Cryptanalysis

- 2024-Spring -

Ji, Yong-Hyeon

A document presented for
the Cryptanalysis

Department of Information Security, Cryptology, and Mathematics
College of Science and Technology
Kookmin University

April 19, 2024

Contents

Chapter 1

Midterm

Test

Python Example

How to calculate a factorial

```
def factorial(n):
product=1                    # Start with 1
for k in range(2,n+1):       # For each k = 2,3,...,n,
product*=k               # Multiply by k
return product
```

Terminal Example

```
user@host:~$ echo "Hello, World!"
Hello, World!
```

Listing 1.1: Simulating a terminal command

1.1 Time Memory Trade Off (TMTO) Attack

A TMTO attack is typically described in the context of finding the secret key k used in a cryptographic function f. The function f is assumed to be a block cipher or a cryptographic hash function.

Setup

Consider a cryptographic function $f : \mathcal{K} \times \mathcal{M} \to C$, where $\mathcal{K}$ is the key space, $\mathcal{M}$ is the message space and C is the cipher space. The goal is to invert f given $f(k)$, i.e., to find k when $f(k)$ is known.

Precomputation Phase

In the precomputation phase, a series of computations are performed to create a trade-off between the computation time and memory usage:

1. Select a subset of keys $\{k_1, k_2, \ldots, k_t\} \subset \mathcal{K}$.
2. Compute $f(k_i)$ for each k_i.
3. Store the pairs $(k_i, f(k_i))$ in a table called the **precomputed table**.

This table is used to accelerate the recovery of k by storing potential outputs and their corresponding inputs.

Recovery Phase

Given a ciphertext c, the attacker attempts to find k such that $f(k) = c$:

1. For each potential key k', compute $f(k')$.
2. Check if $f(k')$ exists in the precomputed table.
3. If a match is found, i.e., $f(k') = f(k_i)$ for some i, retrieve k_i.

Complexity Analysis

The effectiveness of a TMTO attack depends on the sizes of the key space $\mathcal{K}$, the cipher space C, and the table:

- **Memory Requirement:** Proportional to the number of entries t in the table.
- **Time Complexity:** Proportional to $\frac{|\mathcal{K}|}{t}$, assuming uniform distribution and independent choices of k_i.

Example: Hellman's TMTO

Hellman's approach involves structuring the precomputed table in chains where each chain starts from a randomly chosen initial value k_0 and is constructed as follows:

$$\begin{aligned} k_1 &= f(k_0), \\ k_2 &= f(f(k_0)), \\ &\vdots \\ k_t &= f^{(t)}(k_0), \end{aligned}$$

where $f^{(t)}$ denotes the t-th application of f. Only k_0 and k_t are stored, reducing memory usage but requiring more time in the recovery phase to reconstruct chains.