computing@tanet.edu.te.ua
www.tanet.edu.te.ua/computing

ISSN 1727-6209
International Scientific Journal of Computing

PRIVACY PRESERVING PATTERN MATCHING ON SEQUENCES OF EVENTS

Vladimir A. Oleshchuk

Communication and System Security Group, Agder University College
Grooseveien 36, N-4876 Grimstad, Norway
email:vladimir.oleshchuk@hia.no

Abstract: *We propose to use pattern matching on data streams from sensors in order to monitor and detect events of interest. We study a privacy preserving pattern matching problem where patterns are specified as sequences of constraints on input elements. We propose a new privacy preserving pattern matching algorithm over an infinite alphabet A where a pattern P is given as a sequence $\{p_{i_1}, p_{i_2}, ..., p_{i_m}\}$ of predicates p_{i_j} defined on A. The algorithm addresses the following problem: given a pattern P and an input sequence t, find privately all positions i in t where P matches t. The privacy preserving in the context of this paper means that sensor measurements will be evaluated as predicates $p_i(e_j)$ privately, that is, sensors will not need to disclose the measurements $x_i^{(j)}, x_2^{(j)}, ..., x_n^{(j)}$ to the evaluator.*

Keywords: *Pattern matching, string matching, privacy preserving, sensor networks.*

1. INTRODUCTION

Generally, sensor networks are designed to support distributed interaction with the physical environment through measuring and aggregation of data in order to create dynamic global view. These tasks create various streams of measurement data within such networks. However, the data streams from sensors, further referred as sequences of events, are only useful if they can be used to monitor and detect events of interest [6], [11].

In context of this paper we abstract from inessential physical structure of sensor networks and consider a simplified model representing a set of sensors connected to the base station they send measurements to.

We assume that there are n sensors, denoted $s_1, s_2, ..., s_n$, and connected to the base station B. If $x_i^{(t)}$ denotes a measurement value received at a time slot t from sensor s_i then n-tuple $\langle x_1^{(t)}, x_2^{(t)}, ..., x_n^{(t)} \rangle$ is an event e_t that represents states of monitored sensors at a time slot t.

In this paper we analyze sequences of events $e_1, e_2, ..., e_n, ...$ with the main goal to detect subsequences of events that may indicate some predefine activities of interest, for example, fire development in some areas of the monitored building.

Detecting occurrences of a subsequence pattern in sequence of events is an example of more general pattern matching problem. It is an important component of many applications, including real time monitoring and events detection in manufacturing processes by examining noisy sensor data [8].

Formally, the pattern matching means finding one, or more generally, all occurrences of a pattern inside sequential raw data. Raw data can be seen as a sequence over some finite or even infinite alphabet. However in many cases measurements are continuous by their nature. For example, in the case of sensor networks used for environmental, health and security monitoring of buildings, where measurements are temperature, humidity, sound level etc. They need to be discretized in some way to be presented in computers due to the finite presentation restrictions of computer representations. Discretizing is one-to-many process, that is, the same continuous measurement from real world may correspond to many discretized presentations due to, for example, limited equipment resolution, noises or precise measurement problems.

Since digitized presentations are approximate presentations of continuous objects, the traditional string matching algorithms developed for discrete applications areas, for example texts, are either unusable or inefficient [2], [7]. Therefore, it is important to develop new algorithms that are fast, require little memory and limited buffering, and can operate in real-time on digitized presentations of continuous measurements.

Another important aspect of monitoring sequences of events as described above is their sensitivity with respect to privacy and security. Some measurements are very sensitive by their nature since they represent private information about monitored objects. However, we still need to access measurements data in order to perform monitoring and events detection.

In this paper we study privacy preserving aspects of pattern matching of in sequences of events. The paper is organized as follows. In Section 2 we provide necessary definitions and notation. The general pattern matching algorithm is presented in Section 3. The privacy preserving protocol algorithm and its complexity analysis is given in Section 4. Finally, concluding remarks are made in the last section.

2. PATTERN MATCHING ON SEQUENCES

When row data are texts over finite alphabets the pattern matching problem is known as a string-matching problem. Many different variations of string-matching problems have been studied. Efficient solutions proposed in the literature [1]-[3] are based either on the use of automata or on combinatorial properties of strings over finite alphabets. These problems and proposed solutions are heavily depend on finiteness property of underlying alphabets, and these ideas cannot be applied without essential modifications to solve the similar problems in continuous setting where alphabets are infinite or very large and precise matching, as in digitized presentations, often unnecessarily or even impossible.

In this paper we deal with pattern matching problems applied to sequences of events representing n-tuples of unconventional data as, for example, digitized measurement data [9], [10]. Since under digitization both patterns and row data can be mapped into many digitized patterns and many digitized row sequences (as explained previously) it is unlikely that an exact match can always be achieved. Therefore we consider an on-line algorithm that solves the problem in general, and then modify it to develop a more efficient privacy-preserving version for some classes of patterns.

We consider events $e_i, i = 1,2,...$ as elements of an infinite set E, that is, $E = \{e_1, e_2, ...\}$. Let E^* denote the set of all finite-length sequences of elements from E. The length of a finite sequence $s_n = e_1 e_2 ... e_n$ is denoted as $|s_n|$, that is, $|s_n| = n$. We assume that $s_0 = \varepsilon$ where ε denotes the empty sequence. Let E^m denote the set of all sequences of length m formed using elements from E. We say that a sequence w is a prefix of a sequence t if $t = wy$ for some sequence $v \in E^*$. Similarly, we say that a sequence w is a suffix of a sequence t if $t = uw$ for some sequence $u \in E^*$. A sequence w is a subsequence of a sequence t if $t = uwv$ for some $u, v \in E^*$. Let $\Omega_E = \{p_1, p_2, ...\}$ be a set of predicates defined on E, that is, $p_i : E \to \{true, false\}$ for any $p_i \in \Omega_E$. A pattern P of length m is a sequence $\langle p_{i_1}, p_{i_2}, ..., p_{i\,m} \rangle$ of predicates from Ω_E. The pattern $P = \langle p_{i_1}, p_{i_2}, ..., p_{i\,m} \rangle$ represents a subset Q_P of sequences from A^m defined as

$$Q_P = \left\{ e_{i_1} e_{i_2} ... e_{i_m} \mid \mathop{\&}_{j=1}^{m} p_{i_j}\left(e_{i_j}\right) = true \right\}$$

where $e_{i_j} \in E$.

We say that a pattern $P = \langle p_{i_1}, p_{i_2}, ..., p_{i\,m} \rangle$ from Ω_E matches a sequence $s_n = e_1 e_2 ... e_n$ from E^*, if a sequence $y_1 y_2 ... y_m$ from Q_P occurs as a subsequence of s, that is, if $s = u y_1 y_2 ... y_m v$ for some $u, v \in E^*$. The pattern P occurs at position $k+1$ in sequence $s_n = e_1 e_2 ... e_n$ or matches s in position $k+1$ if

$$\mathop{\&}_{j=k+1}^{k+m} p_{i_j}\left(e_j\right) = true$$

where $0 \le k \le n - m$.

The pattern matching problem we deal with in this paper is as following:

Given a pattern P from Ω_E and a sequence of events s from E^*, find all positions in s where P matches s.

The efficient algorithm that solves the problem has been proposed in [9]. In the following section for the sake of clarity we give a short presentation of the algorithm. Readers can find the correctness proof and efficiency evaluation of the algorithm in [9].

3. PATTERN MATCHING ALGORITHM

In this section we present the main idea of the on-line algorithm for constraint-based pattern matching (more details can be found in [9]).

Let t_k be a string $a_1 a_2 ... a_k$ from A^*, and *P* be a pattern $\langle p_{i_1}, p_{i_2}, ..., p_{i\,m} \rangle$ from Ω_A. Let S_k^P denote a set of lengths of all prefixes of *P* matching some suffix of t_k, that is, $j \in S_k^P$ if and only if

$$p_{i_1}(a_{k-j+1}) \,\&\, p_{i_2}(a_{k-j+2}) \,\&\, \cdots \,\&\, p_{i_j}(a_k) = true$$

Therefore, *P* matches t_k if and only if $m \in S_j^P$ for some $j \le k$. Thus, if we want to perform pattern matching based on this idea, we have to construct a sequence of sets $S_0^P, S_1^P, \ldots, S_k^P, \ldots$ for a given pattern $P = \langle p_{i_1}, p_{i_2}, \ldots, p_{i\,m} \rangle$ and input sequence $t = a_1 a_2 ... a_k ...$. In the course of construction sets $S_0^P, S_1^P, \ldots, S_k^P, \ldots$ we check whether *m* is in S_k^P for some $k > 0$. Then, for any S_k^P such that $m \in S_k^P$ we report that *P* matches *t* in position $(k - m + 1)$.

The algorithm *MATCH*, based on the above idea, is presented in Fig. 1. The output of the algorithm *MATCH*(*t,P*) is the set of all positions of occurrences of *P* in *t*. The algorithm uses the procedure *UPDATE*(S^P, x, P) to construct S_i^P based on S_{i-1}^P and *x*, where *x* is a new input element of sequence *t*.

Input: a sequence of data *t* and a pattern $P = \langle p_{k_1}, p_{k_2}, ..., p_{k_m} \rangle$

Output: beginning positions of all matching patterns.

```
Algorithm MATCH(t, P)
1. $S \leftarrow \{0\}$;
2. $pos \leftarrow 0$;
3. while input is not empty do
       $x \leftarrow$ read next element from $t$;
       $pos \leftarrow pos + 1$;
       $S \leftarrow UPDATE(S, x, P)$;
       if $m \in S$ then
               pattern matching at $(pos - m + 1)$
       $S \leftarrow S \setminus \{m\}$
4. end
```

Fig. 1. Algorithm MATCH

Let us consider in details the algorithm *UPDATE* presented in Fig.2. The main purpose of *UPDATE* is to construct $S_0^P, S_1^P, \ldots, S_k^P, \ldots$ for a given pattern $P = \langle p_{i_1}, p_{i_2}, \ldots, p_{i_m} \rangle$ and input sequence $t = a_1 a_2 \cdots a_k \cdots$. The algorithm *UPDATE* is presented in Fig. 2. According to Fig. 2, *UPDATE* calculates S_{i+1}^P based on S_i^P, the next input element *x* and the length of prefix of *P* that has been already matched.

The difference between the algorithm *UPDATE* and a naive brute-force algorithm is that *UPDATE* uses results of previous matching steps to optimize future matching steps and in this way remove redundancy in computations.

Input: Set S^P such that $S^P \subseteq \{0, 1, \ldots, m-1\}$, pattern $P = \{p_{i_1}, p_{i_2}, \ldots, p_{i_m}\}$ from Ω_A, and *x* from *A*.

Output: Set S^P such that $S^P \subseteq \{0, 1, \ldots, m\}$

```
Algorithm UPDATE $(S^P, x, P)$
1. $S' \leftarrow \varnothing$
2. for each j from $S^P$ do
       $S^P \leftarrow S^P \setminus \{j\}$
       if $p_{i_{j+1}} = true$ then $S' \leftarrow S' \cup \{j+1\}$
3. $S^P \leftarrow S' \cup \{0\}$
4. return $S^P$
```

Fig. 2. Algorithm *UPDATE*

For each element *x* of input *t*, algorithm *UPDATE* considers only those predicates of the pattern $P = \langle p_{i_1}, p_{i_2}, \ldots, p_{i_m} \rangle$ that still may be a part of some occurrences of *P* in *t*. When an input of length *i* has been analyzed, *UPDATE* contains in S_i^P references on all predicates that need to be evaluated on the next input *x*, that is, $j \in S_i^P$ represents p_{i_j} from *P*.

However, the time complexity of *UPDATE* depends on properties of the pattern *P*, that is, both on complexity of evaluation of predicates from *P* and their interdependency. The interdependency presents the knowledge of how truth values for some predicates of *P* can be found without their explicit evaluation but inferred from truth values of some already evaluated predicates of *P*.

The time complexity of algorithm *MATCH* is linear of the length of input plus the time complexity of all *UPDATE*'s calls. (Detailed proof can be found in [9].)

4. PRIVACY PRESERVING MATCHING

In this section we show how algorithm from previous section can be modified to perform privacy-preserving detection of activities in sensor networks based on pattern matching over infinite alphabets.

As we can see from the pattern matching algorithm *MATCH* (Fig. 1), evaluation of predicates $p_i(e_j) = p_i(x_1^{(j)}, x_2^{(j)}, \dots, x_n^{(j)})$ is an essential part of the algorithm.

Privacy-preserving in the context of this paper means that the base station *B* conducts evaluation of predicates $p_i(e_j)$ on *private inputs* from sensors, that is, sensors will not need to disclose the measurements $(x_1^{(j)}, x_2^{(j)}, \dots, x_n^{(j)})$ to the base station. In addition, we want to protect the base station *B* from revealing to the sensors both the results of evaluations and the descriptions of predicates. The solution we propose in this paper is based on secure multi-party computation approach similar to approach described in [4], [5].

Our solution can be seen as a solution of well known Yao's Millionaire problem [15] that is formulated as a comparison of two private numbers in order to decide which is larger. However, in context of this work it is more convenient to see this problem as a simplified version of point inclusion problem, namely, point inclusion in an interval. We want to evaluate privately whether a number *x* is within a given interval $[a,b]$ or not.

In the case when only a finite number of elements can occur in the interval, the solution of Yao's Millionaire problem given in [15] can be used. However, in the case when the number of elements from $[a,b]$ is large that solution is not efficient in terms of communication complexity. Therefore we need to find a new more efficient solution that match better our problem settings.

We need to find a way to compute $x \in [a,b]$ without discloser sensor's value x to base station and without discloser base station interval $[a,b]$ to the sensor. It can be seen as a problem of private evaluation of function $f(x) = (x-a)(x-b)$ since it is easy to see that $x \in [a,b]$ if and only if $f(x) \le 0$.

In our further considerations we assume that Alice represents a sensor and Bob represents a base station.

We shall use a public-key cryptosystem with homomorphic property where encryption and decryption are denoted as $E(\bullet)$ and $D(\bullet)$ respectively. The homomorphic property means that there is an operation on encrypted data, denoted $\oplus$, that defines an addition on encrypted data (without intermediate decryption). That is, we assume that $E(x) \oplus E(y) = E(x+y)$. Many such systems have been proposed in the literature [12]-[14].

Further, since $E(x) \oplus E(y) = E(x+y)$, then

$$E(2x) = E(x+x) = E(x) \oplus E(x)$$

and

$$E(yx) = E\left(\underbrace{x + x + \cdots + x}_{y}\right) = \\ = \underbrace{E(x) \oplus E(x) \oplus \cdots \oplus E(x)}_{y}$$

Thus we can multiply encrypted data if one of the multipliers is known (available in unencrypted form). This property we will use in the protocol described later in this section.

To simplify further notation and make our protocol independent of particular selected homomorphic public-key cryptosystem we assume that there are operations $\oplus$ and $\otimes$ on encrypted data are defined as following:

$$E(x) \oplus E(y) = E(x+y)$$
$$E(x) \otimes E(y) = E(xy) = \underbrace{E(x) + E(x) + \cdots + E(x)}_{y}$$

The following protocol describes privacy preserving evaluation of predicates presented in the form $x \in [a,b]$ where x, a, b are integers.

Input: Alice has a homomorphic public key cryptosystem where *E* is an encryption function and *D* is a corresponding decryption function; Alice (a sensor) has a measurement value *x*; Bob (a base station) has an interval $[a,b]$.

Output: Bob learns whether $x \in [a,b]$ is true or false without learning *x*, and without revealing $[a,b]$ to Alice.

Protocol: Privacy preserving predicate evaluation.

- **Step 1:** Alice (sensor) generates a key pair for a homomorphic public key system and sends the public key to Bob (base station). (The corresponding encryption and decryption functions are denoted as $E(\bullet)$ and $D(\bullet)$).
- **Step 2:** Alice encrypts x and x^2 using her

public key, and sends $E(x)$ and $E(x^2)$ to Bob.

- **Step 3:** Bob generates pairs $(a_i, b_i), i = 1,2,...k$, and assumes that $a_0 = a$ and $b_0 = b$.
- **Step 4:** Bob computes $z_i = E(a_i b_i + x^2)$, $t_i = E((a_i + b_i)x)$ for all $i = 0,1,2,...,k$ (without learning x in) as following:
  1. Using the homomorphic property of Alice's public key cryptosystem Bob can see that $E(a_i b_i + x^2) = E(a_i b_i) \oplus E(x^2)$
  2. Since all a_i and b_i are known to Bob, he can compute $E((a_i + b_i)x) = E(a_i + b_i) \otimes E(x)$ as $E(\underbrace{x + x + \cdots + x}_{a_i + b_i}) =$ $\underbrace{E(x) \oplus E(x) \oplus \cdots \oplus E(x)}_{a_i + b_i}$
- **Step 5:** Bob generates $(t_0, z_0), (t_1, z_1), \ldots, (t_k, z_k)$ using the random permutation function π, getting $\pi((t_0, z_0), (t_1, z_1), ..., (t_k, z_k)) =$ $((t_{i_0}, z_{i_0}), (t_{i_1}, z_{i_1}), ..., (t_{i_k}, z_{i_k}))$, and sends it to Alice.
- **Step 6:** Alice evaluates $(a_{i_j} b_{i_j} + x^2) \geq (a_{i_j} + b_{i_j})x$ for $j = 0,1,...,k$ by using her private key, since $(a_{i_j} b_{i_j} + x^2) = D(t_{i_j})$ and $(a_{i_j} + b_{i_j})x = D(z_{i_j})$; Alice sends Boolean vector $(b_{i_0}, b_{i_1}, ..., b_{i_k})$ representing evaluation results to Bob, where b_{i_j} denotes evaluation result of $(a_{i_j} b_{i_j} + x^2) \geq (a_{i_j} + b_{i_j})x$.
- **Step 7:** Based on received from Alice evaluation vector and since Bob knows position j in the permutation π such that $a_0 = a_{i_j}$ and $b_0 = b_{i_j}$, Bob is able to decide based on b_{i_j} without knowing $f(x)$ whether $f(x) \geq 0$ or not.

Presented above protocol shows how we can evaluate privately predicate defined as point inclusion in an interval. Combining the protocol with the pattern matching algorithm presented in Section 3 gives a privacy preserving pattern matching algorithm.

The privacy of Bob's data is achieved by applying random permutation π. The permutation prevents Alice from learning a and b. According to the protocol, the Alice's data x never appears as a part of unencrypted data available to Bob. Therefore the privacy of Alice data is protected by strength of applied homomorphic public-key cryptosystem.

5. CONCLUSION

We have proposed a solution of privacy preserving event detection problem based on privacy preserving pattern matching algorithm. The algorithm can be used to implement monitoring events in sensor networks without violating privacy of objects under observation.

6. REFERENCES

[1] M. Crochemore and W. Rytter, *Text Algorithms*, Oxford, University Press, 1994.

[2] M. Crochemore and C. Hancart, *Pattern Matching in Strings*. In Handbook of Algorithms and Theory of Computation, M. Atallah, ed., CRC Press, Boca Raton, 1996.

[3] M. Crochemore and T. Lecroq, *"Pattern Matching and Text Compression Algorithms,"* In The Computer Science and Engineering Handbook, A. B. Tucker, ed., CRC Press, 1997.

[4] W. Du and M. J. Atallah, "Secure Multi-Party Computation Problems and Their Applications: A review and Open Problems," *Proc. of New Security Paradigms Workshop*, 2001, pp. 13-22.

[5] W. Du and Z. Zhan, "A practical Approach to Solve Secure Multi-Party Computation Problems," *Proc. of New security paradigms*, 2002, pp. 127-135.

[6] R. Gwadera, M. Atallah and W. Szpankowski, "Reliable Detection of Episodes in Events Sequences," *Proc. of the Third IEEE Intern.* Conf. on Data Mining (ICDM'03), 2003.

[7] D. E. Knuth, J. Morris, and V. Pratt, "Fast Pattern Matching in Strings," *SIAM Journ. on Comp.*, vol. 6, 1977, pp. 323-350.

[8] J.P. Morrill, "Distributed Recognition of Patterns in Time Series Data," *Comm. of the ACM*, vol. 41, 1998, pp. 45-51.

[9] V. A. Oleshchuk, "On-line Constraint-based Pattern Matching on Sequences," In Sequences and Their Applications – *Proc. of SETA '98*. C. Ding, T. Helleseth, H. Niederreiter, Eds., Springer-Verlag, 1999, pp. 330-342.

[10] V. A. Oleshchuk, "On-line Fuzzy Pattern Matching on Sequences.," *In Advances in Fuzzy Systems and Evolutionary Computation*. N.E. Mastorakis, Ed., 2001, pp. 144-149.

[11] D. Wagner, "Resilient aggregation in sensor networks," SASN '04: *Proc. of the 2nd ACM workshop on Security of ad hoc and sensor networks*, 2004, pp. 78-87.

[12] D. Naccache and J. Stern, "A New

Cryptosystem Based on Higher Residues," In *Proceedings of the 5th ACM Conf. on Computer and Communication Security*, 1998, pp.59-66.

[13] T. Okamoto and S. Uchiyama, "An Efficient Public-Key Cryptosystem as Secure as Factoring," *In Advanced in Cryptography - EUROCRYPT'98*, LNCS 1403, pp. 308-318, 1998.

[14] P. Paillier. "Public-Key Cryptosystems Based on Composite Degree Residuosity Classes," *In EUROCRYPT'99*, LNCS 1592, 1999, pp. 223-238.

[15] A.C. Yao, "Protocols for Secure Computations," *In Proc. of the 23th IEEE Symp. on Foundations of Computer Science*, 1982.

***Vladimir A. Oleshchuk**, received his MS degree in Applied Mathematics in 1981 and the PhD degree in Computer Science in 1988, both from the National Taras Shevchenko University, Kiev, Ukraine. From 1987 to 1992 he worked as assistant professor and then as associate professor at National Taras Shevchenko University. He is currently a Professor in the Department of Information and Communication Technology and the leader of Commutation and System Security group at Agder University College, Norway. His current research interests formal methods and information security with special attention to telecommunication systems.*