Secure and privacy preserving pattern matching in distributed cloud-based data storage

Vladimir Oleshchuk
Department of Information and Communication Technology, University of Agder, Norway
CTIF-CWI meeting, Copenhagen, October 24, 2011
Outline

- Problem description
- Pattern matching problem
- Current approach and challenges
- Distributed pattern matches
- Security analysis
- Concluding remarks
Problem:

How to provide cloud storage with searching functionality that are secure and privacy preserving while storage providers are not trusted service?
Pattern matching problem

• Pattern $P=p_1,...,p_m$ matches an input $d=t_1 \ t_2 \ ... \ t_n$ if $p_1p_2...p_m$ occurs as a subsequence of $d$, that is, if $d=u \ p_1p_2...p_m \ v$ for some $u$, $v$ from $A^*$

• Example:
  ✓ $d=$ abracadabra
  ✓ $P=$ ada
  ✓ $P$ matches $d$: abracadabra
Current approach

d = t_1 \cdot t_2 \ldots t_n

P = p_1, \ldots, p_m
Challenges:

① Want support for some basic functionalities such as, for example, searching in the storage (pattern matching).

② Encryption of data in the storage makes them not accessible to the provider but also searching not available (without preliminary decryption).

③ In case of encrypted storage in the most cases it will be impossible or very difficult to provide it on encrypted data without encrypting them before such operations.
Alternative approach
Alternative approach

• Distribute data among \( k \) untrusted storage providers in such a way that required functionality can be performed in concurrently by all storage providers and users can combine results to find what is the search result.

• The data are distributed between storage providers such that one needs cooperation of storage providers to restore data.

• How can we do it?
Pattern matching case

- Assume that there are data \( d \) that are a sequence in a finite alphabet, \( d=t_1t_2...t_k \) where \( t_i \) are letters from finite alphabet.
- \( d_a \) - projection of text \( d \) with respect to the letter ‘a’ represents positions of ‘a’ in the \( d \).
- Example:
  - Text: \( d=\text{abracadabra} \)
  - Projections:
    - \( d_a=02112 \) since \( a**a*a*a**a \)
    - \( d_b=16 \) since \( *b******b** \)
    - \( d_c=4 \) since \( *****c****** \)
    - \( d_r=26 \) since \( **r********r* \)
Permutations

• Let $\pi_a$ be a permutation applied to $d_a: (1,2,3,\ldots,k) \rightarrow (n_1,n_2,n_3,\ldots,n_k)$.

• Let $\pi_a(d_a)$ be the result of application of $\pi_a$ to $d_a$.

• Suppose we want to find occurrences of subsequence (pattern) $P$ in $d$.

• $P_a$ occurs in $d_a$ if and only if $\pi_a(P_a)$ occurs in $\pi_a(d_a)$. 
Distributed pattern matching

\[ \pi_a(d_a) \]
\[ \pi_b(d_b) \]
\[ \pi_c(d_c) \]
\[ \pi_r(d_r) \]

\[ \Pi_a(P_a) \]
\[ \Pi_b(P_b) \]
\[ \Pi_c(P_c) \]
\[ \Pi_r(P_r) \]
Security analysis

• To protect $d_a$ from potential guessing (there is only 26 letters in English alphabet - one can easily try each!)

• Possibly it is not difficult to guess 'a' based on given distances between letters. (May be frequency analysis is still possible here since you have number of each letter available.)

• However, knowing which letter is projected without knowing letter's position may not help much.

• So we will change letters positions by applying permutation known only by data owner but not storage provider.

• It will be applied distinct/different permutations to distinct storage providers.
CONCLUDING REMARKS

1. We propose a new approach that provides distributed secure and privacy preserving data storage in the cloud even for colluding providers:
   ✓ Storage providers cannot access data (data confidentiality)
   ✓ Storage providers cannot see what users/owners are searching for (user privacy)

2. Approach provides secure and privacy preserving pattern matching with sub-linear time performance.
Thank you!

Questions?