Algorithms of Information Security. Third Tutorial.

Faculty of Information Technology
Czech Technical University in Prague

November 1, 2023

Basic definition

Definition

Let C be a code of length n over alphabet A.

- C detects u errors if for every codeword $c \in C$ and every $x \in A^n$ with $x \neq c$, it holds that if $d(x, c) \leq u$ then $x \notin C$.
- C corrects v errors if for every codeword $c \in C$ and every $x \in A^n$ it holds that if $d(x, c) \leq v$ then nearest neighbor decoding of x outputs c.

Theorem.

- A code C detects u errors if and only if $d(C) > u$.
- A code C corrects v errors if and only if $d(C) \geq 2v + 1$.

Basic definition

Example 1. We know that the code C corrects 21 errors. State a lower bound on $d(C)$.

Solution.

We know that the code C corrects v errors if and only if $d(C) \geq 2v + 1$. If the code corrects 21 errors, thus in our case $d(C) \geq 43$.

Reed Solomon codes

Example 2. Consider the finite field F_{11} and let $\alpha = 6$. Find the generator polynomial of $RS(10, 7)$ (i.e. length is $n = 10$ and dimension is $k = 7$).

Solution.
Consider a finite field F_{11} and $\alpha = 6$. It is easy to check that $ord(\alpha) = 10$, and α is therefore a primitive element for F_{11}^*.
Note: we create the generator polynomial $g(x)$ of the RS code using the following formula:

$$g(x) = (x - \alpha)(x - \alpha^2) \dots (x - \alpha^{n-k}),$$

where α is a primitive element.
Then the generator polynomial is:

$$g(x) = (x - 6)(x - 6^2)(x - 6^3) = (x - 6)(x - 3)(x - 7) =$$

$$= (x+5)(x+8)(x+4) = (x^2+2x+7)(x+4) = x^3+6x^2+4x+6.$$

Cyclic code

Example 3. Determine whether the following codes are cyclic. Briefly explain your answers.

- The binary code $C = \{0000, 1010, 0101, 1110, 1101, 1011, 0111\}$.
- The ternary code $C = \{0000, 1212, 2121\}$.
- The ternary code $C = \{000, 011, 101, 110\}$.
- The binary code $C = \{0000, 1010, 0101, 1111\}$

Solution.

- Not cyclic since not linear (e.g. $1010 + 0101 = 1111$ is not in the code).
- Ternary code C is cyclic code. Let $c_1 = 0000, c_2 = 1212$ and $c_3 = 2121$, then:
  $c_1 + c_2 = c_2$, $c_1 + c_3 = c_3$, $c_2 + c_3 = c_1$, $c_2 + c_2 = c_3$, $c_3 + c_3 = c_2$
  It is closed under cyclic shifts: $c_1 \to c_1, c_2 \to c_3 \to c_2$ and thus it is cyclic.
- Not cyclic since not linear. Let $c_1 = 011$, then $c_1 + c_1 \notin C$, tj. $011 + 011 = 022$.
- It is linear and closed under cyclic shifts, it is therefore cyclic.

Lamport's OWF-based one-time passwords

- The core of Lamport's scheme requires cooperating client/server components to agree to use a common algorithm to generate a set of one-time passwords (client-side) and to validate the client passwords contained in each client-initiated request (service-side).
- The client generates a final sequence of values starting with the "seed" value, and each subsequent value is generated by applying some transformation algorithm (or $H(w)$ function) to the previous value of the sequence.
- The particular transformation algorithm used can be as simple or as complex as you like, as long as it always produces the same result for a given value.

Lamport's OWF-based one-time passwords

Example conversation.
Let's create an imaginary conversation between two people, Alice and Bob, who represent the client and the service. Suppose that Alice and Bob share an algorithm that was used to generate the following sequence:

$$18, 21, 24, 27, 40, 44, 47, 50.$$

For simplicity, we use whole numbers here. Our seed value is 18. The function $H(w)$ is as follows: add 3 and if the result contains the digit 3, change this digit to 4.

Lamport's OWF-based one-time passwords

This is how this conversation might go:

- Alice: Hello Bob - I want to talk to you. "Identifier = Alice", "Passkey = 50"
- Bob has no reason to reject Alice, so he saves a record of Alice's communication along with the provided key value: „Alice“ → „50“.
- Alice: Bob, can you lend me your car?: "Identifikátor = Alice", "Passkey = 47"
- Bob wants to make sure it's really Alice. Applies the function to "47": $H(47) = 50$. The function returns "50": $H(47) = 50$. which corresponds to the value of the access key stored in Bob's record. Answer Yes and replace in the entry for Alice: „Alice“ → „47“.

Lamport's OWF-based one-time passwords

- Alice receives the answer and is delighted, but realizes that she does not know where Bob's car is. Now she asks: Where is your car? "Identifikátor = Alice", "Passkey = 44".
- Bob wants to make sure again that it's really Alice. As before, it applies the function to the given key: $H(44) = 47$, which corresponds to the sequence value stored in the record for Alice. Bob responds with the information that Alice requested - the subway corner and the station - and substitutes in the record for Alice: „Alice“ → „44“.
- Alice receives the information she needs and then says, Thanks, Bob! Goodbye: "Identifikátor = Alice", "Passkey = 40".
- Bob realizes that Alice is done with this conversation. The last time it applies the function on the provided key is $H(40) = 44$. The record maintained for Alice is now deleted.

One-time passwords based on one-way functions (Lamport's scheme)

- In Lamport's one-time password scheme, the user begins with a secret w. A one-way function (OWF) H is used to define the password sequence: $w, H(w), H(H(w)), \ldots, H^t(w)$.
- The password for the i^{th} identification session, $1 \leq i \leq t$, is defined to be $w_i = H^{t-i}(w)$.

Lamport's OWF-based one-time passwords

Algorithm 1 Lamport's OWF-based one-time passwords

SUMMARY. A identifies itself to B using one-time passwords from a sequence.

1. *One-time setup.*
   - User A begins with a secret w. Let H be a one-way function.
   - A constant t is fixed (e.g.,$t = 100$ or 1000), defining the number of identifications to be allowed. (The system is thereafter restarted with a new w.)
   - A transfers (the *initial shared secret*) $w_0 = H^t(w)$, in a manner guaranteeing its authenticity, to the system B. B initializes its counter for A to $i_A = 1$.

Lamport's OWF-based one-time passwords

Algorithm 1 Lamport's OWF-based one-time passwords

2. *Protocol messages.* The i_{th} identification, $1 \leq i \leq t$, proceeds as follows:

   $$A \rightarrow B : A, i, w_i (= H^{t-i}(w)) \ (1)$$

   Here $A \rightarrow B : X$ denotes A sending the message X to B.

3. *Protocol actions.* To identify itself for session i, A does the following.

   - A's equipment computes $w_i = H^{t-i}(w)$ (easily done either from w itself, or from an appropriate intermediate value saved during the computation of $H^t(w)$ initially), and transmits (1) to B.
   - B checks that $i = i_A$, and that the received password w_i satisfies: $H(w_i) = w_{i-1}$. If both checks succeed, B accepts the password, sets $i_A \leftarrow i_A + 1$, and saves w_i for the next session verification.

Lamport's OWF-based one-time passwords

Example 4. Suppose we have an initial value $w = 0$ and we want to create a sequence of 10 values (except the initial value), the function H looks like this:

$$H(w) = w + 3.$$

Apply Lamport's algorithm (at least 2 iterations).

Lamport's OWF-based one-time passwords

Solution.
One-time setup.
We have the function $H(w) = w + 3$ and the initial value $w = 0$ and $t = 10$. Then the sequence looks like this:

$$0, 3, 6, 9, 12, 15, 18, 21, 24, 27, 30$$

A sends $B : w_0 = H^t(w)$, which in our case is $H^{10}(0) = 30$ and B sets the counter for $A : i_A = 1$. *What does the report look like for the first iteration?*

$$A \rightarrow B : A, 1, w_1 (= H^9(0) = 27) \qquad (1)$$

Lamport's OWF-based one-time passwords

Protocol action for the first iteration.

- A calculates $w_1 = H^9(0)$ and sends to B the message $A, 1, 27$.
- B checks if $i = i_A$, i.e. if $1 = 1$ and checks if the received password satisfies: $H(w_1) = w_0$, i.e. $H(27) = 27 + 3 = 30$. If both checks succeed, then B accepts the password and sets $i_A = 2$ and saves $w_1 = 27$ for the next session verification.

What the report looks like for the second iteration?

$$A \to B : A, 2, w_2 (= H^8(0) = 24)$$

Protocol action for the second iteration.

- A calculates $w_2 = H^8(0) = 24$ and sends to B the message $A, 2, 24$.
- B checks if $i_A = 2$ and checks if the received password satisfies: $H(w_2) = w_1$, i.e. $H(24) = 24 + 3 = 27$. If both checks succeed, then B accepts the password and sets $i_A = 3$ and saves $w_2 = 24$ for the next session verification.

We continue in the same way.