Algorithms of Information Security: Cryptographic Protocols and Malware

Faculty of Information Technology
Czech Technical University in Prague

November 24, 2022

Key transport without a priori shared keys

- Shamir's no-key algorithm is a key transport protocol which, using only symmetric techniques (although involving modular exponentiation), allows key establishment over an open channel without requiring either shared or public keys.
- Each party has only its own local symmetric key.
- The protocol provides protection from passive adversaries only; it does not provide authentication. It thus solves the same problem as basic Diffie-Hellman protocol – two parties sharing no a priori keying material end up with a shared secret key, secure against passive adversaries – although differences include that it uses three messages rather than two, and provides key transport

Shamir's no-key protocol

Algorithm 1 Shamir's no-key protocol

SUMMARY: users A and B exchange 3 messages over a public channel.

RESULT: secret K is transferred with privacy (but no authentication) from A to B.

1. *One-time setup (definition and publication of system parameters).*
   1. Select and publish for common use a prime p chosen such that computation of discrete logarithms modulo p is infeasible.
   2. A and B choose respective secret random numbers a, b, with $1 \leq a, b \leq p-2$, each coprime to $p-1$. They respectively compute a^{-1} and $b^{-1} \mod p-1$.

Shamir's no-key protocol

Algorithm 2 Shamir's no-key protocol

2. *Protocol messages.*

$$A \rightarrow B : K^a \bmod p \tag{1}$$

$$A \leftarrow B : (K^a)^b \bmod p \tag{2}$$

$$A \rightarrow B : (K^{ab})^{a^{-1}} \bmod p \tag{3}$$

Shamir's no-key protocol

Algorithm 2 Shamir's no-key protocol

3. *Protocol actions.* Perform the following steps for each shared key required.
   1. A chooses a random key K for transport to $B, 1 \leq K \leq p-1$. A computes $K^a \mod p$ and sends B message (1).
   2. B exponentiates (mod p) the received value by b, and sends A message (2).
   3. A exponentiates (mod p) the received value by $a^{-1} \mod p-1$, effectively "undoing" its previous exponentiation and yielding $K^b \mod p$. A sends the result to B as message (3).
   4. B exponentiates (mod p) the received value by $b^{-1} \mod p-1$, yielding the newly shared key $K \mod p$.

Shamir's no-key protocol

Example. Alice and Bob are using Shamir's no-key protocol to exchange a secret message. They agree to use the prime $p = 31337$ for their communication. Alice chooses the random number $a = 9999$ while Bob chooses $b = 1011$. Describe the communication between Alice and Bob if the Alice chooses the key $K = 3567$. Use the Extended Euclidean Algorithm to compute $a^{-1} \mod p - 1$.

Shamir's no-key protocol

Solution. We know that the Alice and Bod agree to use the prime $p = 31337$ for their communication. First, we check whether the numbers a, b satisfy $1 \leq a, b \leq p - 2$ and each is coprime to $p - 1$. Then we compute $a^{-1} \mod p - 1$. To compute $a^{-1} \mod p - 1$, we use the Extended Euclidean Algorithm.

$$\begin{aligned} 31336 &= 3 \cdot 9999 + 1339 \\ 9999 &= 7 \cdot 1339 + 626 \\ 1339 &= 2 \cdot 626 + 87 \\ 626 &= 7 \cdot 87 + 17 \\ 87 &= 5 \cdot 17 + 2 \\ 17 &= 8 \cdot 2 + 1. \end{aligned}$$

Shamir's no-key protocol

Therefore $\gcd(31336, 9999) = 1$. It is well known that if the $\gcd(a, b) = 1$ then there exist integers p and s so that:

$$p \cdot a + s \cdot b = 1.$$

By reversing the steps in the Euclidean Algorithm, it is possible to find these integers p and s. Then we compute the inverse of a :

$$\begin{aligned}
1 =& 17 - 8 \cdot 2 \\
=& 17 - 8 \cdot (87 - 5 \cdot 17) = 41 \cdot 17 - 8 \cdot 87 \\
=& 41 \cdot 626 - 295 \cdot 87 \\
=& 631 \cdot 626 - 295 \cdot 1339 \\
=& 631 \cdot 9999 - 4712 \cdot 1339 \\
=& 14767 \cdot 9999 - 4712 \cdot 31336.
\end{aligned}$$

So we have $a^{-1} \mod p - 1 = 14767$.

Shamir's no-key protocol

Protocol messages.

$$A \rightarrow B : K^a \bmod p = 3567^{9999} \bmod 31337 = 6399 \quad (1)$$

$$A \leftarrow B : (K^a)^b \bmod p = 6399^{1011} \bmod 31337 = 29872 \quad (2)$$

$$A \rightarrow B : (K^{ab})^{a^{-1}} \bmod p = 29872^{14767} \bmod 31337 = 24982 \quad (3)$$

k-nearest Neighbors (KNN) Classifier

- The k nearest neighbors (KNN) classifier is one of the most popular supervised learning methods.
- Let $T = \{(x_1, c_1), \ldots, (x_m, c_m)\}$ be the training set, where x_i is the training feature vector and c_i is its corresponding label.
- Let x_q be the test feature vector. Then we determine its label c_q as follows:
  1. Find the set $T' = \{(x_1, c_1), \ldots, (x_k, c_k)\}$ k nearest neighbors of x_q.
  2. Then, determine the class c_q of element x_q based on the majority vote of its nearest neighbors.

k-nearest Neighbors (KNN) Classifier

- A majority vote is defined as:

$$c_q = \arg\max_c \sum_{(x_i, c_i) \in T'} \delta(c, c_i), \qquad (1)$$

where c is the class label, c_i is the class label for the i-th neighbor among the k nearest neighbors of the test vector, and $\delta(c, c_i)$ has the value one if $c = c_i$ and zero otherwise.

Distance-weighted k-nearest neighbor WKNN

- Distance-weighted k-nearest neighbor WKNN was first introduced as an enhancement of KNN (in 1976).
- This extension is based on the idea that closer neighbors have more weight than neighbors that are far from the query (test) vector.
- KNN implicitly assumes that all k nearest neighbors are equally important in deciding the resulting classification, regardless of their distances to the queried feature.
- In WKNN, the nearest neighbors are weighted according to their distance to the queried feature as follows.

Distance-weighted k-nearest neighbor WKNN

- Let $x_1, \ldots, x_k$ be the k nearest neighbors to the queried element, and $d_1, \ldots, d_k$ be the corresponding distances arranged in increasing order.
- The weight w_i for the i-th nearest neighbor is defined as:

$$w_i = \begin{cases} \frac{d_k - d_i}{d_k - d_1} & \text{if } d_k \neq d_1 \\ 1 & \text{otherwise} \end{cases}$$

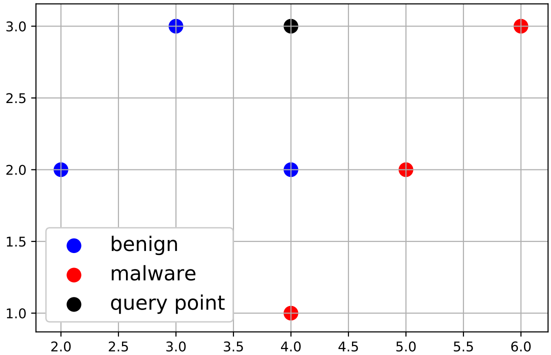
- The resulting class of the queried element is then defined by weighted majority voting as follows:

$$c_q = \arg\max_c \sum_{(x_i, c_i) \in T'} w_i \cdot \delta(c, c_i). \tag{2}$$

KNN - exercise 1

- Let $T = \{((2,2),\mathcal{C}),((3,3),\mathcal{C}),((4,2),\mathcal{C}),((4,1),\mathcal{M}),((5,2),\mathcal{M}),((6,3),\mathcal{M})\}$ be a training set, where $\mathcal{C}$ denotes the class of benign (clean) samples and $\mathcal{M}$ denotes the class of malicious samples.
- Let $x_q = (4,3)$ be testing feature vector and the parameter $k = 3$ be number of nearest neighbors.
- Use the k-Nearest Neighbor classifier and determine c_q.

KNN - solution 1



- $T' = \{((3,3),\mathcal{C}),((4,2),\mathcal{C}),((5,2),\mathcal{M})\}$ is the set of $k = 3$ nearest neighbors.
- Because the majority class is "clean" (i.e. not malware) then $c_q = \mathcal{C}$.

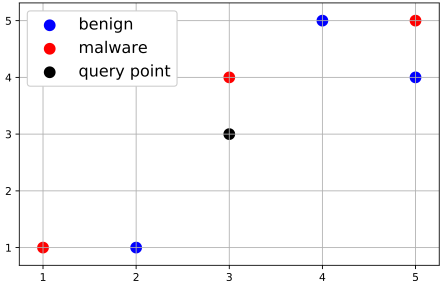
KNN - exercise 2

- Let $T = \{((2,3),\mathcal{C}),((3,3),\mathcal{C}),((3,2),\mathcal{C}),$ $((0,1),\mathcal{M}),((1,0),\mathcal{M}),((0,0),\mathcal{M})\}$ be a training set, where $\mathcal{C}$ denotes the class of benign (clean) samples and $\mathcal{M}$ denotes the class of malicious samples.
- Let $x_q = (2,1)$ be testing feature vector and the parameter $k = 3$ be number of nearest neighbors.
- Use the KNN classifier and determine c_q.

WKNN - exercise

- Let $T = \{((2,1), \mathcal{C}), ((4,5), \mathcal{C}), ((5,4), \mathcal{C}), ((1,1), \mathcal{M}), ((3,4), \mathcal{M}), ((5,5), \mathcal{M})\}$ be a training set, where $\mathcal{C}$ denotes the class of benign (clean) samples and $\mathcal{M}$ denotes the class of malicious samples.
- Let $x_q = (3,3)$ be testing feature vector and the parameter $k = 3$ be number of nearest neighbors.
- Use Distance Weighted k-Nearest Neighbor classifier and determine c_q.

WKNN - solution



- $T' = \{((3,4),\mathcal{M}),((4,5),\mathcal{C}),((5,4),\mathcal{C})\}$ is the set of $k=3$ nearest neighbors.
- The weight is defined as $w_i = \frac{d_k - d_i}{d_k - d_1}$, therefore $w_1 = \frac{\sqrt{5}-1}{\sqrt{5}-1} = 1$. $w_2 = \frac{\sqrt{5}-\sqrt{5}}{\sqrt{5}-1} = w_3 = 0$
- Since $1 > 0 + 0$ then $c_q = \mathcal{M}$.

Naive Bayes

- Naive Bayes classifier for the binary (i.e. 2 classes) classification problem works as follows.
- Naive Bayes classifier is a probability algorithm based on Bayes' theorem, which predicts the class with the highest *a posteriori* probability.
- Bayes' theorem states that:

$$P(A|B) = \frac{P(B|A)P(A)}{P(B)}, \quad (P(B) \neq 0).$$

- Let's have a set of two classes $\{\mathcal{C}, \mathcal{M}\}$, where $\mathcal{C}$ denotes the class of benign samples and $\mathcal{M}$ denotes the malware class.

Naive Bayes

- Let's have a training dataset and a test element (with unknown label) that is represented by a feature vector $x = (x_1, \dots, x_n)$.
- Let $P(\mathcal{M}|x)$denote the probability that the sample is malware given the feature vector x, which represents the sample. Similarly, let $P(\mathcal{C}|x)$ denote the probability that the sample is benign given the feature vector x. The Naive Bayes classification rule is defined by

$$\begin{aligned} \text{If } P(\mathcal{M}|x) &< P(\mathcal{C}|x), \; x \text{ x is classified as benign} \\ \text{If } P(\mathcal{M}|x) &> P(\mathcal{C}|x), \; x \text{ is classified as malware} \end{aligned} \tag{3}$$

Naive Bayes

- *a posteriori* probabilities $P(C|x)$ can be expressed as *a priori* probabilities and $P(x|C)$ probabilities using the Bayesian theorem:

$$P(C|x) = \frac{P(x|C)\ P(C)}{P(x)} \tag{4}$$

- Assuming that the values of the features are conditionally independent of each other, the equation (4) can be expressed as

$$P(C|x) = \frac{\prod_{i=1}^{n} P(x_i|C)\ P(C)}{P(x)} \tag{5}$$

Naive Bayes

- The probabilities $P(x_i|C)$ can be estimated from the training set by calculating the feature values for each class.
- More specifically, the probability $P(x_i = h|C)$ is represented as the number of elements of class C in the training set with the value h for the feature x_i divided by the number of elements of class C in the training set.
- The output of the classifier is the class C' with the highest probability:

$$C' = \arg\max_{C} \left(P(C) \prod_{i=1}^{n} P(x_i|C) \right) \tag{6}$$

Naive Bayes - exercise

- Let $T = \{((a,a,b),\mathcal{C}),((a,b,a),\mathcal{C}),((b,a,a),\mathcal{C}),((a,b,b),\mathcal{M}),((b,a,b),\mathcal{M}),((b,b,a),\mathcal{M})\}$ be a training set, where $\mathcal{C}$ denotes the class of benign (clean) samples and $\mathcal{M}$ denotes the class of malicious samples.
- Let $x_q = x = (b,b,b)$ be testing feature vector.
- Use Naive Bayes classifier and determine the c_q.

Naive Bayes - solution

- We need to compute $P(\mathcal{M}|x)$ and $P(\mathcal{C}|x)$
- Bayes' theorem says that

$$P(C|x) = \frac{P(x|C)\ P(C)}{P(x)},$$

where $C \in \{\mathcal{M}, \mathcal{C}\}$.

- Assuming that the values of the features are conditionally independent of each other, the equation

$$P(C|x) = \frac{\prod_{i=1}^{3} P(x_i|C)\ P(C)}{P(x)}$$

- The denominator $P(x)$ can be omitted since it does not depend on the class C.

Naive Bayes - solution

- For each $i = 1, 2, 3$ and for each $C \in \{\mathcal{M}, \mathcal{C}\}$ we need to calculate (using the training set T) $P(x_i = b|C)$ and also $P(C)$:
- $P(x_1 = b|\mathcal{M}) = \frac{2}{3}, P(x_2 = b|\mathcal{M}) = \frac{2}{3}, P(x_3 = b|\mathcal{M}) = \frac{2}{3}$
- $P(x_1 = b|\mathcal{C}) = \frac{1}{3}, P(x_2 = b|\mathcal{C}) = \frac{1}{3}, P(x_3 = b|\mathcal{C}) = \frac{1}{3}$
- $P(\mathcal{C}) = \frac{3}{6} = \frac{1}{2} = P(\mathcal{M})$

Naive Bayes - solution

- $P(\mathcal{C}|x) \to P(x_1 = b|\mathcal{C}) \cdot P(x_2 = b|\mathcal{C}) \cdot P(x_3 = b|\mathcal{C}) \cdot P(\mathcal{C}) = = \frac{1}{3} \cdot \frac{1}{3} \cdot \frac{1}{3} \cdot \frac{1}{2} = \frac{1}{54}$
- $P(\mathcal{M}|x) \to P(x_1 = b|\mathcal{M}) \cdot P(x_2 = b|\mathcal{M}) \cdot P(x_3 = b|\mathcal{M}) \cdot \cdot P(\mathcal{M}) = \frac{2}{3} \cdot \frac{2}{3} \cdot \frac{2}{3} \cdot \frac{1}{2} = \frac{8}{54}$
- Since $\frac{8}{54} > \frac{1}{54}$, then x is classified as malware, i.e., $c_q = \mathcal{M}$.