Homework 3 for Cryptology 1

Christian Majenz & Carsten Baum, DTU

March 24, 2023

1 Paillier Encryption - 8 points

We now look at a public-key encryption scheme that, like RSA, is secure if factoring large numbers is difficult. In comparison, we will see that this new scheme is IND-CPA secure by construction such that we won't need the OAEP transform! The encryption scheme is called Paillier encryption [Pai99], although we will look at a version due to Damgård and Jurik [DJ01].

Paillier allows to encrypt any message in Z_N, where N is a biprime like in RSA. The encryption scheme is built using two homomorphisms from Z_N into $Z^*_{N^2}$, i.e. the set of numbers that are coprime to N^2. Intuitively, the first homomorphism takes care of the message while the second integrates the randomness and makes sure that the scheme is IND-CPA secure.

The first homomorphism from Z_N into $Z^*_{N^2}$ is defined as follows:

$$\alpha(x) = (1 + xN) \bmod N^2$$

Exercise 1.

Verify that the mapping is homomorphic i.e.,

$$\alpha(x) \cdot \alpha(y) \bmod N^2 = \alpha(x + y \bmod N)$$

and therefore also

$$\alpha(x)^y \bmod N^2 = \alpha(x \cdot y \bmod N)$$

Unfortunately, only applying α to the message x does not lead to an encryption of x. Instead, it is trivially invertible.

Exercise 2.

Find the inverse mapping $\alpha^{-1}(\cdot)$ such that, for all elements of $Z^*_{N^2}$ in the image of α it is possible to compute the preimage x. In other words, how do you compute $x \in Z_N$ from $(1 + xN) \bmod N^2$?

The second homomorphism β maps elements from Z^*_N into $Z^*_{N^2}$, and is essentially the same as RSA encryption, but modulo N^2, where the public exponent e is fixed to N.

$$\beta(r) = r^N \bmod N^2$$

❓ Exercise 3.

Verify that the mapping is multiplicatively homomorphic i.e., $\beta(x) \cdot \beta(y) = \beta(x \cdot y) \bmod N^2$

The security of Paillier encryption is based on the assumption that it is hard to distinguish between uniformly random elements in $Z^*_{N^2}$ and the output of $\beta(r)$ on a uniformly random r.

On the other hand it is very easy to distinguish $\beta(r)$ from random elements in $Z^*_{N^2}$ if one knows the factorization of N: from the factorization of N one can compute $\phi(N) = (p-1)(q-1)$ and then one can test if a given y is in the image of β by checking if

$$y^{\phi(N)} = 1 \bmod N^2$$

❓ Exercise 4.

Show that if $y = \beta(r)$, then the aforementioned identity holds. Then show that if $y = \alpha(x)\beta(r)$ for $x \in Z_N, x \neq 0$, then $y^{\phi(N)} \neq 1 \bmod N^2$ if $r \in Z^*_N$. To show the second statement, look at the expansion of $\alpha(x)^2, \alpha(x)^3, \ldots \bmod N^2$ and check under which conditions this could be 1.

Now we are ready to describe the Paillier cryptosystem:

Key Generation: Sample primes p, q of the same length and compute $N = p \cdot q$. Using the Chinese Remainder Theorem, let $sk \in Z_{N \cdot \phi(N)}$ such that

$$sk = \begin{cases} 0 \bmod \phi(N) \\ 1 \bmod N \end{cases}$$

Output sk and $pk = N$.

Encryption: On input a message $m \in Z_N$, sample a random $r \in Z^*_N$, and output

$$C = \alpha(m) \cdot \beta(r) = (1 + mN)r^N \bmod N^2$$

Decryption: Compute $m = \alpha^{-1}(C^{sk} \bmod N^2) \bmod N$

❓ Exercise 5.

Use the Chinese Remainder Theorem to write down a formula for the secret key, sk. Using this, check that decryption works correctly.

The security of Paillier encryption follows immediately from the assumption that $\beta(r)$ is indistinguishable from a random element of $Z^*_{N^2}$.

❓ Exercise 6. (Bonus exercise - not mandatory!)

Assume that there exists an algorithm A that can efficiently break the IND-CPA security of the Paillier cryptosystem with advantage τ. Then construct an algorithm B that, on input a biprime N and an element $y \in Z^*_{N^2}$, can decide if $y = \beta(r)$ for $r \in Z^*_N$ or not with advantage approximately $\tau/2$.

❓ Exercise 7. (Paillier is not IND-CCA secure)

Show that Paillier is not IND-CCA secure. To do so, construct an attack that exploits the homomorphic properties of ciphertexts!

2 Merkle Trees - 2 points

In this exercise, you will learn a different technique to construct a collision-resistant hash function $G : \{0,1\}^* \to \{0,1\}^n$ given a collision-resistant hash function $H : \{0,1\}^{2n} \to \{0,1\}^n$.

We first construct the hash function $G' : \{0,1\}^{4n} \to \{0,1\}^n$. On input $x = (x_1|x_2|x_3|x_4)$ where $x_i \in \{0,1\}^n$ and $|$ denotes string concatenation, it outputs $H(H(x_1|x_2)|H(x_3|x_4))$.

❓ Exercise 8. (Expansion)

Show that if H is collision-resistant, then so is G'! To do so, consider what happens if you have another input $x' = (x'_1|x'_2|x'_3|x'_4)$ such that $G'(x') = G'(x)$. What can you say about $H(x_1|x_2)$ vs. $H(x'_1|x'_2)$ and $H(x_3|x_4)$ vs. $H(x'_3|x'_4)$?

Now this looks like some construction that can be applied recursively, and indeed this is how one obtains the full Merkle tree construction.

❓ Exercise 9.

Show how, by recursing, you can construct a collision-resistant hash function G for inputs of length $n \cdot 2^k$ for arbitrary $k \in \mathbb{N}, k > 1$.

Now a similar result could have been obtained from the Merkle-Damgård construction. Hence we should consider their differences.

❓ Exercise 10.

Assume that you are implementing Merkle-Damgård and Merkle tree hashing for an input of length $n2^k$ on a machine that allows you to compute multiple calls to the underlying compression function H in parallel. What can you say about the runtime of a program for Merkle-Damgård vs. a program for Merkle tree hashing?

What you should do

- Write the solutions to the exercises in one document.
- Upload your document via the “Assignments” link (DK: “Opgaver”) on Inside.
- Deadline: see course page on DTU Learn.
- You may work in groups of at most three students.
- The format of your document should be PDF.

- If you use program code of any kind, please include it **and** describe your solution to that it can be understood without looking at the code.

References

[DJ01] Ivan Damgård and Mats Jurik. A generalisation, a simplification and some applications of Paillier's probabilistic public-key system. In Kwangjo Kim, editor, *PKC 2001*, volume 1992 of *LNCS*, pages 119–136. Springer, Heidelberg, February 2001.

[Pai99] Pascal Paillier. Public-key cryptosystems based on composite degree residuosity classes. In Jacques Stern, editor, *EUROCRYPT'99*, volume 1592 of *LNCS*, pages 223–238. Springer, Heidelberg, May 1999.