Cryptology complementary

✧

RSA encryption & signatures

Pierre Karpman

pierre.karpman@univ-grenoble-alpes.fr

https://www-ljk.imag.fr/membres/Pierre.Karpman/tea.html

2018–04–12

An objective: asymmetric/public-key cryptography

Asymmetric/public-key encryption:

- One has encryption and decryption functions Enc, Dec
- One has key pairs (pk, sk) s.t. $\mathrm{Dec}_{sk} \circ \mathrm{Enc}_{pk}$ is the identity function
- The key *pk* can be announced "publicly", for everyone to encrypt
- The key *sk* must be kept secret
- It should be hard to
  - find *sk* from *pk*
  - decrypt w/o *sk*
  - learn information about encrypted messages
  - etc.

$\rightsquigarrow$ A good asymmetric encryption scheme meets IND-CCA security

IND-CCA for Public-Key encryption

IND-CCA for $(\mathrm{Enc}, \mathrm{Dec})$: An adversary cannot distinguish $\mathrm{Enc}(pk_C, 0)$ from $\mathrm{Enc}(pk_C, 1)$, when given (restricted) oracle access to $\mathrm{Dec}(sk_C, \cdot)$ oracle:

1. The Challenger chooses a key pair (pk_C, sk_C), a random bit b, sends $c = \mathrm{Enc}(pk_C, b)$, pk_C to the Adversary
2. The Adversary may repeatedly submit queries $x_i \neq c$ to the Challenger
3. The Challenger answers a query with $\mathrm{Dec}(sk_C, x_i) \in \{0, 1, \perp\}$
   - This assumes w.l.o.g. that the domain of Enc is $\{0, 1\}$, and that decryption may fail
4. The Adversary tries to guess b

Public-key signatures

Public-key signatures

- On has signing and verifying algorithms Sig, $\mathrm{Ver} : x \mapsto \{\top, \bot\}$
- One has signing and verifying keys sk, pk, s.t. $\mathrm{Ver}_{pk} \circ \mathrm{Sig}_{sk}$ is the constant function $\top$
- The key pk can be announced "publicly", for everyone to verify signatures
- The key sk must be kept secret
- It should be hard to
  - find sk from pk
  - find valid signatures w/o sk
  - etc.

$\rightsquigarrow$ A good public-key signature scheme meets EUF-CMA security

EUF-CMA for Public-Key signatures

EUF-CMA for $(\mathrm{Sig}, \mathrm{Ver})$: An adversary cannot forge a valid signature σ for a message m such that $\mathrm{Ver}(pk_C, \sigma, m)$ succeeds, when given (restricted) oracle access to $\mathrm{Sig}(sk_C, \cdot)$:

1. The Challenger chooses a pair (pk_C, sk_C) and sends pk_C to the Adversary
2. The Adversary may repeatedly submit queries m_i to the Challenger
3. The Challenger answers a query with $\sigma_i = \mathrm{Sig}(sk_C, m_i)$
4. The Adversary tries to forge a signature σ_f for a message $m_f \neq_i m_i$, s.t. $\mathrm{Ver}(pk_C, \sigma_f, m_f) = \top$

The RSA permutation

RSA (Rivest, Shamir, Adleman, 1977) in a nutshell: a family of "one-way permutations with trapdoor"

- Publicly define $\mathcal{P}$ that everyone can compute
- Knowing $\mathcal{P}$, it is "hard" to compute $\mathcal{P}^{-1}$ (even on a single point)
- There is a *trapdoor* associated w/ $\mathcal{P}$
- Knowing the trapdoor, it is easy to compute $\mathcal{P}^{-1}$ everywhere

RSA: how?

- Let p, q be two (large) prime numbers
- Let $N = pq$
- Any $0 < x < N$ s.t. $\gcd(x, N) = 1$ is invertible in $\mathbb{Z}/N\mathbb{Z}$
    - Note that knowing $x \notin (\mathbb{Z}/N\mathbb{Z})^\times \Leftrightarrow$ knowing p and q
    - Why?

Proposition: order of $(\mathbb{Z}/N\mathbb{Z})^\times$

Let N be as above, the order of the multiplicative group $(\mathbb{Z}/N\mathbb{Z})^\times$ is equal to $(p-1)(q-1)$. (More generally, it is equal to $\varphi(N)$)

- So for any $x \in (\mathbb{Z}/N\mathbb{Z})^\times$, $x^{k\varphi(N)+1} = x$

RSA: more on how

- Let e be s.t. $\gcd(e, \varphi(N)) = 1$; consider $\mathcal{P} : x \mapsto x^e \mod N$
- $\mathcal{P}$ is a permutation over $(\mathbb{Z}/N\mathbb{Z})^\times$ (in fact over the entire $\mathbb{Z}/N\mathbb{Z}$)
- Knowing e, N, it is easy to compute $\mathcal{P}$
- Knowing e, $\varphi(N)$, it is easy to compute d s.t. $ed = 1 \mod \varphi(N)$
- Knowing d, x^e, it is easy to compute $x = x^{ed}$

$\Rightarrow$ We have a permutation with trapdoor, but how good is the latter?

RSA: how secure?

Knowing $ed = k\varphi(N) + 1$, it is easy to find $\varphi(N)$ (admitted)

Knowing $N = pq$, $\varphi(N) = (p-1)(q-1)$, it is easy to find p and q

- $\varphi(N) = pq - (p+q) + 1$; $p + q = -(\varphi(N) - N - 1)$
- For any a, b, knowing ab and $a+b$ allows to find a and b
  - Consider the polynomial $(X-a)(X-b) = X^2 - (a+b)X + ab$
  - $\Delta = (a+b)^2 - 4ab = (a-b)^2$
  - $a = ((a+b) + (a-b))/2$

$\Rightarrow$ Knowing, N, e, d, it is easy to factor N, plus:

- e does (basically) not depend on N

$\Rightarrow$ If it is easy to compute d from N, e, it is easy to factor N, and

- It is a hard problem to factor $N = pq$ when p, q are large random primes

BUT it might not be necessary to know d to (efficiently) invert $\mathcal{P}$‽

To sum up: the RSA permutation family

- Let $N = pq$, with p, q prime numbers
- Let e be s.t. $\gcd(e, \varphi(N) = (p-1)(q-1)) = 1$
  - In practice, e is often fixed to 3 or 65537
- The RSA permutation $\mathcal{P}$ over $\mathbb{Z}/N\mathbb{Z}$ is given by $m \mapsto m^e$
- The inverse $\mathcal{P}^{-1}$ is given by $m \mapsto m^d$, where $ed \equiv 1 \mod \varphi(N)$
- N, e are the *public parameters* defining $\mathcal{P}$
- N, e, d are the *private parameters* defining $\mathcal{P}$, $\mathcal{P}^{-1}$

Assumption: Given only the public parameters, it is "hard" to invert $\mathcal{P}$

RSA for PKC

The objective: use RSA to build

- Public-key (asymmetric) encryption
  - Can then be used for asymmetric key exchange
- Public-key signatures

These schemes will need to satisfy the usual security notions

- For encryption: IND-CPA/CCA ("semantic security")
- For signatures: Existential unforgeability under chosen-message attacks (EUF-CMA)

RSA Encryption: first attempt

Let $\mathcal{P}, \mathcal{P}^{-1}$ be RSA permutations with parameters N, e, d. Define:

- $\mathrm{Enc}(pk = (N, e), m) = \mathcal{P}(m) = (m^e \mod N)$
- $\mathrm{Dec}(sk = (N, e, d), c) = \mathcal{P}^{-1}(c) = (c^d \mod N)$

Not randomized $\Rightarrow$ fails miserably, not IND-CCA

- When receiving $c = \mathcal{P}(b)$, the Adversary compares with $c_0 = \mathcal{P}(0)$, $c_1 = \mathcal{P}(1)$

More issues with raw RSA

- If m, e are small, it may be that $m^e \mod N = m^e$ (over the integers) $\Rightarrow$ trivial to invert
  - Example: N is of 2048 bits, $e = 3$, m is a one-bit challenge: adding 512 random bits of padding before encrypting does not provide IND-CCA security!
- Consider a *broadcast* setting where m is encrypted as $c_i = m^3 \mod N_i$, $i \in [1, 3]$. Suppose that $\forall i$, $m < N_i < m^3$. Using the CRT, one can reconstruct $m^3 \mod N_1 N_2 N_3 = m^3$ and retrieve m.
  - Even random padding might not prevent this attack, if too structured (Hastad, Coppersmith)

More issues with (semi-)raw RSA

A very useful result for analysing the security of RSA is due to Coppersmith (1996):

Finding small modular roots of univariate polynomials

Let P be a polynomial of degree k defined modulo N, then there is an efficient algorithm that computes its roots that are less than $N^{1/k}$

- The complexity of the algorithm is polynomial in k (but w. a high degree)
- Example application: if $c = (2^k B + x)^3 \mod N$ is an RSA image, B is known and of size $2/3 \log(N)$, one can find x of size $k < 1/3 \log(N)$ by solving $(2^k B + k)^3 - c = 0$
- Other applications: in the previous slide; in slide #19, ...

Proper RSA-ENC

Let $\mathcal{P}, \mathcal{P}^{-1}$ be RSA permutations with parameters N, e, d. Let Pad, Pad^{-1} be a padding function and its inverse. Define:

- $\text{Enc}(pk = (N, e), m) = \mathcal{P}(\text{Pad}(m)) = (\text{Pad}(m)^e \mod N)$
- $\text{Dec}(sk = (N, e, d), c) = \text{Pad}^{-1}(\mathcal{P}^{-1}(c)) = \text{Pad}^{-1}(c^d \mod N)$

Necessary conditions on Pad:

- It must be invertible
- It must be randomized (with a large-enough number of bits)
- For all m, N, e, $\text{Pad}(m)^e$ must be larger than N

OAEP: A good padding function for RSA-ENC

OAEP: Optimal Asymmetric Encryption Padding (Bellare & Rogaway, 1994):

- Let $k = \lfloor \log(N) \rfloor$, κ be a security parameter
- Let $\mathcal{G} : \{0,1\}^\kappa \to \{0,1\}^n$, $\mathcal{H} : \{0,1\}^n \to \{0,1\}^\kappa$ be two hash functions
- Define $\mathrm{Pad}(x)$ as $(y_L \| y_R) = x \oplus \mathcal{G}(r) \| r \oplus \mathcal{H}(x \oplus \mathcal{G}(r))$, where $r \xleftarrow{\$} \{0,1\}^\kappa$
- One has $x = \mathrm{Pad}^{-1}(y_L \| y_R) = y_L \oplus \mathcal{G}(y_R \oplus \mathcal{H}(y_L))$

More on OAEP

- OAEP essentially uses a two-round Feistel structure
- To be instantiated, it requires two hash functions $\mathcal{H}$ and $\mathcal{G}$ with variable output size
- A possibility is to use a single XOF $\mathcal{X} : \{0,1\}^* \rightarrow \{0,1\}^*$, such as SHAKE-128

OAEP: Why does it work (kind of)?

Intuitively, full knowledge of $(y_L||y_R)$ is necessary to invert:

- If part of y_L is unknown, $\mathcal{H}(y_L)$, then $\mathcal{G}(y_R \oplus \mathcal{H}(y_L)$ are uniformly random
- If part of y_R is unknown, $\mathcal{G}(y_R \oplus \mathcal{H}(y_L))$ is uniformly random
- In both cases $\Rightarrow$ x is hidden by a "one-time-pad"

More formally, we would like a reduction of the form:

Breaking RSA-OAEP w. Adv. $\epsilon \Rightarrow$ Inverting RSA w. Adv. $\approx \epsilon$

OAEP woes

- The original proof that OWP-OAEP is IND-CCA (for any good OWP) (Bellare & Rogaway, 1994) was incorrect
- Shoup showed that there can be no such proof (2001)
- But when OWP is RSA, then there *is* a proof (Shoup, 2001; Fujisaki & al., 2000)!
    - Exploits Coppersmith's algorithm!
- Not all the proofs are *tight* (e.g. Adv. $\epsilon \Rightarrow$ Adv. ϵ^2)
    - Need large parameters to give a meaningful guarantee

What about RSA-SIG now?

Let $\mathcal{P}, \mathcal{P}^{-1}$ be RSA permutations with parameters N, e, d. Define:

- $\text{Sig}(sk = (N, e, d), m) = \mathcal{P}^{-1}(m)$
- $\text{Ver}(pk = (N, e), \sigma, m) = \mathcal{P}(\sigma)$ `==` m `?` $\top$ `:` $\bot$

Why this might work:

- Correctness: $(m^d)^e \equiv m \mod N$ $(\mathcal{P}^{-1} \circ \mathcal{P} = \mathcal{P} \circ \mathcal{P}^{-1} = \text{Id})$
- Security: Comes from the hardness of inverting $\mathcal{P}$ w/o knowing d $\rightsquigarrow$ forging a signature for $m \Leftarrow$ compute $\mathcal{P}^{-1}(m)$

Raw RSA-SIG: That's no good!

- If $m \equiv m' \mod N$, then $\mathcal{P}^{-1}(m) = \mathcal{P}^{-1}(m) \Rightarrow$ trivial forgeries
- $\mathcal{P}^{-1}(m)\,\mathcal{P}^{-1}(m') = (m^d)(m'^d) \mod N = (mm')^d$
  $\mod N = \mathcal{P}^{-1}(mm') \Rightarrow$ trivial forgeries over $[0, N-1]$

Again, some padding is necessary!

Proper RSA-SIG

Let $\mathcal{P}, \mathcal{P}^{-1}$ be RSA permutations with parameters N, e, d. Let Pad be a padding function. Define:

- $\mathrm{Sig}(sk = (N, e, d), m) = \mathcal{P}^{-1}(\mathrm{Pad}(m))$
- $\mathrm{Ver}(pk = (N, e), \sigma, m) = \mathcal{P}(\sigma)$ `==` $\mathrm{Pad}(m)$ `? ⊤ : ⊥`

- Pad does not need to be invertible
- It does not need to be randomized (tho this can help)

What padding functions for RSA-SIG?

Let $k = \lfloor \log(N) \rfloor$

Full-Domain Hash (FDH) (Bellare & Rogaway; 1993):

- Let $\mathcal{H} : \{0,1\}^* \to \{0,1\}^k$ be a hash function, $\mathrm{Pad}(m) = \mathcal{H}(m)$

PFDH (Coron, 2002):

- Let $\mathcal{H} : \{0,1\}^* \to \{0,1\}^k$ be a hash function, $r \xleftarrow{\$} \{0,1\}^n$, $\mathrm{Pad}(m) = \mathcal{H}(m||r)$
  - r is not included in the padding *per se*, but must be transmitted along

- Both are pretty simple, both provable in the random oracle model (ROM)
- The proof is *tighter* for PFDH ("good" security is obtained for smaller N)
- $\mathcal{H}$ can instantiated by a XOF

Another nice padding: PSS-R

PSS-R (Bellare & Rogaway, 1996):

- Let $\lfloor \log(N) \rfloor = k = k_0 + k_1 + k_2$, $\mathcal{H} : \{0,1\}^{k-k_1} \to \{0,1\}^{k_1}$, $\mathcal{G} : \{0,1\}^{k_1} \to \{0,1\}^{k-k_1}$ be two hash functions, $r \xleftarrow{\$} \{0,1\}^{k_0}$
- $\mathsf{Pad} : \{0,1\}^{k_2} \to \{0,1\}^{k}$ is defined by $\mathsf{Pad}(x) = \mathcal{H}(x\|r)\|(x\|r \oplus \mathcal{G}(\mathcal{H}(x\|r)))$
- If $|x| < k_2$, PSS-R is invertible (then, the message m does not need to be transmitted with the signature)
- Otherwise, e.g. compute $\mathsf{Pad}(x')$ where $x' = \mathcal{I}(x)$, $\mathcal{I} : \{0,1\}^{*} \to \{0,1\}^{k_2}$ a hash function (then, k_2 must be "large enough")

More on PSS-R

- In fact, PSS-R may also be used as padding for RSA-ENC (Coron & al., 2002)!
  - Notice the relative similarity between PSS-R and OAEP
- Both SIG and ENC cases are provably secure in the ROM
  - In the specific case of RSA, same as OAEP

RSA-SIG: Quick implementation comments

- The signer knows N, e, d, and also the factorization $p \times q$ of N
- Thanks to the CRT, any computation mod N (in particular $m \mapsto m^d$ may be done mod p and mod q
- A CRT implementation is more efficient, as multiplying two numbers does not have a linear cost
- In fact, such CRT decomposition is a useful approach for general big number arithmetic
- $\Rightarrow$ "RSA-CRT" implementations
  - More efficient, but beware of fault attacks! (That's a general warning, tho)