Asymmetric Cryptanalysis – Submission Guidelines

- Timeline: Release **12. 05. 2016**; Question time **02. 06. 2016**; Submission **23. 06. 2016**
- Upload your team's submission on `https://stics.iaik.tugraz.at/`. Each task is uploaded separately. Don't forget to tick the selected tasks!
- Submit your code as a {`zip`, `tar.gz`} archive. Add a file `README.`{`md`, `txt`, `pdf`} on the top level that documents interesting properties and limitations of your submission.
- You can use your favourite programming language and libraries for the necessary big-integer and modular arithmetic. Consider using a computer algebra system, such as Sage (`https://sage.tugraz.at` with TUGRAZonline login). Please document your choices, as well as how to compile and use your implementations in the `README`. If you intend to use non-free software, please clarify with us beforehand.

2–A Wiener's Attack on RSA (4 Points)

Implement the attack by Wiener [Wie90] to recover small RSA private exponents d.

(a) Continued fractions for $\mathbb{Q}$ (2 Points): Implement the computation of n-th convergents of the continued fraction expansion to approximate rational numbers.

(b) Recover RSA private key (2 Points): Use your implementation to recover Bob's private key d, p, q from the following 1024-bit RSA public key with modulus N and exponent e:

$$
\begin{aligned}
N = \texttt{0x1cb33aad9d96e572b13204d77700515cf489028e35811ba23a2aa7763fb443a2}\\
\texttt{2d6a7bac40a279e98faa91f82898efa3cfc57e11417f47f604093cd97371ff4e}\\
\texttt{6d6e572c9c476ac6155719a4dbba3d93ce8891ea5116fe7a0502612052879f1c}\\
\texttt{3c82c699dfb69518c6ed43871b88edb40da0b3cec6eabecc9988adf8a4294547}
\end{aligned}
$$

$$
\begin{aligned}
e = \texttt{0x07f5e658edd83082b49740286814b8c63a2b0c4957d909af3a9b574e2f9d8d72}\\
\texttt{bbe332244148a313350dac5657287e1e383e4b50f0a7d5078d4f6d48c19144e0}\\
\texttt{4be6cb5bfef0e7fe100d03d51a5c6df8911ad49cf61eeaae849866ae9bbfa17a}\\
\texttt{820374ad62ac1fec5692dc88cc85006975e87836cdda7888ddd822c0b47d5311}
\end{aligned}
$$

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

2–B Discrete Logarithms with Pollard-ρ (8 Points)

Implement the algorithm by Pollard [Pol78] to compute the discrete logarithm x in $y = g^x \pmod p$. Use it to recover Alice's and Bob's private keys a, b from a Diffie-Hellman key exchange.

(a) Find equation for 32-bit problem (4 Points): Implement the Pollard-ρ algorithm to find an equation $x \cdot (a_j - a_k) \equiv (b_k - b_j) \pmod{p-1}$ for x. Apply your implementation to get equations for α, β from the following Diffie-Hellman key exchange:

Domain parameters:	generator $g =$ `0x00000002` $\in \mathbb{Z}_p^*$
	prime $p =$ `0xfffffffb`
Alice $\to$ Bob:	$g^\alpha =$ `0x4ebb660a`
Bob $\to$ Alice:	$g^\beta =$ `0xe9467263`

(b) Determine correct solution for 32-bit problem (2 Points): Recover α, β from the key exchange in (a) by testing all candidate solutions of the equations.

(c) Find solution for 64-bit problem (2 Points): Find α or β to recover the key $g^{\alpha\beta}$:

Domain parameters:	generator $g =$ `0x0000000000000005` $\in \mathbb{Z}_p^*$
	prime $p =$ `0x8ac25c704f8e5947`
Alice $\to$ Bob:	$g^\alpha =$ `0x82c2c001a3ccff71`
Bob $\to$ Alice:	$g^\beta =$ `0x07a3fb4211394fd1`

[Pol78] J. M. Pollard. "Monte Carlo Methods for Index Computation $(\bmod\, p)$". In: Mathematics of Computation 32.143 (1978), pp. 918–924. DOI: `10.2307/2006496`.

2–C Factoring with Continued Fractions (12 Points)

Implement CFRAC factoring [LP31; MB75], and use it to factor the RSA modulus N.

(a) Factoring with random squares (6 Points): Implement Dixon's basic factor-base factoring algorithm: Pick random x_i, factor $x_i^2 \pmod n$ with respect to a factor base $\mathcal{B}$, and use linear algebra to combine these factorizations to find solutions $x^2 \equiv y^2 \pmod n$.

(b) Continued fractions for $\mathbb{R}$ (2 Points): Implement the computation of n-th convergents of the continued fraction expansion to approximate irrational numbers like $\sqrt{N}$.

(c) Factor 32-bit number (2 Points): Combine your ingredients and factor $N =$ `0x347b702f`.

(d) Factor 64-bit number (2 Points): Factor $N =$ `0x8de50360f22507bf`.

[LP31] D. H. Lehmer and R. E. Powers. "On Factoring Large Numbers". In: Bulletin of the American Mathematical Society 37.10 (1931), pp. 770–776. DOI: `10.1090/S0002-9904-1931-05271-X`.

[MB75] M. A. Morrison and J. Brillhart. "A Method of Factoring and the Factorization of F_7". In: Mathematics of Computation 29.129 (1975), pp. 183–205. DOI: `10.2307/2005475`.