Cryptography and Architectures for Computer Security

Exam Code: 095947 (old 090959), A.Y. 2015–2016, Semester: 2

Prof. G. Pelosi

July 4th, 2016 – Exam Session

Name: ..... Surname: .....

Student ID: ..... Signature: .....

Time: 2h:30'. Use of textbooks, notes, phones or Internet connected devices is not allowed. Prior to turn in your paper, write your name on any additional sheet and sign it.

Question 1 [4 pts]
A5/1 is a stream cipher with a 64-bit key employed to secure GSM communications, which is still in use whenever the link between the mobile phone and the cell is downgraded to GSM. Performing an exhaustive key search to find a key is thus within practical feasibility, although taking quite some computation. To this end, Time-to-Memory Tradeoffs (TMTO) are an appealing approach, and have effectively been used to crack A5/1 encrypted messages in practice.

(a) Assuming to employ a nVidia Geforce GTX 1070, able to compute around 29×2^{30} A5/1 keystreams per second, each one 64-bit long, calculate how much memory is required to break A5/1 in less than 0.1s in the following two cases:

i) GDDR5 DRAM, access time 0.5μs,

ii) SATA-SSD 10μs.

Assume, for the sake of simplicity, that the data structure in memory allows you to check for the presence of a given value with a single access, and that you are given an A5/1 plaintext-ciphertext pair, where both are 64-bit long to the end of finding the key.

Argue on which one of the two strategies may be practically viable.

(b) Which one(s) of the following compensation for the weakness of A5/1 would render unfeasible a TMTO based strategy? Does it prevent an exhaustive search for a single key too?

1. Employ two different instances of A5/1, one running on the actual key, and one on a 32-bit random salt (supposed to be sent at the beginning of the communication in clear), padded with zeros up to 64-bit. Encrypt the plaintext with the first engine, and then with the second one.
2. Add a 192b random *salt* (supposed to be sent at the beginning of the communication in clear) to the key, splitting it into three 64-bit words and combining them via `xor` with the key, employing the result as the A5/1 key.

Discuss the resistance against a bruteforce attack of each of these strategies.

Solution:

(a) Given that it is required to obtain a 64-bit key via TMTO, each element stored in the rainbow tables chains will need to be $2 \times 64\text{b} = 16\text{B}$ wide yielding 64 Mchains (2^{26}) per GiB of available space. The time required to perform a single computation and lookup can be obtained as $\frac{1}{29 \times 2^{30}} + 5 \times 10^{-7}$s for the GDDR based solution and $\frac{1}{29 \times 2^{30}} + \times 10^{-5}$s for the disk based one. This in turn caps the maximum number of lookups which can be performed in 0.1s to $2 \times 10^5 \approx 2^{17}$ for the GDDR case and $10^4 \approx 2^{13}$ in the SSD case. Consequentially, the remaining amount of computation should be traded off entirely for space: in particular, in the GDDR case requires $2^{64-17} = 2^{47}$ chains to be tabulated, while the SSD case calls for $2^{64-13} = 2^{50}$. Turning these figures into required space yields 2PiB for the GDDR case and 16PiB for the SSD one.

(b) Solution 1) is able to ward off TMTOs (under the assumption that A5/1 is not a group) as the required key+salt space to be tabulated is effectively bumped up to 2^{96}, which is beyond feasibility. However, it does not hinder a single exhaustive key search since the salt is sent in cleartext. Solution 2) does not add to the security of the system in any way. Indeed, adding the salt via xor to the key does not change the fact that the effective keyspace of the cipher is 64 bits, and thus the attacker will be able to retrieve the value of the combination between the key and the salt employing the same rainbow tables computed for the unsalted version.

Question 2 [4 pts]

While evaluating block cipher designs, you encounter an SPN one, where you are free to tweak one out of the three following features: the S-Box, the linear diffusion layer or the number of rounds. The basic version of the block cipher employs 16 8×8 bit identical S-boxes for the substitution layer, and a single linear diffusion layer per round. The entire cipher is r round long, so to provide a 2^{128} linear and differential cryptanalysis immunity, as the block size of the cipher is 128b.

(a) Compute r so that the desired immunity is guaranteed, assuming that the best differential probability for the S-box is $\frac{1}{8}$, the best linear bias is $\frac{1}{4}$ and the diffusion layer guarantees that a single bit in input to it will influence at least 4 different S-boxes in the next round. Perform the computations providing a reasonable, but conservative approximation.

(b) Assume to be measuring the latency of the cipher in round-sized units, and its area if implemented in hardware, following the same criterion. As a consequence, the cipher you just analyzed has a latency of r_t, and an area of r_a. Do the following tweaks turns out to be more efficient than the original cipher, i.e., have a lower area-time product?

- Strengthen the S-box: $\frac{1}{32}$ maximum differential probability, $\frac{1}{16}$ maximum linear bias, +50% on the area of a single round, with no latency penalty.
- Perfect diffusion layer: a single bit change in the diffusion layer will change the input of all the following S-Boxes. +25% on the area of a single round, +5% on the latency.

Solution:

(a) Given the properties of the cipher, the number of active boxes in its round will be 1 for the first round, 4 for the second and 16 for the third onwards. Computing a conservative estimate for the differential probability after the third round yields $2^{-3} \times (2^{-3})^4 \times (2^{-3})^{16} = 2^{-63}$, in turn implying the need for 2^{63} plaintext-ciphertext pairs to

perform differential cryptanalysis on the cipher. Adding a round beyond the third adds a factor 2^{-48} to the differential probability, in turn implying that a differential after the fifth round will be the first one with a probability smaller than 2^{-128}. As a consequence, to prevent differential cryptanalysis, $r = 5 + 1$ is sufficient. Computing the linear bias for the first three rounds yields $2^{-2} \times (2^{-2})^4 \times (2^{-2})^{16} \times 2^{21-1} = 2^{-42+20} = 2^{-22}$. This in turn results in the need of $(2^{-22})^{-}2 = 2^{44}$ plaintext and ciphertext pairs being needed to break a cipher with $3 + 1$ rounds with linear cryptanalysis. Adding a round provides a $2 \times 2^{-32} = 2^{-31}$ factor on the linear bias. As a consequence, the cipher needs to be $5 + 1$ rounds long to resist linear cryptanalysis.

(b) The area-time cost of the original cipher is $6r_a r_t$. Strengthening the S-Box results in a $2^{-5} \times (2^{-5})^4 \times (2^{-5})^{16} = 2^{-105}$ differential probability and a $2^{-4} \times (2^{-4})^4 \times (2^{-4})^{16} \times 2^{21-1} = 2^{-84+20} = 2^{-62}$ linear bias after the third round, requiring 2^{105} and 2^{124} plaintext-ciphertext pairs to perform the respective cryptanalyses. Adding a single round contributes enough in both cases to make the cipher immune, in turn resulting in $r = 4 + 1$ rounds. The area-time cost of the cipher will thus be $5 \times 1.5 \times 1 = 7.5 r_t r_a$ Changing the diffusion layer yields a $2^{-3} \times (2^{-3})^1 6 \times (2^{-5})^{16} = 2^{-99}$ differential probability and a $2^{-2} \times (2^{-2})^1 6 \times (2^{-2})^{16} \times 2^{21-1} = 2^{-66+20} = 2^{-46}$ linear bias after the third round, requiring 2^{99} and 2^{92} plaintext-ciphertext pairs to perform the respective cryptanalyses. Adding a round is enough to ward off both linear and differential cryptanalyses (2^{154} and 2^{147} ptx-ctx pairs needed, respectively). The total length of the cipher is thus $4+1$ rounds, yielding an area-time cost of $5 \times 1.25 \times 1.05 = 6.5625 r_t r_a$. As a consequence, the original cipher, despite having an extra round is still the most efficient solution, followed by the alteration to the diffusion layer and, finally, the strengthened S-box version.

Question 3 [8 pts]
Consider the cyclic group G=$(\mathbb{Z}_{41}^*, \cdot)$.

(a) List the subgroups of G (both proper and trivial) showing for each of them the order and one generator.

(b) Given the following discrete logarithms, x_0, x_1, state if each one of them exists, motivating your answer and compute them.

$$x_0 \equiv \log_2^{\mathtt{D}}((-1)^{40^3-1})$$
$$x_1 \equiv \log_4^{\mathtt{D}}(2)$$

(d) Describe how to properly choose the parameters of the (Ephemeral) Diffie-Hellman protocol and point out the advantages of using it.

Solution:

(a) Let H_m denote the subgroup of order m.
$H_1 = \langle 1 \rangle$,
$H_2 = \langle 40 \rangle$,
$H_4 = \langle 9 \rangle$,
$H_5 = \langle 10 \rangle = \langle 37 \rangle$,
$H_8 = \langle 3 \rangle = \langle 38 \rangle$,
$H_{10} = \langle 4 \rangle = \langle 23 \rangle$,

$H_{20} = \langle 2 \rangle = \langle 8 \rangle$,
$H_{40} = \langle 7 \rangle$.

Note: testing g=2 and g=3 as generators, allows to list all the subgroups $H_1 - - H_{20}$ with at least one generator

(b) $x_0 \equiv_{20} \log_2^{\mathtt{D}}((-1)^{40^3-1}) \equiv_{20} \log_2^{\mathtt{D}}((-1)^{40^3-1 \bmod 40}) \equiv_{20} -\log_2^{\mathtt{D}}(-1)$.
The logarithm exists, as the argument is a value of the subgroup generated by the base (note that -1 belongs to the group since $2^{20} \equiv 1$, then $(2^{10})^2 \equiv (-1)^2 \equiv 1$.
$x_0 \equiv_{20} -\log_2^{\mathtt{D}}(2^{10}) \equiv_{20} -10 \equiv_{20} 10$
(Alternate method: apply DLog finding algorithm)

$x_1 \equiv_{10} \log_4^{\mathtt{D}}(2)$ does not exists.
The base and the argument belongs to separate subgroups
($4^x \equiv_{41} 2 \Rightarrow 2^{2x} \equiv_{41} 2 \Rightarrow 2x \equiv_{40} 1$, and the inverse of 2 does not exist as $gcd(2, 40) \neq 1$)

(c) see lectures...

Question 4 [4 pts]
Consider the finite field $\mathbb{F}_{3^4}$

(a) Write the number of irreducible and primitive polynomials

(b) Verify that $f_1(x) = x^4 + x - 1 \in \mathbb{F}_3[x]$ is primitive and show each of its roots in $\mathbb{F}_{3^4}$.

Solution:
(a) $N_1(3)=3$, $N_2(3)=\frac{3^2-3}{2}=3$, $3^4=N_1(3)+2N_2(3)+4N_4(3) \Rightarrow N_4(3)=\frac{81-3-6}{4}=18$.
$n=3^4-1=80=2^4 \cdot 5$. $\varphi(n)=32$. $M_4(3)=\frac{\varphi(n)}{4}=8$.

(b) Assume $f(x)$ to be primitive, and let $\alpha \in \mathbb{F}_{3^4} \setminus \mathbb{F}_3$ be a root of $f(x)$.
$\alpha^4 \equiv -\alpha + 1$.

$n = |\mathbb{F}_{3^4}^*| = 80 = 2^4 \cdot 5$.

$\alpha^2 \not\equiv 1$, ok
$\alpha^4 \equiv -\alpha + 1 \not\equiv 1$, ok
$\alpha^5 \equiv -\alpha^2 + \alpha \not\equiv 1$, ok
$\alpha^8 \equiv \alpha^5 \cdot \alpha^3 \equiv -\alpha^5 + \alpha^4 \equiv \alpha^2 - \alpha - \alpha + 1 \equiv \alpha^2 + \alpha + 1 \not\equiv 1$, ok
$\alpha^{10} \equiv \alpha^8 \cdot \alpha^2 \equiv \alpha^4 + \alpha^3 + \alpha^2 \equiv \alpha^3 + \alpha^2 - \alpha + 1 \not\equiv 1$, ok
$\alpha^{16} \equiv (\alpha^8)^2 \equiv -\alpha^3 + \alpha - 1 \not\equiv 1$, ok
$\alpha^{20} \equiv \alpha^{16} \cdot \alpha^4 \equiv \cdots \equiv -\alpha^3 - \alpha^2 + \alpha \not\equiv 1$, ok
$\alpha^{40} \equiv (\alpha^{20})^2 \equiv \cdots \equiv -1 \not\equiv 1$, ok

α is a generator (i.e., a primitive element), therefore $f(x) \in \mathbb{F}_3[x]$ is a primitive polynomial for $\mathbb{F}_{3^4}$.

Question 5 [12 pts]

(a) Apply the Pollard's ρ method to factorize the RSA modulus $n = p \cdot q = 1537$.
(As a backup alternative, apply a "trivial division" strategy).

(b) Choose an admissible public exponent e between the values $e{=}9_{\texttt{dec}}$ and $e{=}13_{\texttt{dec}}$ and compute the value of the corresponding RSA private key $k_{priv}{=}(p, q, \varphi(n), d)$. Show every step of the computation.

(c) Sign the message $m{=}333_{\texttt{dec}}{\in}\mathbb{Z}_n$ (without employing any padding scheme) through applying the CRT. Describe each step of the procedure.

(d) A system administrator wants to set up the RSA cryptosystem in a network including U users, and for doing so he decides to generate a pool of P prime numbers each 2048 bits long and keeping $1{<}P{<}\frac{U^2}{2}$. Subsequently, he draws pairs of numbers from such a pool to set up the public moduli. The public exponent is assumed to be the same for every user and equal to $e{=}2^{16}+1$, while the secret parameters are computed knowing the factorization of each modulus.

Discuss whether the aforementioned procedure is secure or not, justifying your answer.

(e) Assume to work into the Montgomery domain: $(\widetilde{\mathbb{Z}}_N, +, \times),\ N{=}22$

- Show the definition of the Montgomery Multiplication and explain the advantages of applying it for implementing the RSA cryptosystem.
- Compute the Montgomery multiplication $C{=}A{\times}B \bmod N$, where $A{=}19_{\text{dec}}$ and $B{=}16_{\text{dec}}$ are values in the Montgomery domain, assuming a binary encoding of the operands

Solution:

(a) see lectures... $p = 29,\ q = 53$

(b) $\varphi(n){=}28 \cdot 52{=}1456{=}2^4 \cdot 7 \cdot 13$, $gcd(\varphi(n), 9){=}1$, and $gcd(\varphi(n), 13){\neq}1$, $\Rightarrow e{=}9{\in}\mathbb{Z}^*_{\varphi(n)}$
$d{=}e^{\varphi(\varphi(n))-1} \bmod \varphi(n){\equiv}_{1456}9^{576-1}{\equiv}_{1456}\ 9^{575}{\equiv}_{1456}9^{\langle 1000111111\rangle_2}{\equiv}_{1456}\cdots{\equiv}_{1456}809.$

(c) $s \equiv_{1537} 333^{809}$

$$\begin{cases} x_p \equiv_{29} (333 \bmod 29)^{809 \bmod 28} \equiv_{29} 14^{25} \equiv_{29} 21 \\ x_q \equiv_{53} (333 \bmod 53)^{809 \bmod 52} \equiv_{53} 15^{29} \equiv_{53} 36 \end{cases}$$

$q^{-1} \bmod p = 53^{-1} \equiv_{29} 24^{-1} \equiv 29 \cdots \equiv_{29} 23.$
$p^{-1} \bmod q = 29^{-1} \equiv_{53} 29^{-1} \equiv 53 \cdots \equiv_{53} 11.$
$s \equiv_n (q^{-1} \bmod p) \cdot q \cdot x_p + (p^{-1} \bmod q) \cdot p \cdot x_q \equiv_{1537} 23 \cdot 53 \cdot 21 + 11 \cdot 29 \cdot 36 \equiv_{1537} 195.$

(d) see lectures ...

(e)
- see lectures...
- $N = 22$ is an even number, the Montgomery radix R cannot be a power of 2, as the Montgomery product requires that N and R be relatively prime!

  However, it is possible to partition an RSA computation (as well as a modular multiplication) into two modular operations with respect to the prime factors of the user's modulus. Let N be factored such that $N = q \cdot 2^l$, with q odd.
  By the application of the CRT the computation $c \equiv_N m^e$ is split up into
  $$\begin{cases} c \equiv_q c_1 \equiv_q m^e \\ c \equiv_{2^l} c_2 \equiv_{2^l} m^e \end{cases}$$
  and the result obtained with the CRT formula: $c \equiv_N c_1 + (c_2 - c_1)(q^{-1} \bmod 2^l)$.

The computation of c_1 can be performed using the Montgomery algorithm since q is odd, while the computation of c_2 can be performed even more easily, since it involves arithmetic modulo 2^l.