naturals exist

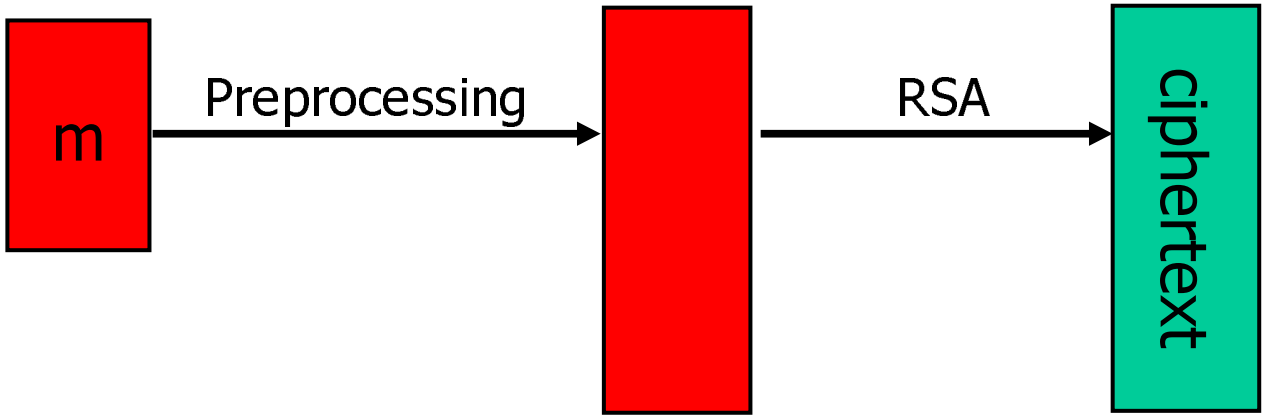
  → Correct solution can be found like this.

Wiener's Attack: Example

- Suppose N = 160523347 and e = 60728973
- Continued fraction expansion of e/N is [0,2,1,1,1,4,12,102,1,1,2,3,2,2,36]
- First convergents are 0, 1/2, 1/3, 2/5, 3/8, 14/37
- For 14/37, we obtain
  1. $60728973 \cdot 37 = 14 \cdot \varphi(N) + 1$
  2. $N = p \cdot q$
- Solving equations yields p = 12347,q = 13001 And indeed $p \cdot q = N$. Thus d = 37 (!)

Towards Good RSA encryption

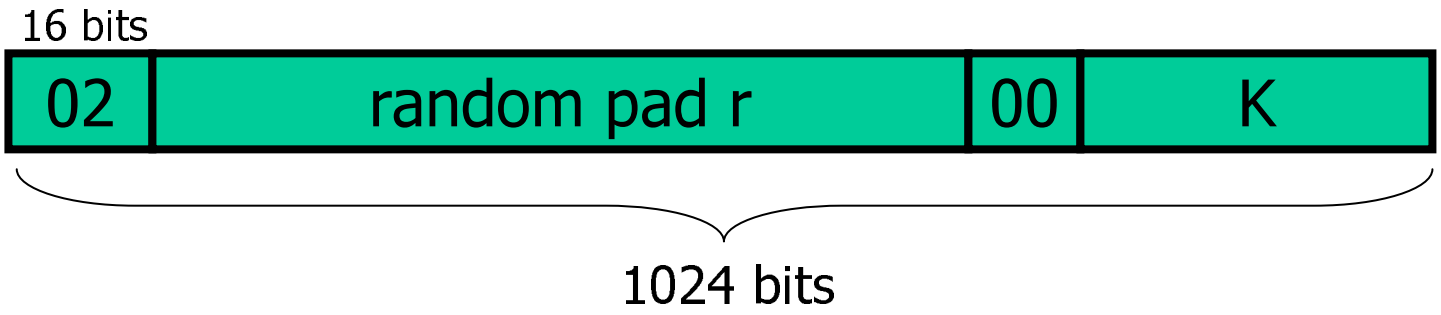
- Naïve RSA insecure
- How to do it correctly:



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

PKCS1 V1.5

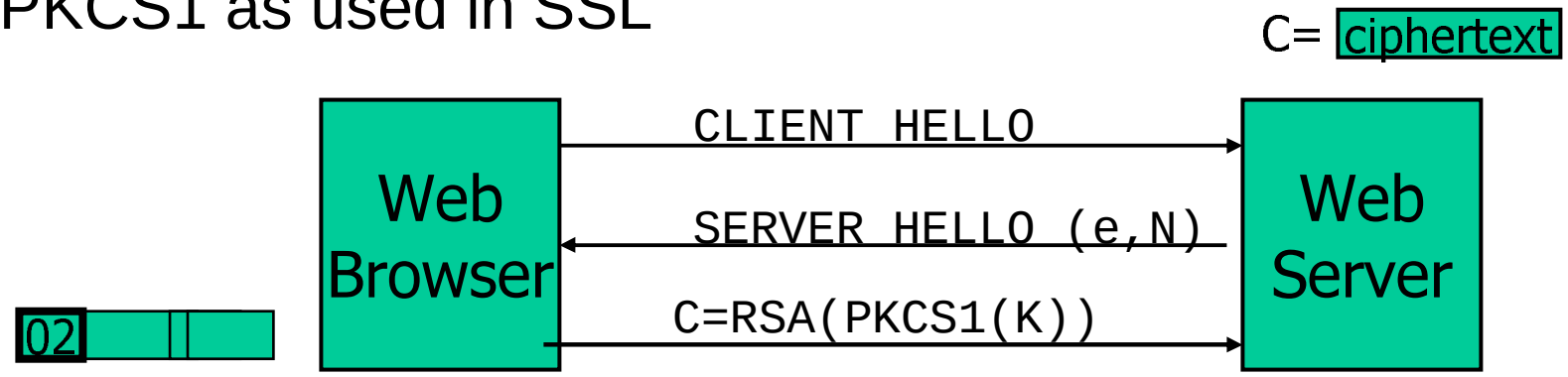
- PKCS1 mode 2 (encryption): RSA Labs, 1991
- Widely used (SSL/TLS, SSH, …)
- To encrypt session key $K \in \{0,1\}^{128}$, pad as follows:



- Then $c = (PKCS1(K))^e \bmod N$
- But only little security analysis

Bleichenbacher Attack on PKCS1 V1.5

- Bleichenbacher Attack 98 (Chosen-ciphertext attack)
- PKCS1 as used in SSL



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

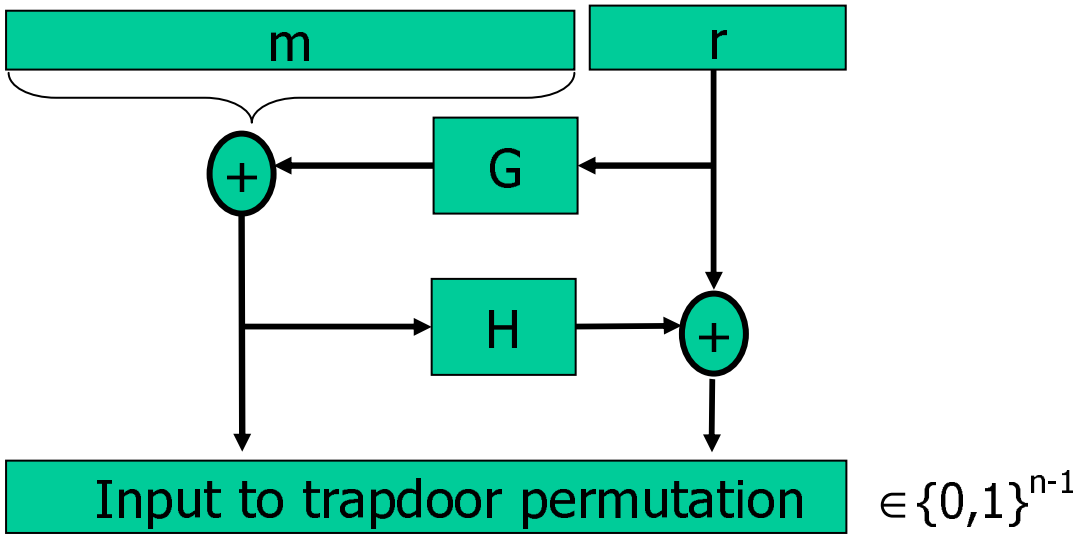
- Attack: to decrypt a given ciphertext c do:
  - Pick random $r \in Z_N$. Compute $c' = r^e \cdot c = (r \cdot m)^e$.
  - Send c' to web server and exploit response.

Bleichenbacher Attack on PKCS1 V.1.5

- Given arbitrary ciphertext c
- Attacker computes $c_i = c \cdot r_i^e \bmod N$ and sends c_i to the browser.
- After about 4 million queries, you learned enough information to decrypt c. (Called the “million message attack”)

PKCS1 V2.0 - OAEP

- New PKCS1: V2.0, 1999
- New preprocessing function: OAEP by Bellare, Rogaway 94.

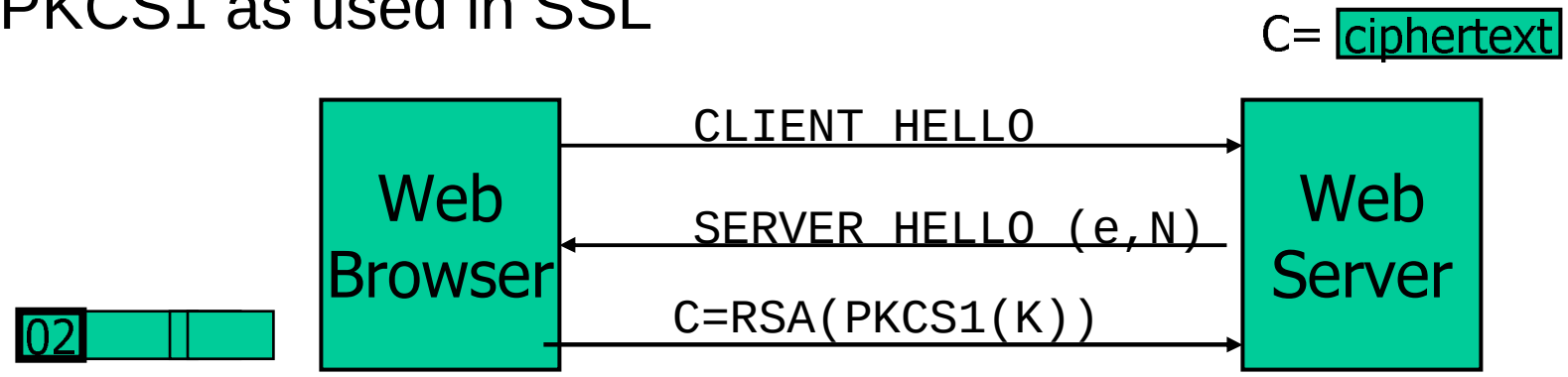


PKCS1 V2.0 - OAEP

- Theorem (Fujisaki, Okamoto, Pointcheval, Stern, 2001): If RSA is a trapdoor permutation, then RSA-OAEP is CCA2-secure provided that G,H are "*random oracles*".
- In practice: use SHA-1 or MD5 for G and H.

Bleichenbacher Attack on PKCS1 V.1.5

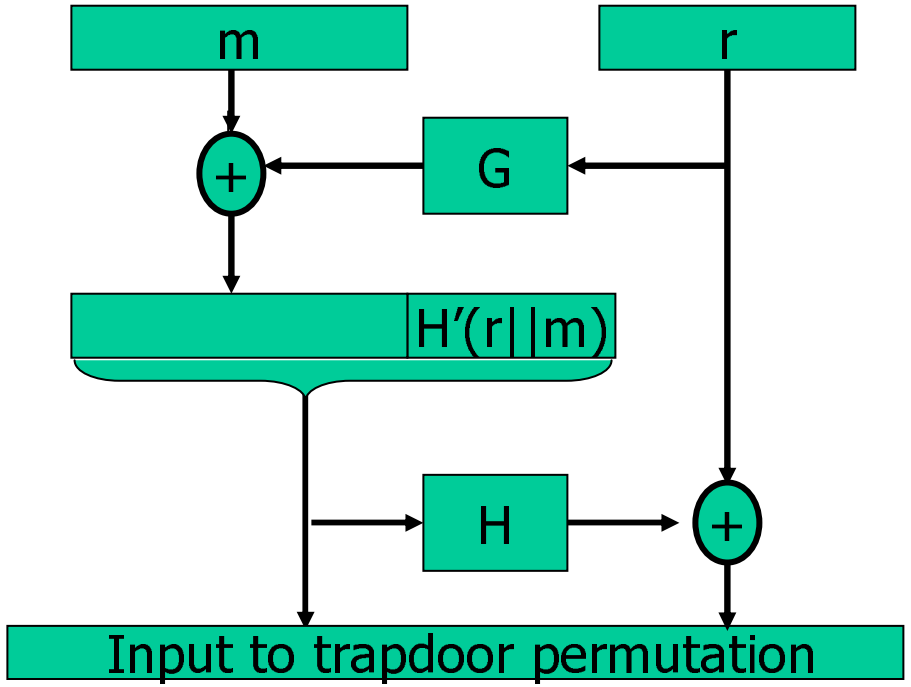
- Bleichenbacher Attack 98 (Chosen-ciphertext attack.)
- PKCS1 as used in SSL



- $\Rightarrow$ attacker can test if 16 MSBs of plaintext = '02'.
- Attack: to decrypt a given ciphertext C do:
  - Pick random $r \in Z_N$. Compute $C' = r^e \cdot C = (rM)^e$.
  - Send C' to web server and use response.

OAEP Improvements

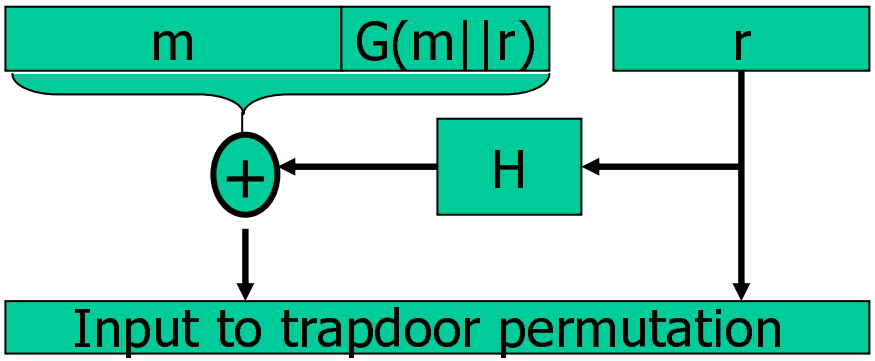
- OAEP+ (Shoup, 2001)



- Theorem: For all trapdoor permutations F, F-OAEP+ is CCA2-secure if G,H',H are "*random oracles*".

OAEP Improvements (cont'd)

- SAEP+ (Boneh, 2001)



- Theorem: If RSA is a trapdoor permutation, then RSA-SAEP+ is CCA2-secure provided that G,H are "*random oracles*".

On (in-)secure Implementations

```
decrypt(C)  {
    error = 0;
    ........
    if  ( RSA^-1(C) > 2^(n-1) )
        { error =1;  goto exit; }
    ........
}   if  ( pad(RSA^-1(C)) “not correct” )
        { error = 1;  goto exit; }
```

- Problem: If timing information leaks type of error
  → Attacker can decrypt any ciphertext c

Implementation Attacks

- Attack the implementation of RSA
- Timing attack: (Kocher 97)
  The time it takes to compute c^d mod N can expose d.
- Power attack: (Kocher 99)
  The power consumption of a smartcard while it is computing c^d mod N can expose d.
- Faults attack: (BDL 97)
  A computer error during c^d mod N can expose d.

Recommended RSA Key Lengths

- Size of the composite should be chosen to match the security of the symmetric key system
- NIST recommendation (similar to ElGamal, and again people ignore this)

Symmetric key length	Public key length
64 bits	512 bits
80 bits	1024 bits
128 bits	3072 bits
256 bits	15360 bits

Summary of Public-key Encryption

- Need cryptographic assumptions: Usually number-theoretic: DLog, CDH, DDH, Factoring, RSA, Quadratic residuosity, etc.
- ElGamal: Public-key encryption scheme strongly related to Diffie-Hellman key exchange, CPA-secure in G_q, vulnerable to active attacks
- Cramer-Shoup encryption: CCA-2 secure encryption scheme, extension of ElGamal
- RSA: Trapdoor one-way permutation, extensively used for encryption and signatures, naïve use as encryption insecure
- OAEP, OAEP+, SAEP+: CCA2-secure encryption schemes in the random oracle model, rely on trapdoor permutation/RSA
- Next: Digital signatures