Technical Note on Haskell Cryptography

This document provides deeper explanations of the mathematical aspects of the theoretical results that we introduce in the specifications of the Haskell Cryptography lab. In Section 1, we give a proof of the correctness of RSA based on Fermat's little theorem and we present varied applications of RSA. We then discuss the complexity and security of RSA encryption in Section 2. Finally, we quickly prove the soundness of the modes of operations that we mention in this lab.

1 RSA Correctness

There exist different proofs of the RSA correctness. The one we will present here is based on Fermat's little theorem. A beautiful higher order variant relies on Euler's theorem[1] which generalises Fermat's little theorem and involves some notions of group theory.

1.1 Choice of the public exponent e

During the key generation, the choice of e ensures that step 4 is feasible, i.e. that e does have an inverse d modulo $(p-1)(q-1)$.

Proof. As a matter of fact, if we let $\phi = (p-1)(q-1)$, by virtue of Bézout's theorem, we have:

$$\begin{aligned} gcd(e,\phi) = 1 \quad &\implies \quad (\exists (u,v) \in \mathbb{Z}^2)(eu + \phi v = 1) \\ &\implies \quad (\exists (u,v) \in \mathbb{Z}^2)(eu = 1 - \phi v) \\ &\implies \quad (\exists u \in \mathbb{Z})(eu = 1 \mod \phi) \end{aligned}$$

□

1.2 Fermat's Little Theorem

In order to prove the RSA correctness, we will need Fermat's little theorem which states that for any integer a and any prime p, we have:

$$a^p \equiv a \mod p$$

Proof. In order to make this proof self contained, we will also present a concise proof of Fermat's little theorem based on the binomial expansion. We will use a proof by induction on a for any prime p, and

[1] which is further generalised by Carmichael's theorem

define the following hypothesis for $a \in \mathbb{N}$:

$$(H_a) : a^p \equiv a \mod p$$

The hypothesis is trivial for $a = 0$. Let us now assume that it is verified for a given $a \in \mathbb{N}$. By virtue of the binomial expansion, we have:

$$(a+1)^p = \sum_{k=0}^{p} \binom{p}{k} a^k = 1 + \sum_{k=1}^{p-1} \binom{p}{k} a^k + a^p$$

where $\binom{p}{k} = \frac{p!}{k!(p-k)!}$ refers to the binomial coefficients.

However, for all k in $[\![1, p-1]\!]$, no divisor of p appears in the denominator of $\binom{p}{k}$ since p *is prime.* Thus, $\binom{p}{k}$ is a multiple of p and so:

$$(a+1)^p \equiv 1 + a^p \mod p$$

which gives us, using the induction hypothesis:

$$(a+1)^p \equiv a + 1 \mod p$$

□

We can add that if a and p are coprime, then:

$$a^{p-1} \equiv 1 \mod p$$

1.3 RSA Encryption Scheme

Let us prove that for a given key pair $((e, N), (d, N))$ and message x, we have $decryptRSA_{(d,N)}(encryptRSA_{(e,N)}(x)) = x$, or in other words:

$$x^{ed} \equiv x \mod N$$

Proof. Case 1: Let us first study the case where x and N are coprime. Then x and p are coprime and Fermat's little theorem gives us $x^{p-1} \equiv 1 \mod p$ from which we have:

$$x^{(p-1)(q-1)} \equiv 1 \mod p$$

A similar reasoning with q gives us:

$$x^{(p-1)(q-1)} \equiv 1 \mod q$$

As p and q are distinct primes, they are in particular coprime, and thus the Chinese remainder theorem

gives us:

$$x^{(p-1)(q-1)} \equiv 1 \mod pq$$

But now, step 4 of the key generation ensures that there exists an integer k such that $ed = 1 + k(p-1)(q-1)$, so:

$$\begin{aligned} x^{ed} &= x^{1+k(p-1)(q-1)} && \mod N \\ &= x.\left(x^{(p-1)(q-1)}\right)^k && \mod N \\ &= x && \mod N \end{aligned}$$

Case 2: Let us now study the case where x and N are not coprime, and let us assume without loss of generality that $gcd(x, N) = p$. Using again the Chinese remainder theorem, it suffices to show the following:

$$\begin{cases} x^{ed} \equiv x \mod p \\ x^{ed} \equiv x \mod q \end{cases}$$

The first equation $x^{ed} \equiv x \mod p$ is trivial as x is a multiple of p. Moreover, we now know that x and q are coprime, so Fermat's little theorem applies and ensures that:

$$x^{q-1} \equiv 1 \mod q$$

Thus:

$$\begin{aligned} x^{ed} &\equiv x^{1+k(p-1)(q-1)} \mod q \\ &\equiv x.\left(x^{q-1}\right)^{k(p-1)} \mod q \\ &\equiv x \mod q \end{aligned}$$

□

1.4 Discussion

The main advantage of public key algorithms like the RSA is that the parties do not need to agree on a shared private key before the execution of the protocol in order to exchange some data. The main flaw however, is that the size of the data being exchanged is bounded by the size of the RSA modulus $N = pq$[2]. Even though we could split the data into chunks that we could send separately, the RSA remains relatively slow because of the exponentiation that is required by both the encryption and the decryption.

In *hybrid encryption*, we take advantage of the efficiency of a symmetric key algorithm to encrypt a data with a private key of fixed size[3], that we send to the receiver using a public key cryptosystem like RSA[4].

The RSA is also used in *digital signatures* where a sender wants to sign his message in order to prove to

[2]The RSA modulus is generally at least 1024 bits long.
[3]The key is 256 bits long for example in the protocol AES-256.
[4]Other cryptographic protocols have specifically been designed for key exchange such as the Diffie-Hellman Key Exchange.

a receiver that he is the actual sender of the message. Let us consider two parties Ann and Bob with two asymmetric encryption schemes $(pub_A, priv_A)$ and $(pub_B, priv_B)$ respectively. When Ann wants to send a message x to Bob, she sends him $pub_B(x)$. Intuitively, a digital signature can be achieved if she also sends him $pub_B(priv_A(x))$ so that Bob can recover both x and $priv_A(x)$. The latter message could only be calculated and sent by A and thus constitutes a signature. In practice and for efficiency reasons, Ann only signs a hash of her message.

2 RSA Security

Some researches are still in progress in order to determine whether breaking RSA[5] is equivalent to prime factorisation or not. As we pointed out, decrypting a ciphertext c may be easier than factorising N, e.g. if $c = 0$. However, finding the private exponent d given e and N is indeed equivalent to factorising N. The decomposition of N into its prime factors gives us a straightforward way of recovering the private exponent d by repeating step 4 of RSA. We will show that given both exponents d and e, we are able to factorise $N = pq$. Let us present a probabilistic algorithm that realises this task.

Let us assume that we know e and d, and we want to factorise N. Let us define $\phi = (p-1)(q-1)$.

1. Choose a random $g \in [\![2, N-1]\!]$. If $gcd(g, N) \neq 1$, then we have found a non trivial divisor of N and the algorithm terminates. Otherwise, $gcd(g, N) = 1$ and Fermat's little theorem applies[6] and gives $g^{ed-1} \equiv 1 \mod N$ since $ed - 1$ is a multiple of ϕ.
2. As ϕ is even, we have $ed - 1 = 2^t r$ where r is odd and $t \geq 1$. Calculate $i \in [\![0, t[\![$ as the greatest exponent such that $(g^{2^i r} \neq 1) \vee (i = 0)$. The value $y = g^{2^i r}$ is a square root of unity in $\mathbb{Z}_n^*$ such that $y^2 \equiv 1 \mod N$. By virtue of the Chinese Remainder Theorem[7], y can take the value of each of the four square roots of 1 modulo $N = pq$.
3. If $y = \pm 1 \mod N$ then restart the algorithm from step 1 with another random g. Otherwise, we have $(y-1)(y+1) = 0 \mod N$ and thus we can recover a non trivial factor of N e.g. by calculating $gcd(y - 1, N)$.

Intuitively, the value of y at the end of step 2 is different from ± 1 with probability $1/2$. The expected number of trials before finding one random g revealing a non trivial factor of N is thus 2.

Example 1. *Let us show a baby example of how to factorise N given both the public and private exponents e and d of RSA. Let us choose* $p = 11$, $q = 7$, $N = pq = 77$, $\phi = (p-1)(q-1) = 60$. *We select* $e = 7$ *and the corresponding* $d = 43$.

1. *Let us choose a random* $g = 65$. *We note that* $ed - 1 = 300$ *and* $g^{ed-1} = 1$.
2. *We also have* $g^{(ed-1)/2} = 1$ *but* $g^{(ed-1)/4} = 43$. *Thus we have*[8] $y = 43$.
3. *A non trivial factor of N can thus be obtained with* $gcd(y - 1, N) = gcd(42, 77) = 7$.

The algorithm is probabilistic in the sense that its outcome depends on the random choice of g. If we had chosen $g = 62$ *for example, we would have been led to* $g^{(ed-1)/4} = -1 \mod N$ *and we would have had to*

[5] Breaking the RSA stands for inverting the RSA encryption function, i.e. to recover the plain text given a cipher text c and the public key (e, N).

[6] as in Case 1 of Appendix 1.3

[7] There are four square roots y associated to each of the possible combinations of $y = \pm 1 \mod p$ and $y = \pm 1 \mod q$

[8] Here $ed - 1 = 2^2.75$ and the exponent i is 0.

start over with another random g.

3 Modes of operations for block ciphers

The correctness of the Electronic CodeBook is straightforward as the encryption and decryption functions e_k and d_k are simply applied to every block independently. The correctness of the scheme only relies on the correctness of the encryption functions e_k and d_k used.

Let us show the correctness of the Cipher Block Chaining. We consider a plain text x and the corresponding ciphertext c. Let us write x' for the decrypted ciphertext. We have:

$$\begin{aligned} x'_1 &= d_k(c_1) \ominus iv \\ &= d_k(e_k(x_1 \oplus iv)) \ominus iv \\ &= x_1 \oplus iv \ominus iv \\ &= x_1 \end{aligned}$$

A similar reasoning applies for the following blocks $1 < i \leq l$:

$$\begin{aligned} x'_i &= d_k(c_i) \ominus c_{i-1} \\ &= d_k(e_k(x_i \oplus c_{i-1})) \ominus c_{i-1} \\ &= x_i \oplus c_{i-1} \ominus c_{i-1} \\ &= x_i \end{aligned}$$

Note 1. *We can first observe that we can choose any other symmetric encryption scheme e_k and d_k to encrypt the blocks without altering the proof.*

Note 2. *In practice, block ciphers operate on lists of bits (rather than lists of characters) that they divide into blocks of bits (rather than into single letters). The keys, and the initialisation vector (in the CBC mode) are thus replaced by bit vectors, and both operations $\oplus$ and $\ominus$ are replaced by the bitwise operation XOR.*

Note 3. *There are other modes of operations: Output Feedback (OFB), Cipher FeedBack Mode (CFB), Counter Mode (CTR) which present other features.*