Padding Schemes for RSA and their Security

Calvin Ryan D'Souza

Advisor:- Prof. Stanisław Radziszowski

REVIEW OF PROJECT GOALS

- Rivest-Shamir-Adleman (RSA) Encryption scheme is deterministic i.e., it has no random component. This makes RSA susceptible to chosen plaintext attacks. Padding schemes help solve this problem by making RSA probabilistic in nature.

- Analyze computational costs of using padding schemes with RSA, since RSA is already expensive to implement. Do padding schemes introduce significant overhead?

- Probabilistic RSA is shown to be secure against chosen plaintext attacks but implementing RSA incorrectly could be detrimental. Analyze security aspects related to RSA implementations.

MILESTONE 1 GOALS

- Review literature on RSA and Padding Schemes:
  - RSA Cryptography Specifications version 2.2.
  - RSA Cryptography Specifications version 1.5.
- Generate parameters for RSA:
  - Generating primes/safe primes to calculate RSA modulus.
  - Bit lengths for primes generated are 512, 647, 813, 1024 and 2048.
  - Public key exponent (e) was chosen as 65537.
- Implement RSA Cryptosystem.

ENVIRONMENT SPECIFICATIONS

- Device Specifications:
  - Processor: AMD Ryzen 9 5900HS
  - RAM: 16 GB
- Programming Environment:-
  - IDE:- Visual Studio Code
  - Language:- Java 11
- Prime numbers are used in code using java.math.BigInteger.

IMPLEMENTATION OF PARAMETER SELECTION ALGORITHM

- For number of primes to be generated:
  - Initialize bit generator
  - Generate required number of bits (n)
  - Set bits 0 and (n-1) to 1 (i.e., Generate large odd number). Let this number be p.
  - Check if p has any factors within 2^{20}-1.
  - If no factors present, perform primality tests to determine if p is prime.
    - Primality tests performed are Miller-Rabin and Lucas-Lehmer.

- Safe prime check is performed as follows:-
  - Pollards p-1 algorithm can be used for determining p from p-1.
  - When p is prime, check (p – 1) = 2 * prime.
  - If above condition is satisfied, save generated prime.

RESULTS

- Old Implementation of Parameter Selection (in Java):-
  1. Randomly choose Seed value for bit generation.
  2. Randomly generate n-bit odd number ***p***.
  3. Perform primality tests to check if ***p*** prime.
  4. If prime, check if ***(p - 1) = 2 * (prime) [Safe prime]***. Else go to Step 1.
  5. If yes, choose number for RSA. Else go to Step 1.

	Algorithm	Number of primes	Time taken (approx.)
Basic Primes	Old	10	3 secs.
Safe Primes	Old	3	6 mins.
Basic Primes	Current	5	8 mins.

WHAT'S NEXT?

- Evaluate computational performance of RSA:-
  - Variation of performance across bit lengths.
  - Current implementation vs default implementations.
- Observe security of RSA:-
  - Safe primes vs basic primes.
  - Security of RSA with Public key Cryptography Standards (PKCS) #1 v1.5
- Proposed use of SHA-3/SHAKE with Optimal asymmetric encryption padding.

ANY QUESTIONS?

THANK YOU