TU Graz
SCIENCE
PASSION
TECHNOLOGY

T1 Asymmetric Analysis and Multiparty Computation

Maria Eichlseder   Daniel Kales

Applied Cryptography 2 – ST 2020

A Wiener's Attack on RSA

Wiener's Attack on RSA	Points
Implement Wiener's attack on RSA to recover small d ($\rightarrow$ L1):	
a Compute n-th convergents of continued fractions for $\mathbb{Q}$	
b Recover private key from given RSA public key & decrypt message	

[Wie90] Michael J. Wiener. **Cryptanalysis of short RSA secret exponents.** IEEE Transactions on Information Theory 36.3 (1990), pp. 553–558. DOI: 10.1109/18.54902.

A Questions

Example:
$N = 9449868410449$, $e = 6792605526025$, $d < \frac{1}{3}\sqrt[4]{N} \approx 584$.

1. Perform Wiener's attack by computing the convergents of $\frac{e}{N}$:

$$\frac{p_i}{q_i} = \quad \frac{1}{1}, \quad \frac{2}{3}, \quad \frac{3}{4}, \quad \frac{5}{7}, \quad \frac{18}{25} \quad \frac{23}{32}, \quad \frac{409}{569}, \quad \dots$$

2. Test: $d = 569$ if $M^{e \cdot 569} = M$ ✓

3. Pick $x = 2$:

$$x^{(ed-1)/2^1} = x^{(ed-1)/2^2} = \dots = x^{(ed-1)/2^5} = 1$$
$$x^{(ed-1)/2^6} = x^{60390508504816} \equiv 8233548335126 = y \neq \pm 1$$

4. $p = \gcd(N, y-1) = 1234577 \;\Rightarrow\; N = 1234577 \cdot 7654337$

B Multiparty Computation with Oblivious Transfers

Multiparty Computation with Oblivious Transfers	Points
Implement 1-out-of-2 OT to enable Multiparty Computation problem ($\rightarrow$ L2)	
a Implement 1-out-of-2 OT	
b Implement 1-out-of-N OT	
c Implement GMW and evaluate a small circuit	
d Evaluate the AES-128 encryption circuit	

[NP01] Moni Naor and Benny Pinkas. **Efficient oblivious transfer protocols.** SODA. ACM/SIAM, 2001, pp. 448–457.

B Naor-Pinkas OT

Naor-Pinkas Oblivious Transfer

Alice		Bob
Input: (m_0, m_1)	Public parameters: mult. Group G of prime order q, generator g	Input: ν
		$a, b \xleftarrow{\$} \mathbb{Z}_q$
		$c_\nu = ab, c_{1-\nu} \xleftarrow{\$} \mathbb{Z}_q$
		$x = g^a, y = g^b$
		$z_0 = g^{c_0}, z_1 = g^{c_1}$
	$\xleftarrow{(x, y, z_0, z_1)}$	
verify $z_0 \neq z_1$		
$r_0, s_0, r_1, s_1 \xleftarrow{\$} \mathbb{Z}_q$		
$w_0 = x^{s_0} \cdot g^{r_0}, k_0 = z_0^{s_0} \cdot y^{r_0}$		
$w_1 = x^{s_1} \cdot g^{r_1}, k_1 = z_1^{s_1} \cdot y^{r_1}$		
	$\xrightarrow{(w_0, \mathbb{E}_{k_0}(m_0), w_1, \mathbb{E}_{k_1}(m_1))}$	
		$k_\nu = (w_\nu)^b$
		$m_\nu = \mathbb{E}_{k_\nu}^{-1}(\mathbb{E}_{k_\nu}(m_\nu))$

B Building $\binom{N}{1}$-OT

- Instantiate directly if supported (e.g., Naor-Pinkas OT)
- Build from $\binom{2}{1}$-OT (Idea: transfer encryption key per bit of ν)

$\binom{N}{1}$-OT from $\binom{2}{1}$-OT

- Sender prepares $L = \log_2(N)$ keypairs: $(k_1^0, k_1^1), \ldots, (k_L^0, k_L^1)$
- Sender encrypts and sends item m_i:

$$C_i = m_i \oplus \left(\bigoplus_{j=1}^{L} \mathbb{E}(k_j^{i_j}, i) \right)$$

- For each bit b of ν: Perform $\binom{2}{1}$-OT to give chooser k_j^b
- Chooser has all keys to decrypt C_ν

C Factoring with Factor Bases and Sieving

Factoring with Factor Bases and Sieving	Points
Implement Quadratic Sieve factorization ($\rightarrow$ L1):	
a Implement factoring with factor bases 1: Collect squares	
b Implement factoring with factor bases 2: Combine relations	
c Implement sieving with a quadratic equation	
d Apply the combined QS algorithm to factor 100-bit N	

[Pom85] Carl Pomerance. **The Quadratic Sieve Factoring Algorithm.** Advances in Cryptology – EUROCRYPT 84. Vol. 209. LNCS. Springer, 1985, pp. 169–182. DOI: 10.1007/3-540-39757-4_17.

C Factoring with Sieving – Cheatsheet

1. Select factor base of small prime numbers $\mathcal{B} = \{p_1, p_2, \ldots, p_k\}$
   For each prime p_j, solve $\alpha_j^2 - n \equiv 0 \pmod{p_j}$ (0 to 2 solutions)

2. For all x_i in $[\sqrt{n} - C, \sqrt{n} + C]$, store (x_i, Y_i) with $Y_i = x_i^2 - n$

3. Sieve: For each α_j, divide all Y_i for $x_i = \alpha_j + k \cdot p_j$ by p_j

4. Collect all relations (x_i, Y_i) with smooth $Y_i = \prod_j p_j^{e_{ij}}$

5. Solve: select subset of Y_i's such that their product is square

6. $x = \prod x_i$ and $y = \sqrt{\prod Y_i} \to$ hope that $\gcd(x \pm y, n) \in \{p, q\}$

Factoring with Quadratic Sieve: Example I

Factor $n = 2769$:

1 Choose factor base $\mathcal{B} = \{2, 3, 5, 7, 11\}$ and find α_j:

$$\begin{array}{llll}
\alpha_2^2 \equiv 2769 \equiv 1 \mod 2 & \rightarrow \alpha_2 = \pm 1, & a_i = 49, 51, 53, 55, 57 \\
\alpha_3^2 \equiv 2769 \equiv 0 \mod 3 & \rightarrow \alpha_3 = 0, & a_i = 51, 54, 57 \\
\alpha_5^2 \equiv 2769 \equiv 4 \mod 5 & \rightarrow \alpha_5 = \pm 2, & a_i = 52, 53, 57 \\
\alpha_7^2 \equiv 2769 \equiv 4 \mod 7 & \rightarrow \alpha_7 = \pm 2, & a_i = 54, 58 \\
\alpha_{11}^2 \equiv 2769 \equiv 8 \mod 11 & \text{no solution!} &
\end{array}$$

If your library can't compute $\alpha = \sqrt{n} \in \mathbb{Z}_p$: Tonelli-Shanks algo

Factoring with Quadratic Sieve: Example II

2 Collect relations by sieving $[\sqrt{n} - C, \sqrt{n} + C] = [49, 57]$:

a_i	49	50	51	52	53	54	55	56	57
b_i	-368	-269	-168	-65	40	147	256	367	480
$\div 2$	2^4		2^3		2^3		2^8		2^5
$\div 3$			3^1			3^1			3^1
$\div 5$				5^1	5^1				5^1
$\div 7$			7^1			7^2			
(-1)	(-1)	(-1)	(-1)	(-1)					
Rest	23	269	1	13	1	1	1	367	1

Factoring with Quadratic Sieve: Example III

3 Solve the linear system mod 2: $a_i \in \{53, 54, 57\}$ sum to 0

a_i	49	50	51	52	53	54	55	56	57
b_i	-368	-269	-168	-65	40	147	256	367	480
$\div 2$			1		1	0	0		1
$\div 3$			1		0	1	0		1
$\div 5$			0		1	0	0		1
$\div 7$			1		0	0	0		0
(-1)			1		0	0	0		0
Rest	✗	✗	✓	✗	✓	✓	✓	✗	✓

(keyword: kernel/nullspace of a matrix over $\mathbb{Z}_2$)

4 $x = \prod a_i = 53 \cdot 54 \cdot 57 = 163134$

$y = \sqrt{\prod b_i} = 2^{(3+5)/2} \cdot 3^{(1+1)/2} \cdot 5^{(1+1)/2} \cdot 7^{2/2} = 1680$

5 $\gcd(x + y, n) = \gcd(164814, 2769) = 39$ $(n = 3 \cdot 13 \cdot 71)$

Rules

Coding:

- Pick your favourite programming language
- You'll need support for big-integer and modular arithmetic
- Number theory (primes) and linear algebra recommended
- If it's not open-source, please ask us first!
- Suggestions: C/C++ with libraries, Java, Python/Sage

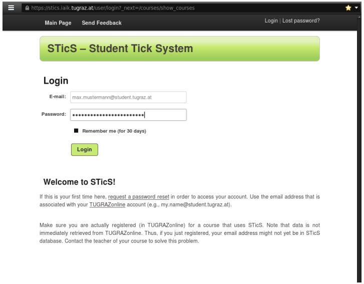
Submission:

- Submit your team's code as a {`zip`, `tar.gz`} archive
- Add a file `README.`{`md`, `txt`, `pdf`} on the top level (design choices, limitations, howto, runtime).
- Include `Makefile` and/or clear instructions to run

Submission (Deadline 30 April 2020)

Upload your KU submissions in the Student Tick System (STicS):

https://stics.iaik.tugraz.at



Questions?

If you're unsure how to tackle some of the steps (finding $\alpha, \ldots$),

ask anytime (lecture, newsgroup, email, your colleagues)!