DD2448 Foundations of cryptography

Homework I krypto17

Persnr	Name	Email
951113-7656	Artem Los	arteml@kth.se
760513-7319	Mati Rachamim	rachamim@kth.se
-	- -	-

1 (2T) A cryptographic scheme that is computationally secure (but not perfectly secure) can always be broken given enough time. To prove unconditionally that some scheme is computationally secure would thus require proving a lower bound on the time needed to break the scheme; specifficaly it would be necessary to prove that the scheme cannot be broken by any polynomial-time algorithm. Unfortunately, the current state of affairs is such that we are unable to prove lower bounds of this type. In fact, an unconditional proof of security for any modern encryption scheme would require breakthrough results in complexity theory that seem far out of reach today. Instead of blithely assuming that a given cryptographic construction is secure, our strategy will be to assume that some low-level problem is hard to solve, and to then prove that the construction in question is secure given this assumption.
The proof that a given construction is secure as long as some underlying problem is hard generally proceeds by presenting an explicit reduction showing how to convert any efficient adversary (Any kind of an attacking algorythem.) A that succeeds in "breaking" the construction with non-negligible probability into an efficient algorithm $\tilde{A}$ that succeeds in solving the problem that was assumed to be hard.
We begin with an assumption that some problem X cannot be solved (in some precisely-defined sense) by any polynomial-time algorithm except with negligible probability. We want to prove that some cryptographic construction Π is secure (again, in some sense that is precisely-defined). To do this:

1. Fix some efficient (i.e., probabilistic polynomial-time) adversary A attacking Π. Denote this adversary's success probability by $\varepsilon(n)$.
2. Construct an efficient adversary $\tilde{A}$ that attempts to solve problem X using adversary A as a sub-routine. An important point here is that $\tilde{A}$ knows nothing about "how" A works; the only thing $\tilde{A}$ knows is that A is expecting to attack Π. So, given some input instance x of problem X, our algorithm $\tilde{A}$ will simulate for A an execution of Π such that:
   - As far as A can tell, it is interacting with Π. More formally, the view of A when it is run as a sub-routine by $\tilde{A}$ should be distributed identically to (or at least close to) the view of A when it interacts with Π itself.
   - Furthermore, if A succeeds in "breaking" the execution of Π that is being simulated by $\tilde{A}$, this should allow $\tilde{A}$ to solve the instance x it was given, at least with inverse polynomial probability $\frac{1}{p(n)}$.
3. The previous section if $\varepsilon(n)$ is not negligible, then $\tilde{A}$ solves problem X with non-negligible probability $\frac{\varepsilon(n)}{p(n)}$. Since $\tilde{A}$ is efficient, and runs the ppt adversary $\tilde{A}$ as a sub-routine, this implies an efficient algorithm solving X with non-negligible probability, contradicting the initial assumption.

4. We conclude that, given the assumption regarding X, no efficient algorithm A can succeed with probability $\varepsilon(n)$ that is not negligible $\Longrightarrow \Pi$ is computationally secure.

2 (2T) In computational complexity theory, P, also known as PTIME or DTIME($n^{o(1)}$), is a fundamental complexity class. It contains all **decision problems** that can be solved by a deterministic Turing machine using a polynomial amount of computation time, or polynomial time.
Cobham's thesis holds that P is the class of computational problems that are "efficiently solvable" or "tractable"; in practice, some problems not known to be in P have practical solutions, and some that are in P do not, but this is a useful rule of thumb.
An important characteristic of cryptography algorithm is that it deals with decision problems. A decision problem is any arbitrary yes-or-no question on an infinite set of inputs. Because of this, it is traditional to define the decision problem equivalently as: the set of possible inputs together with the set of inputs for which the problem returns yes.
These inputs can be natural numbers, but may also be values of some other kind, such as strings over the binary alphabet 0,1 or over some other finite set of symbols. The subset of strings for which the problem returns "yes" is a formal language, and often decision problems are defined in this way as formal languages.
A classic example of a decidable decision problem is the set of prime numbers. It is possible to effectively decide whether a given natural number is prime by different algorithms.
Unlike decision problems, for which there is only one correct answer for each input, optimization problems are concerned with finding the best answer to a particular input. Optimization problems arise naturally in many applications, such as the traveling salesman problem and many questions in linear programming. There are standard techniques for transforming function and optimization problems into decision problems. For example, in the **traveling salesman problem**, the optimization problem is to produce a tour with minimal weight. The associated decision problem is: for each N, to decide whether the graph has any tour with weight less than N. By repeatedly answering the decision problem, it is possible to find the minimal weight of a tour. As those problems have many different way to answer a question and could be that like the case above of the **traveling salesman problem** there will be more than one solution to the optimization problem (Different tour for example .) a lot more calculations are needed and hence using a polinomial complexity algorithm to check each of the options can be very time consuming .

3 (3T) If $\exists\, c, n_0 > 0 \mid f(n) > n^{-c}\; \forall\, n > n_0$ then we could choose $c^* \mid c^* > c \rightarrow n^{-c^*} < n^{-c}$ and hence $f(n) > n^{-c^*}$
By definition of Negligible function a function $f(n)$ is negligible $\iff \forall c, \exists n_0 | f(n) < n^{-c}\; \forall\, n > n_0$ but we have shown that it is not the case for each c^* we have chosen so $f(n)$ is ***not negligible***

4 (2T) $3, 4$

5 (10I) Artem's solution (`1812179`)

6

6a (1T) Yes, it's possible. It's n bits of entropy. Every bit is not biased ($Pr = \frac{1}{2}$), so we can use the formula for entropy:

$$H(Y) = -\sum_{x \in X} \Pr[x] \log_2(\Pr[x]) = n$$

6b (1T) NOT SOLVED

6c (1T) NOT SOLVED

6d (1T) Both X, S, T can get values $0 \dots q-1$, so we have q values. Each of have the probability of $\frac{1}{q}$ if we assume that they are uniformly distributed. Otherwise we cannot know anything. In case of the former case,

$$H(Y) = -\sum_{x \in X} \Pr[x] \log_2(\Pr[x]) = 4 \times \frac{1}{q} \log_2 \left(\frac{1}{q}\right) = \log_2(q) \tag{1}$$

Hence $\log_2(q)$ bits of entropy.

6e (2T) We should note that becuase we have a data known to us some of the bits will be removed from our Summation : A_0 and A_1 to $A_{\lfloor (i-1)/2 \rfloor}$ ... hence our summation in the antrophy will start from $A_{\lfloor (i-1)/2 \rfloor + 1}$

6f (2T) NOT SOLVED

6g (2T) NOT SOLVED

7 (6T) The Hill cipher can be a victim to different attacks as described in the paper of *Khazaei and Ahmadi* .
We will first divide them into two main types of attacks KPA and COA.

- **KPA**: Hill can be easily broken in the standard KPA model.Suppose that the attacker knows **d** linearly independent blocks of plaintext ,(P_{i_1},P_{i_2},...,P_{i_d}) and the corresponding ciphertext blocks,(C_{i_1},C_{i_2},...,C_{i_d}). All block pairs (C_{i_j},C_{i_j}) can be collected from one single plaintext/ciphertext pair or from multiple plaintext/ciphertext pairs. $U = (P^T_{i_1}, P^T_{i_2}, ..., P^T_{i_d})^T$ and $W = (C^T_{i_1}, C^T_{i_2}, ..., C^T_{i_d})^T$ and easily calculate the corresponding key matrix as $K = U^{-1}W$.Because of linear independence of the matrix U rows, the invertibility of this matrix is ensured.

- **COA** :Although Hill is easily broken with KPA, there is no reported attack faster than brute-force attack in COA model, with the assumption that the plaintext is an English text. The existing redundancy of English text can be used to mount COA on Hill, e.g, using a brute-force attack. In this case, we have :
$|K| \approx 26^{d^2}, |C| = 26$ and $|H| = 1.0 \Rightarrow |K| \approx 0.787$
According to the The unicity distance theorm the attack can determine the secret key almost uniquely if the ciphertext is of length at least: $\approx 1.27d^2$.

Now we will describe better COA methods than the brute force one presented above :

- ***COA on Hill using monograms***:

  - **Brute-force attack on Hill**
    For a long enough ciphertext and for a well-chosen threshold, all the $d!$ matrices derived from the correct key matrix are the only representative keys. We conclude that in order to find all the representative keys, one can exhaust all the $O(26^{d^2}/d!)$ which are equivalent in terms of column permutation ? instead of a brute-force attack over all possible matrices. Let us discuss the unicity distance of the attack, i.e., the minimum ciphertext length to determine the $d!$ correct representative keys almost uniquely.
    Following the calculation presented in the paper we can see that if the ciphertext length is about $\approx 8.96^{d^2}$, then thesecret matrix is almost uniquely determined up to an unknown permutation over its columns. he bigram frequencies can then be used to distinguish the correct permutation of a correct representative key with computational complexity of $O(d^2)$,ignorable in comparison with the overall exponential complexity of the attack. The total complexity of the attack $O(d^2 26^{d^2}/\log d)$
  - **A divide-and-conquer attack on Hill**
    Divide-and-conquer technique can be used to transform the brute-force attack into a new attack with high improvement in computational complexity. The full algorithm can be read in the above paper we would just note now that the number of decrypted letters for almost uniquely determining a column,n fact all the d columns, of the decryption matrix can be calculated as $\approx 8.96d^2$
    The na?ve implemen- tation of the attack has a computational complexity of $O(d^2 26^{d^2})$
    By eliminating repeated calculations, we present an improved attack in Algorithm 1 with com- plexity of $O(d26^{d^2})$.

- **CRT based divide-and-conquer attack on Hill**
  Based on the following Observation, the computational complexity of the divide- and-conquer attack can be improved using the Chinese Reminder Theorem (CRT).

Observation ($Z_{13} and Z_2$ entropy) Let Xdenote a random variable over Z_{26} with the probability distribution of that of English letters. The Z_{13} entropy of English monograms is defined to be the entropy of the random variable ($X mod 13$). Similarly, we define the Z_2 entropy. A simple calculation shows that $Z_{13} and Z_2$ entropy of English monograms are 3.4052 and 0.9865, respectively. The corresponding redundancies are then 0.0798 and 0.0135.
With this observation, the same procedure for divide-and-conquer attack can be done in order to find the columns of the decryption key matrix modulo 2 and 13. To be more precise, for example, in order to find the decryption key matrix columns modulo 13, the ciphertext is first reduced modulo 13. Then all possible $13^d - 1$ values for a column modulo 13 are tried using the brute-force attack. Finally, the best dcandidates for the probable columns modulo 13 are identified. The minimum ciphertext length required to almost uniquely find a representative decryption key matrix modulo 13 can be approximated as $\approx 12.5d^2$
To find a representative key modulo 26, the attack can be devised in two different ways using the CRT. Although the computational complexities are the same, the required ciphertext lengths for a successful attack are different

- **Combinational attack.**
  Perform the divide-and-conquer attack modulo 13 and 2, and attain a representative decryption key matrix over $Z_{13} and over Z_2$.The computational complexity of this method is $O(d13^d)$ and the needed cyphertext length is $\approx 74d^2$

- **Lifting attack.**
  Perform the divide-and-conquer attack over modulo 13 only, and attain a representative key over Z_{13} Then for each column j of the matrix do the following steps to lift the Z_{13} representative key into one over Z_{26}:
  1. combine all the $2^d - 1$ non-zero vectors over Z_2 with the j?th column and compute the corresponding vectors overZ_{26} using the CRT.
  2. calculate the IML for each one and choose the vector with the largest index as the j?th column of the representative key matrix over Z_{26}.The computational complexity of this method is $O(d13^d)$ and the needed cyphertext length is $\approx 12.5d^2$

8 Our answer is based on `https://eprint.iacr.org/2012/359.pdf`.

8a (1T) Both *uniform* and *non-uniform* adversaries refer to the complexity of the algorithm (eg. used by the adversary). In the uniform case, the algorithm has a fixed set of instructions independent of the input. In other words, the same instructions are used for all inputs. In the non-uniform case, the algorithm can be thought of having different set of instructions for each input.

Any uniform algorithm is a non-uniform algorithm. When proving something, we can either use the uniform or the non-uniform model. The non-uniform proofs tend to be simpler.

Another view of a non-uniform algorithm is that it has different kinds of *advice* on how to solve the problem for a given input. This advice needs to exist mathematically but there does not have to be a way to find it computationally.

8b (2T) There is a difference between the choice of perspective. If we can prove a result in the uniform model, it also true in the non-uniform model. For example (from source), if we have the uniform theorem "If the RSA is problem is intractable, then protocol P is secure against certain type of adversaries.". Therefore, we essentially get two theorems at the cost of one, i.e. both "If RSA problem cannot be solved by a uniform algorithm, then protocol P cannot be attacked by a uniform adversary" and "If RSA problem cannot be solved by a non-uniform algorithm, then protocol P cannot be attacked by a non-uniform adversary. Only one proof is enough in the uniform model.

The difference is that the non-uniform model tends to be based on theoretical results, for example that something is secure because it reduces to an intractable problem. However, how to do we assess the level of intractability? The uniform approach allows us to assess intractability to some extent by using cryptanalytic algorithms, for example.

The uniform model is more practical: eg. we show that a problem is intractable and provide some evidence that it's difficult computationally (for example, factoring 2048 bit numbers). This is a stronger statement. A non-uniform approach can, for instance, show that a problem is intractable in theory, but that does not mean that there is no practical method to solve/approximate it.

9

9a (2T) NOT SOLVED

9b (4T) NOT SOLVED

9c (10T) NOT SOLVED

10 (4I) Artem's solution (`1774841`)

11 (3I) Artem's solution (`1776482`)

12 (5I) Artem's solution (`1781044`)

13 (2I) Artem's solution (`1812199`)

14 (4T) A *side channel attack* is a way to obtain secret parameters using information from the physical implementation of the cryptosystem, for example *cache attack* or *timing attack*. The aim is to exploit the weaknesses in the implementation rather than brute force or use theoretical weaknesses.

- **Simple Power Analysis** - The idea is to exploit the fact that different operations consume different amount of power. An example is the *exponentiation-by-squaring* algorithm in RSA, which is used during decryption. When the bit in the key is equal to 1, an extra multiplication will be triggered. This can be observed and thus the secret parameter (the key) can be retrieved (extra multiplication consumes more power).[1]
- **Differential Power Analysis** - This attack is applied in cases where a crypto system has many iterations (such as AES-128). The idea is similar to simple power analysis, except that we don't try to draw a direct relationship between the secret and the power consumption. Instead, we infer the secret data statistically.[2]
- **Conventional Timing** - The idea is to directly measure the execution time of the algorithm to find the secret parameters. For example, this is quite clear in RSA, where the multiplication operation will reveal the exponent based on time.[3]
- **Cache Timing** - The idea is to exploit the relation ship between cache hits and misses to find the secret parameter. An example where this can be used is AES.[4]
- **Conventional Fault Analysis** - The idea is to analyise results of faulty encryptions. For example, in RSA signature verification method, we can find the factorisation of n by analysing the difference between the correct signature and the signature where a fault has been introduced. [5]

15 For reference, the improvement was suggested by Gueron S. (University of Haifa) and Krasnov V. (Inter Corporation), see `https://eprint.iacr.org/2013/816.pdf`.

15a (1T) File path: `/openssl/crypto/ec/ecp_nistz256.c`.

15b (1T) Issue 4621 in OpenSSL *nistz256 point addition check for a = +/-b doesn't work for unreduced values.* It is already fixed. This could have potentially be exploited by attackers.

15c (3T) It makes modular multiplication and calculation of multiplicative inverse constant-time. The idea is to use n algorithm based on Femat's little theorem rather than Extended Euclidean algorithm.

16

16a (3T) The curves in question are called *Edwards curves*. The main idea behind them is that both addition and doubling have explicit formulas, allowing them to be computed efficiently. Edwards curves can be converted to Montgomery form (i.e. curves that support simple scalar multiplication method)

[1] `http://www.cs.uu.nl/docs/vakken/b3sec/Proj13/SideChannel.pdf`, Last used 2017-04-21
[2] Ibid
[3] Ibid
[4] Ibid
[5] Ibid

16b (2T) Point multiplication is faster in Edwards curves (that support Montgomery ladder) and constant time, whereas this is not the case with standard Weierstrass form multiplication. Although speed is one of the advantages, they also offer more protection against side-channel attacks (doubling occurs at no extra cost). Many elliptic curves are also birationally equivalent to Edwards curves (where addition works for all pairs of curve points, including inverses and neutral element) [6].

17

17a (2T) NOT SOLVED

17b (2T) NOT SOLVED

18

18a (2T) We know that Z_p^* is isomorphic to Z_q, so we can always find a function that maps between these two.

18b (1T) Z_q os sparse, i.e. i.e. not that representative in Z_p^*. We can use Pollads algorithm. Ask enough of questions (note the sparsity)

18c (1T) Increase the size of the groups.

19

19a (7T) NOT SOLVED

19b (3T) NOT SOLVED

[6]`http://cr.yp.to/newelliptic/newelliptic-20070906.pdf`, Last used 2017-04-21