CISC 468: CRYPTOGRAPHY

LESSON 9: THE RSA CRYPTOSYSTEM

Furkan Alaca

READINGS

- Section 1.4.1: Modular Arithmetic, Paar & Pelzl
- Section 1.4.2: Integer Rings, Paar & Pelzl
- Section 7.1: Introduction to RSA, Paar & Pelzl
- Section 7.2: Encryption and Decryption, Paar & Pelzl
- Section 7.3: Key Generation and Proof of Correctness, Paar & Pelzl
- Section 7.6: Finding Large Primes, Paar & Pelzl

RSA: INTRODUCTION

- RSA is one of the oldest and most widely-used asymmetric cryptographic schemes
- It is most often used for:
  - Encryption of small amounts of data (e.g., for key transport)
  - Digital signatures
- The underlying one-way function of RSA is the integer factorization problem:
  - Multiplying two large primes is computationally easy
  - Factoring the result is very hard

RSA: INTEGER RINGS

- RSA encryption and decryption is done in the integer ring $\mathbb{Z}_n$
- An integer ring $\mathbb{Z}_n$ consists of a set $\mathbb{Z}_n$ with two operations $+$ and $\times$, where
  - With respect to $+$ it is closed, associative, commutative, has an identity element, and each element has an inverse (i.e., $(\mathbb{Z}_n, +)$ is an Abelian group)
  - With respect to $\times$ it is associative, has an identity element, and is distributive over $+$, i.e., for $a, b, c \in \mathbb{Z}_{\mathbb{n}}$ we have

$$a \times (b + c) = (a \times b) + (a \times c).$$

    - Every element need not have a multiplicative inverse

RSA: ENCRYPTION

- RSA encrypts a plaintext $x \in \mathbb{Z}_n$ to a ciphertext $y \in \mathbb{Z}_n$
- A sender encrypts a plaintext x using the recipient's public key $k_{pub} = (n, e)$ as follows:

$$y = e_{k_{pub}}(x) = x^e \mod n.$$

RSA: DECRYPTION

- The recipient decrypts a ciphertext y using their private key $k_{pr} = d$ as follows:

$$x = d_{k_{pr}}(y) = y^d \mod n.$$

- In practice, x, y, n, d are very long numbers, usually 2048 bits (i.e., ~617 decimal digits) or more

RSA: A FEW REQUIREMENTS

- It must be computationally infeasible to determine the private key d given the public key (n, e)
- Since $x \in \mathbb{Z}_n$, we cannot encrypt more than l bits of plaintext, where $l = \lceil \log_2(n) \rceil$
- The encryption and decryption functions should be efficient to compute
- For a given n, there should be enough private-public key pairs to ensure that a brute-force attack is infeasible

RSA: KEY GENERATION

1. *Choose two large primes* p *and* q.
2. *Compute* $n = p \cdot q$.

We will later discuss how the two large primes are chosen.

Note that n is referred to as the *modulus*.

RSA: KEY GENERATION (CONT'D)

3. Compute $\Phi(n)$.

We know the prime factorization of n is pq, so we can use the formula:

$$\Phi(n) = (p-1)(q-1).$$

RSA: KEY GENERATION (CONT'D)

4. Select a public exponent
$e \in \{1, 2, \ldots, \Phi(n) - 1\}$ that is relatively prime with $\Phi(n)$.

This is often done by selecting a value for e and applying the EEA to find integers s and t such that

$$\gcd(\Phi(n), e) = s \cdot \Phi(n) + t \cdot e = 1.$$

If $\gcd(\Phi(n), e) = 1$, e is invertible, so a corresponding private key exists. Otherwise, select a new e and repeat the process.

RSA: KEY GENERATION (CONT'D)

5. Compute the private key d such that

$$d \cdot e \equiv 1 \mod \Phi(n).$$

After applying the EEA in the previous step, we know that the parameter t is the inverse of e, and thus

$$d = t \mod \Phi(n).$$

RSA: EXAMPLE

Alice encrypts and sends the message $x = 4$ to Bob:

Alice		**Bob**
message $x = 4$		1. choose $p = 3$ and $q = 11$ 2. $n = p \cdot q = 33$ 3. $\Phi(n) = (3-1)(11-1) = 20$ 4. choose $e = 3$ 5. $d \equiv e^{-1} \equiv 7 \bmod 20$
	$\xleftarrow{k_{pub}=(33,3)}$	
$y = x^e \equiv 4^3 \equiv 31 \bmod 33$		
	$\xrightarrow{y=31}$	
		$y^d = 31^7 \equiv 4 = x \bmod 33$

RSA: PARAMETER LENGTH REQUIREMENTS

- Practical RSA parameters should be much longer
- See the RSA Factoring Challenge to get an idea of which key lengths are currently breakable
  - Refer here to understand the numbering scheme
- 1024-bit RSA offers 80-bit level of security
  - No longer sufficient: According to NIST recommendation, 1024-bit RSA keys should no longer be in use after 2015
- 2048-bit RSA offers 112-bit level of security: This is the currently-recommended minimum

RSA: EXAMPLE WITH 1024-BIT MODULUS

$$
\begin{aligned}
p = {} & E0DFD2C2A288ACEBC705EFAB30E4447541A8C5A47A37185C5A9 \\
& CB98389CE4DE19199AA3069B404FD98C801568CB9170EB712BF \\
& 10B4955CE9C9DC8CE6855C6123_h \\
q = {} & EBE0FCF21866FD9A9F0D72F7994875A8D92E67AEE4B515136B2 \\
& A778A8048B149828AEA30BD0BA34B977982A3D42168F594CA99 \\
& F3981DDABFAB2369F229640115_h \\
n = {} & CF33188211FDF6052BDBB1A37235E0ABB5978A45C71FD381A91 \\
& AD12FC76DA0544C47568AC83D855D47CA8D8A779579AB72E635 \\
& D0B0AAAC22D28341E998E90F82122A2C06090F43A37E0203C2B \\
& 72E401FD06890EC8EAD4F07E686E906F01B2468AE7B30CBD670 \\
& 255C1FEDE1A2762CF4392C0759499CC0ABECFF008728D9A11ADF_h \\
e = {} & 40B028E1E4CCF07537643101FF72444A0BE1D7682F1EDB553E3 \\
& AB4F6DD8293CA1945DB12D796AE9244D60565C2EB692A89B888 \\
& 1D58D278562ED60066DD8211E67315CF89857167206120405B0 \\
& 8B54D10D4EC4ED4253C75FA74098FE3F7FB751FF5121353C554 \\
& 391E114C85B56A9725E9BD5685D6C9C7EED8EE442366353DC39_h \\
d = {} & C21A93EE751A8D4FBFD77285D79D6768C58EBF283743D2889A3 \\
& 95F266C78F4A28E86F545960C2CE01EB8AD5246905163B28D0B \\
& 8BAABB959CC03F4EC499186168AE9ED6D88058898907E61C7CC \\
& CC584D65D801CFE32DFC983707F87F5AA6AE4B9E77B9CE630E2 \\
& C0DF05841B5E4984D059A35D7270D500514891F7B77B804BED81_h
\end{aligned}
$$

RSA: PROOF OF CORRECTNESS

To prove that decryption works, we show that

$$(x^e)^d \equiv x \bmod n.$$

Since $n = pq$, by the Chinese Remainder Theorem it is equivalent to instead show that $(x^e)^d \equiv x \bmod p$ and $(x^e)^d \equiv x \bmod q$, so that is what we will do.

RSA: PROOF OF CORRECTNESS (2)

1. Since $ed \equiv 1 \bmod \Phi(n) \equiv 1 \bmod (p-1)(q-1)$, there exists an integer k such that $ed = 1 + k(p-1)(q-1)$.
2. If $\gcd(x, p) = 1$, then by Fermat's Little Theorem,

$$x^{p-1} \equiv 1 \bmod p.$$

3. Raising both sides to the power $k(q-1)$ and then multiplying both sides by x gives

$$x^{1+k(p-1)(q-1)} \equiv x \bmod p.$$

RSA: PROOF OF CORRECTNESS (3)

4. Since $ed = 1 + k(p-1)(q-1)$ and $x^{1+k(p-1)(q-1)} \equiv x \bmod p$, we have $(x^e)^d \equiv x \bmod p$.
5. We started with the assumption that $\gcd(x, p) = 1$, but the above congruence is also valid if $\gcd(x, p) = p$,

RSA: PROOF OF CORRECTNESS (4)

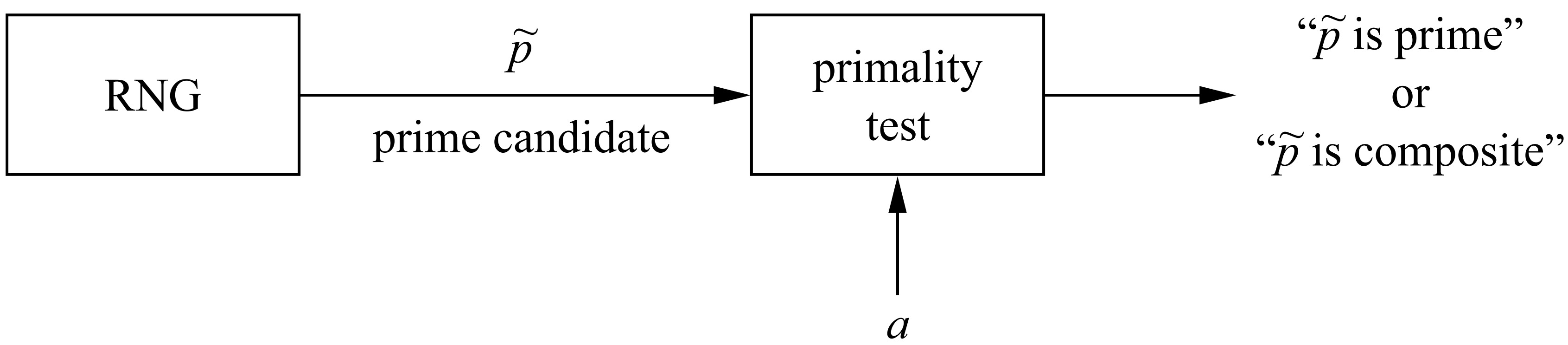
6. We can repeat Steps 1-5 to also show that $(x^e)^d \equiv x \bmod q$.

7. Since $(x^e)^d \equiv x \bmod p$ and $(x^e)^d \equiv x \bmod q$,

$$y^d \equiv (x^e)^d \equiv x \bmod pq \equiv x \bmod n.$$

RSA: FINDING LARGE PRIMES

- Each prime p and q should be about half the bit length of n
- The general approach is to generate integers at random and check them for primality
  - RNG should be non-predictable: Guessing one or two of the primes suffices an attacker for decrypting the ciphertext
    - Refer to the infamous Debian random number bug



RSA: FINDING LARGE PRIMES (2)

The approach's feasibility relies on the answer to two questions:

1. How many random integers do we need to test before we find a prime?
   - If the likelihood of a prime is too small, it may take too long.
2. How fast can we check whether a random integer is prime?
   - If the test is too slow, it would make this approach impractical.

HOW COMMON ARE PRIMES?

- Primes become less frequent as their values increase
- By the prime number theorem, the probability of a random integer $\tilde{p}$ being prime is approximately $\frac{1}{\ln(\tilde{p})}$
- We only check odd integers, raising the probability to $\frac{2}{\ln(\tilde{p})}$

HOW COMMON ARE PRIMES? (CONT'D)

- **Example**: For RSA with a 2048-bit modulus, p and q should each have a length of ~1024 bits, i.e., $p, q \approx 2^{1024}$. The probability that such a random odd number $\tilde{p}$ is prime is $\frac{2}{\ln(2^{1024})} = \frac{2}{1024 \ln(2)} = \frac{1}{512 \ln(2)} \approx \frac{1}{355}$, meaning that we can expect to test up to 355 numbers before we find a prime.

PRIMALITY TESTING

- To test whether a randomly-generated number is prime, it might be tempting to try to factor it
  - But with RSA we use numbers too large to factor
- More efficient primality tests exist, which output either:
  - "$\tilde{p}$ is composite", which is always a true statement; or,
  - "$\tilde{p}$ is prime", which is only true with a high probability
- These tests are typically repeated s times, where s is a *security parameter*, to reduce the error probability

MILLER-RABIN PRIMALITY TEST

1. For an integer $\tilde{p}$, write $\tilde{p} - 1 = 2^u r$ such that r is odd.
2. $\tilde{p}$ is definitely composite if we find an integer a such that $a^r \not\equiv 1 \mod \tilde{p}$ and $a^{r2^j} \not\equiv \tilde{p} - 1 \mod \tilde{p}$ for all $j \in \{0, 1, \ldots, u-1\}$.
   (Note that for $j = 0$, the second expression simplifies to $a^r \not\equiv \tilde{p} - 1 \mod \tilde{p}$)
3. Otherwise, $\tilde{p}$ is probably prime.

MILLER-RABIN PRIMALITY TEST: EXAMPLE

Let $\tilde{p} = 91$, and select a security parameter of $s = 4$.
We write $\tilde{p} - 1 = 2^1 \cdot 45$ and choose random values for a:

1. Let $a = 12$. So, $a^r \equiv 12^{45} \equiv 90 \bmod 91$.
   Hence, $\tilde{p}$ is probably prime.
2. Let $a = 17$. So, $a^r \equiv 17^{45} \equiv 90 \bmod 91$.
   Hence, $\tilde{p}$ is probably prime.
3. Let $a = 38$. So, $a^r \equiv 38^{45} \equiv 90 \bmod 91$.
   Hence, $\tilde{p}$ is probably prime.
4. Let $a = 39$. So, $a^r \equiv 39^{45} \equiv 78 \bmod 91$.
   Hence, $\tilde{p}$ is composite.
   (The numbers 12, 17, and 38 are called "liars for 91")

MILLER-RABIN PRIMALITY TEST: NUMBER OF ROUNDS REQUIRED

- The Miller-Rabin error-probability bound is at most $(\frac{1}{4})^s$, for a security parameter s
- NIST recommends "matching" the error probability with the security level of the key
  - e.g., for 2048-bit RSA, an error probability of 2^{-112} is sensible, but 2^{-100} (achieved by 50 rounds) is said to be sufficient for all prime lengths for many applications
- For RSA, if a non-prime p or q survives the test, this will be detected since the algorithm will not work correctly
  - Can be more problematic for other algorithms