

WYDZIAŁ PODSTAWOWYCH PROBLEMÓW TECHNIKI
POLITECHNIKA WROCŁAWSKA

MEMORY-HARD FUNCTIONS

KONRAD ŚWIERCZYŃSKI
NR INDEKSU: 229818

Promotor
dr Filip Zagórski



Politechnika
Wrocławska
WROCŁAW 2019

Spis treści

| | | |
|----------|---------------------------------------|-----------|
| 1 | Wprowadzenie | 1 |
| 2 | Preliminaries | 3 |
| 2.1 | Etykietowanie Grafu | 3 |
| 2.2 | Superkoncentrator | 4 |
| 2.3 | Grafy Depth-Robust | 5 |
| 2.4 | Dispersed Graphs | 5 |
| 2.5 | Grafy Warstwowe | 6 |
| 2.6 | Ograniczenia Złożoności RSG | 6 |
| 3 | Riffle Scrambler | 9 |
| 4 | Implementacja | 11 |
| 5 | Instalacja i wdrożenie | 13 |
| 6 | Podsumowanie | 15 |
| | Bibliografia | 17 |
| A | Zawartość płyty CD | 19 |

Wprowadzenie

Moderately hard functions. Funkcje umiarkowanie ciężkie do obliczenia mają wiele zastosowań takich jak dowody pracy (ang. *proofs of work*), funkcje wyprowadzenia klucza oraz password hashing. Przy przechowywaniu haseł ważne jest, aby zminimalizować skutki wycieknienia pliku z hasłami. Zamiast przechowywać krotki (*login, password*) tekstem jawnym, dodaje się losową sól i przechowuje w postaci (*login, f(password, salt), salt*), gdzie f jest moderately hard function. Oznacza to, że funkcja ta musi być obliczana podczas każdego uwierzytelniania w celu sprawdzenia poprawności hasła. Nie może być ona zatem zbyt ciężka do obliczenia dla aplikacji uwierzytelniającej. Z drugiej strony, gdy krotka (*login, y, salt*) wycieknie, adversarz może przeprowadzać atak słownikowy obliczając funkcję f przy każdej próbie, co powinno być kosztowne. W tym celu zaczęto stosować funkcje, które obliczają wiele razy kryptograficzną funkcję skrótu. Popularnym przykładem takiej funkcji jest PBKDF2 (ang. *Password-Based Key Derivation Function 2*), dla której zalecanym parametrem bezpieczeństwa w 2000 roku było 1024 iteracji, a już w 2005 zaczęto zalecać 4096 iteracji, z powodu wzrostu wydajności CPU. Niestety takie podejście nie gwarantuje zabezpieczenia przed adversarzem używającym specjalizowany układ scalony (ang. *ASIC - Application-Specific Integrated Circuit*). Układy takie są znacznie bardziej wydajne pod względem szybkości obliczania funkcji skrótu takich jak SHA256 czy MD5 niż tradycyjne architektury.

– Porównanie Antminer - GPU - CPU –

Zauważono jednak, że na różnych architekturach koszt dostępu do pamięci jest dużo bardziej zrównoważony niż koszt obliczeń. [Percival [16]] Zaproponowano więc memory-hard functions (MHF), które wywołują podczas obliczania wiele kosztownych czasowo odwołań do pamięci.

scrypt - pierwsza taka funkcja

O MHF można myśleć jako o pewnej kolejności dostępu do komórek pamięci. Odwołania następują do już wcześniej obliczonych wartości w komórkach. Zatem kolejność tę można opisać jako acykliczny graf skierowany (DAG).

.....



Preliminaries

W dalszej części używana będzie następująca notacja. Zbiory $\mathbb{N} = \{0, 1, 2, \dots\}$, $\mathbb{N}^+ = \{1, 2, \dots\}$. $[c] := \{1, 2, \dots, c\}$ oraz $[b, c] = \{b, b+1, \dots, c\}$, gdzie $b, c \in \mathbb{N}$ oraz $b \leq c$.

Skierowany graf acykliczny (ang. *directed acyclic graph*, DAG) $G = (V, E)$ jest rozmiaru n jeżeli $|V| = n$. Wierzchołek $v \in V$ ma stopień wchodzący δ równy największemu stopniowi wchodzącemu wśród jego wierzchołków $\delta = \text{indeg}(v)$, jeżeli istnieje δ wchodzących krawędzi $\delta = |(V \times v) \cap E|$. Graf G ma stopień wchodzący $\delta = \text{indeg}(G) = \max_{v \in V} \text{indeg}(v)$. Wierzchołki o stopniu wchodzącym 0 nazywane są źródłami, a wierzchołki bez krawędzi wychodzących nazywane są ujściami.

Zbiór rodziców wierzchołka $v \in V$ oznaczany jest jako $\text{parents}_G(v) = \{u \in V : (u, v) \in E\}$. Uogólniając, zbiór przodków v oznaczany jest jako $\text{ancestors}_G(v) = \bigcup_{i \geq 1} \text{parents}_G^i(v)$, przyjmując $\text{parents}_G^{i+1}(v) = \text{parents}_G(\text{parents}_G^i(v))$. Jeżeli wybór grafu G wynika z kontekstu, będziemy oznaczać te zbiory jako parents oraz ancestors .

Zbiór wszystkich ujść w grafie G oznaczany jest jako $\text{sinks}(G) = \{v \in V : \nexists (v, u) \in E\}$. DAG G , który jest spójny, a tylko takie będą rozważane w dalszej części pracy, zachowuje równość $\text{ancestors}(\text{sinks}(G)) = V$.

Dla skierowanej ścieżki $p = (v_1, v_2, \dots, v_z)$ w G , jej długość jest równa ilości wierzchołków przez które przechodzi $\text{length}(p) := z$. Mając DAG G , oznaczamy długość jego najdłuższej ścieżki jako $\text{depth}(G)$.

Mając podzbiór wierzchołków grafu $S \subset V$, poprzez $G - S$ oznaczamy będziemy DAG otrzymany z G poprzez usunięcie wierzchołków z S oraz krawędzi wychodzących lub wchodzących do wierzchołków z S .

2.1 Etykietowanie Grafu

Definicja 2.1 (*Parallel/Sequential Graph Pebbling*). Niech $G = (V, E)$ będzie grafem skierowanym grafem acyklicznym i niech $T \subset V$ będzie zbiorem wierzchołków do oetykietowania. T będzie nazywane celem. Stanem etykietowania G jest zbiór $P_i \subset V$. Poprawnym etykietowaniem równoległym jest ciąg $P = (P_0, \dots, P_t)$ stanów etykietowania G , gdzie $P_0 = \emptyset$ oraz gdzie spełnione są warunki 1 oraz 2 poniżej. Etykietowanie sekwencyjne musi dodatkowo spełniać warunek 3.

1. Każdy wierzchołek z celu jest w pewnej konfiguracji oetykietowany (nie koniecznie wszystkie jednocześnie).

$$\forall x \in T \exists x \leq t : x \in P_x$$

2. Oetykietować wierzchołek można tylko wtedy, gdy wszyscy jego rodzice są oetykietowani w poprzednim kroku.

$$\forall i \in [t] : x \in (P_i \setminus P_{i-1}) \Rightarrow \text{parents}(x) \subset P_{i-1}$$

3. W każdym kroku można oetykietować co najwyżej jeden wierzchołek.

$$\forall i \in [t] : |P_i \setminus P_{i-1}| \leq 1$$

Zbiory poprawnych etykietowań sekwencyjnych i równoległych grafu G z celem T oznaczamy odpowiednio jako $\mathcal{P}_{G,T}$ oraz $\mathcal{P}_{G,T}^{\parallel}$. Etykietowania najbardziej interesujących przypadków, gdy $T = \text{sinks}(G)$, oznaczamy \mathcal{P}_G oraz $\mathcal{P}_G^{\parallel}$.

Można zauważyć, że $\mathcal{P}_{G,T} \subset \mathcal{P}_{G,T}^{\parallel}$.



Definicja 2.2 *Złożoność czasową (ang. time, t), pamięciową (ang. space, s), pamięciowo-czasową (ang. space-time, st) oraz łączna (ang. cumulative, cc) etykietowania $P = (P_0, \dots, P_t) \in \mathcal{P}_G^\parallel$ są zdefiniowane jako*

$$\Pi_t(P) = t, \Pi_s(P) = \max_{y \in [t]} |P_y|, \Pi_{st}(P) = \Pi_t(P) * \Pi_s(P), \Pi_{cc}(P) = \sum_{i \in [t]} |P_i|$$

Dla $\alpha \in \{s, t, st, cc\}$ oraz celu $T \subset V$, złożoności sekwencyjnego oraz równoległego etykietowania grafu G definiujemy jako

$$\Pi_\alpha(G, T) = \min_{P \in \mathcal{P}_{G,T}} \Pi_\alpha(P)$$

$$\Pi_\alpha^\parallel(G, T) = \min_{P \in \mathcal{P}_{G,T}^\parallel} \Pi_\alpha(P)$$

Kiedy $T = \text{sinks}(G)$, piszemy $\Pi_\alpha^\parallel(G)$ oraz $\Pi_\alpha(G)$.

Ponieważ $\mathcal{P}_{G,T} \subset \mathcal{P}_{G,T}^\parallel$, dla dowolnej złożoności etykietowania $\alpha \in \{s, t, st, cc\}$ oraz dowolnego grafu G złożoność etykietowania równoległe jest nie większa, niż złożoność etykietowania sekwencyjnego $\Pi_\alpha(G) \geq \Pi_\alpha^\parallel(G)$, a złożoność łączna jest nie większa, niż czasowo-pamięciowa $\Pi_{st}(G) \geq \Pi_{cc}(G)$ i $\Pi_{st}^\parallel(G) \geq \Pi_{cc}^\parallel(G)$.

W tej pracy głównie rozważane jest badanie złożoności Π_{st} , oraz Π_{cc}^\parallel , ponieważ ukazują one kolejno koszt przeprowadzania etykietowania na jednordzeniowej maszynie (np. procesor x86) oraz koszt etykietowania na wyspecjalizowanym układzie.

Aby zobaczyć jakie wartości mogą przyjąć przedstawione złożoności, rozważmy graf rozmiaru n . Każdy graf rozmiaru n może zostać oetykietowany w n krokach, ponieważ ma tylko n wierzchołków. Każdy stan etykietowania nie może również zawierać więcej niż n elementów. Zatem górne ograniczenie możemy przedstawić następująco

$$\forall G_n \in \mathbb{G}_n : \Pi_{cc}^\parallel(G_n) \leq \Pi_{st}(G_n) \leq n^2.$$

Zobaczmy jak wyglądają złożoności dla grafu pełnego $K_n = (V = [n], E = \{(i, j) : 1 \leq i < j \leq n\})$ oraz dla $Q_n = (V = [n], E = \{(i, i+1) : 1 \leq i \leq n\})$.

$$n(n-1)/2 \leq \Pi_{cc}^\parallel(K_n) \leq \Pi_{st}(K_n) \leq n^2$$

Graf K_n maksymalizuje złożoności etykietowania, a Π_{cc}^\parallel jest różne tylko o stałą od Π_{st} . Co oznacza, że koszt obliczania funkcji opartej na takim grafie byłby zdominowany przez koszt dostępu do pamięci, a wyspecjalizowane układy nie dały by dużej przewagi nad tradycyjnym procesorem. Jest to bardzo pożądane dla funkcji memory-hard, jednak ze względu na bardzo wysoki stopień grafu K_n , nie jest on przydatny przy konstruowaniu MHF. Stopień wchodzący w grafie Q_n jest równy 1, jednak złożoność etykietowania jest bardzo niska

$$\Pi_{cc}^\parallel(Q_n) = \Pi_{st}(Q_n) \leq n.$$

Oznacza to, że koszt obliczania funkcji opartej na takim grafie (taką funkcją jest PBKDF2) nie jest zależny w dużej mierze od kosztu dostępu do pamięci nawet dla dużego n .

2.2 Superkoncentrator

Superkoncentrator jest grafem, w którym moc zbioru przodków dla wierzchołków szybko rośnie wraz z numerem pokolenia. Oznacza to, że zbiór kolejnych wierzchołków, będzie posiadał liczebny zbiór rodziców, co czyni superkoncentrator bardzo przydatnym do konstrukcji MHF.

Definicja 2.3 (*N-Superkoncentrator*) *Skierowany graf acykliczny $G = (V, E)$ o ustalonym stopniu wchodzącym, N wejściach i N wyjściach nazywany jest N-Superkoncentratorem, gdy dla każdego $k \in [N]$ oraz dla każdej pary podzbiorów $V_1 \subset V$ k wejść i $V_2 \subset V$ k wyjść istnieje k wierzchołkowo-rozłącznych ścieżek łączących wierzchołki ze zbioru V_1 z wierzchołkami w V_2 .*

Definicja 2.4 ((N, λ) -Superkoncentrator) Niech $G_i, i = 0, \dots, \lambda - 1$ będą N -Superkoncentratorami. Niech graf G będzie połączeniem wyjść G_i do odpowiadających wejść w G_{i+1} dla $i = 0, \dots, \lambda - 2$. Graf G jest nazywany (N, λ) -Superkoncentratorem.

Twierdzenie 2.1 (Ograniczenie dolne dla (N, λ) -Superkoncentratora) Etykietowanie (N, λ) -Superkoncentratora używając $S \leq N/20$ etykiet, wymaga T kroków, gdzie

$$T \geq N \left(\frac{\lambda N}{64S} \right)^\lambda.$$

2.3 Grafy Depth-Robust

Alwen i Blocki w swojej pracy [1] pokazali, że istnieje zależność między złożonością etykietowania, a depth-robustness. Na podstawie tej właściwości potrafimy określić dolne i górne ograniczenie Π_{cc}^{\parallel} .

Definicja 2.5 (Depth-Robustness) Dla $n \in \mathbb{N}$ oraz $e, d \in [n]$ DAG $G = (V, E)$ jest (e, d) -depth-robust, jeżeli

$$\forall S \subset V \quad |S| \leq e \Rightarrow \text{depth}(G - S) \geq d.$$

Twierdzenie 2.2 Niech DAG G będzie (e, d) -depth-robust, wtedy $\Pi_{cc}^{\parallel} > ed$.

Definicja 2.6 Jeżeli graf G nie jest (e, d) -depth-robust to nazywany jest (e, d) -reducible.

Twierdzenie 2.3 Niech $G \in \mathbb{G}_{n, \delta}$ taki, że G jest (e, d) -reducible. Wtedy

$$\Pi_{cc}^{\parallel}(G) = O \left(\min_{g \in [d, n]} \left\{ n \left(\frac{dn}{g} + \delta g + e \right) \right\} \right)$$

biorąc $g = \sqrt{\frac{dn}{\delta}}$ upraszcza się to do $\Pi_{cc}^{\parallel}(G) = O \left(n(\sqrt{dn\delta} + e) \right)$.

2.4 Dispersed Graphs

Definicja 2.7 (Dependencies) Niech $G = (V, E)$ będzie acyklicznym grafem skierowanym. Niech $L \subseteq V$. Mówimy, że L ma (z, g) -dependency jeżeli istnieją wierzchołkowo rozłączne ścieżki p_1, \dots, p_z kończące się w L , gdzie każda jest długości co najmniej g .

Definicja 2.8 (Dispersed Graph) Niech $g, k \in \mathbb{N}$ i $g \geq k$. DAG G jest nazywany (g, k) -dispersed jeżeli istnieje uporządkowanie jego wierzchołków takie, że następujące warunki są spełnione. Niech $[k]$ oznacza ostatnie k wierzchołków o uporządkowaniu G i niech $L_j = [jg, (j+1)g - 1]$ będzie j -tym podprzedziałem. Wtedy $\forall j \in [k/g]$ przedział L_j ma (g, g) -dependency. W ogólności, jeżeli dla $\epsilon \in (0, 1]$ każdy przedział L_j ma tylko $(\epsilon g, g)$ -dependency, graf G nazywany jest (ϵ, g, g) -dispersed.

Definicja 2.9 Acykliczny graf skierowany $G = (V, E)$ nazywany jest $(\lambda, \epsilon, g, k)$ -dispersed jeżeli istnieje $\lambda \in \mathbb{N}^+$ rozłącznych podzbiorów wierzchołków $\{L_i \subseteq V\}$, każdy o rozmiarze k oraz spełnione są następujące warunki.

1. Dla każdego L_i istnieje ścieżka przechodząca przez wszystkie wierzchołki L_i .
2. Dla ustalonego porządku topologicznego G . Dla każdego $i \in [\lambda]$ niech G_i będzie podgrafem G , zawierającym wszystkie wierzchołki z G , aż do ostatniego wierzchołka z L_i . G_i jest (ϵ, g, k) -dispersed.

Zbiór grafów, które są $(\lambda, \epsilon, g, k)$ -dispersed oznaczamy jako $\mathbb{D}_{\epsilon, g}^{\lambda, k}$.

Twierdzenie 2.4 Niech $G \in \mathbb{D}_{\epsilon, g}^{\lambda, k}$.

$$\Pi_{cc}^{\parallel}(G) \geq \epsilon \lambda g \left(\frac{k}{2} - g \right)$$



2.5 Grafy Warstwowe

Definicja 2.10 (λ -Stacked Sandwich Graphs) Niech $n, \lambda \in \mathbb{N}^+$ takie, że $\lambda + 1$ dzieli n oraz niech $k = n/(\lambda + 1)$. Mówimy, że graf G jest λ -stacked sandwich DAG, jeżeli G zawiera ścieżkę przechodzącą przez n wierzchołków (v_1, \dots, v_n) oraz dzieląc go na warstwy $L_j = \{v_{jk+1}, \dots, v_{j(k+\lambda)}\}$, dla $j \in \{0, \dots, \lambda\}$, pozostałe krawędzie łączą wierzchołki z niższej warstwy L_j jedynie z wierzchołkami z wyższych warstw L_i , $i \in \{j + 1, \dots, \lambda\}$.

Lemat 2.1 Niech G będzie λ -stacked sandwich DAG, wtedy dla dowolnego $t \in \mathbb{N}^+$, G jest $(n/t, \lambda t)$ -reducible.

2.6 Ograniczenia Złożoności RSG

Twierdzenie 2.5 Niech $\lambda, n \in \mathbb{N}^+$ takie, że $n = \bar{n}(2\lambda c + 1)$, gdzie $c \in \mathbb{N}$ i $\bar{n} = 2^c$. Wtedy dla $g = \lfloor \sqrt{\bar{n}} \rfloor$ $RSG_{\lambda}^{\bar{n}} \in \mathbb{D}_{1,g}^{\lambda, \bar{n}}$ oraz $\Pi_{cc}^{\parallel}(RSG_{\lambda}^{\bar{n}}) = \Omega\left(\frac{n^{1.5}}{c\sqrt{c\lambda}}\right)$.

Dowód. Niech $G = RSG_{\lambda}^{\bar{n}}$, niech $G_1, G_2, \dots, G_{\lambda}$ będą podgrafami G opisanymi w DEF[...]. Pokażemy, że każdy G_i jest (g, \bar{n}) -dispersed dla $g = \lfloor \sqrt{\bar{n}} \rfloor$.

Wybermy $i \in [\lambda]$ niech L_1 będzie ostatnimi \bar{n} wierzchołkami w porządku topologicznym grafu G_i . Oznaczamy wierzchołki zbioru L_1 poprzez $1 \times [\bar{n}]$, gdzie druga pozycja odpowiada kolejności wierzchołka w porządku topologicznym. Niech $\bar{g} = \lfloor \bar{n}/g \rfloor$, dla każdego $j \in [\bar{g}]$ $L_{1,j} = \{< 1, jg + x > : x \in [0, g - 1]\}$. Pokażemy, że wszystkie $L_{1,j}$ mają (g, g) -dependency.

Niech L_0 będzie \bar{n} pierwszymi wierzchołkami G_i , które oznaczamy $0 \times [\bar{n}]$ (ponownie druga pozycja odpowiada porządkowi topograficznemu). Zauważmy, że dla $n > 1$ i $g = \lfloor \sqrt{\bar{n}} \rfloor$ prawdą jest, że $g(g - 2c + 1) \leq n$. Zatem zbiór $S = \{< 0, i(g - 2c + 1) > : i \in [g]\}$ jest całkowicie zawarty w L_0 .

Z własności RSG [Superconcentrator-] wynika, że skoro zbiory S oraz $L_{1,j}$ mają po g wierzchołków, to istnieje g wierzchołkowo-rozłącznych ścieżek o długości $2c$ między wierzchołkami tych zbiorów. Zatem $L_{1,j}$ ma $(g, 2c)$ -dependency.

Rozszerzmy to do (g, g) -dependency. Niech ścieżka p zaczynająca się w wierzchołku $< 0, v > \in S$ będzie ścieżką w $(g, 2c)$ -dependency $L_{1,j}$. Zauważmy, że istnieje ścieżka przechodząca przez wszystkie wierzchołki L_0 oraz, że wierzchołki zbioru S są oddzielone między sobą o $g - 2c$ wierzchołków. Możemy dodać na początek ścieżki p ścieżkę $< 0, v - (g - 2c - 1) >, < 0, v - (g - 2c - 2) >, \dots, < 0, v >$. Otrzymujemy w ten sposób ścieżkę p_+ o długości $2c + g - 2c = g$. Ponieważ każda para ścieżek $p \neq q$ w $(g, 2c)$ -dependency $L_{1,j}$ jest wierzchołkowo-rozłączna, to w szczególności zaczynać się muszą w różnych wierzchołkach $< 0, v_p > \neq < 0, v_q >$. Ponieważ wierzchołki w S są od siebie oddalone o $g - 2c$ wierzchołków, zatem ścieżki p^+ i q^+ nadal pozostają rozłączne. Rozszerzając w ten sposób wszystkie ścieżki z $(g, 2c)$ -dependency otrzymujemy ścieżki wierzchołkowo-rozłączne długości g . Z tego wynika, że $L_{1,j}$ ma (g, g) -dependency, co dowodzi, że $RSG_{\lambda}^{\bar{n}} \in \mathbb{D}_{1,g}^{\lambda, \bar{n}}$. Pozostaje obliczyć górne ograniczenie używając [Theorem 6 ABP2017].

$$\Pi_{cc}^{\parallel}(RSG_{\lambda}^{\bar{n}}) = \lambda g \left(\frac{\bar{n}}{2} - g \right) \geq \lambda \lfloor \sqrt{\bar{n}} \rfloor \left(\frac{\bar{n}}{2} - \lfloor \sqrt{\bar{n}} \rfloor \right) = \lambda \sqrt{\bar{n}} \left(\frac{\bar{n}}{2} - \sqrt{\bar{n}} \right) - O(\bar{n}) = \Omega(\lambda \bar{n}) = \Omega\left(\frac{n^{1.5}}{c\sqrt{c\lambda}}\right) \quad \square$$

Twierdzenie 2.6 Niech $\lambda, g \in \mathbb{N}^+$, $N = 2^g$, $n = N(2\lambda g + 1)$, wtedy

$$\Pi_{cc}^{\parallel}(RSG_{\lambda}^{\bar{n}}) = O\left(n^{1.6}\right)$$

Dowód. Kozytając, z [...], RSG_{λ}^N jest λ -Stacked Sandwich Graph. Z [Lemma 4.2 AB16] wynika więc, że RSG_{λ}^N jest $(n/t, \lambda + t - \lambda - 1)$ -reducible dla dowolnego $t \geq 1$. Z twierdzenia 10 [ABP17] wynika, że $\Pi_{cc}^{\parallel}(RSG_{\lambda}^{\bar{n}}) = O\left(n \left(\sqrt{(\lambda t + t - \lambda - 1)n\delta + \frac{n}{t}} \right)\right)$. Aby dostać najdokładniejsze (najmniejsze) ograniczenie górne trzeba zminimalizować $n \left(\sqrt{(\lambda t + t - \lambda - 1)n\delta + \frac{n}{t}} \right)$. Zanim jednak przejdziemy do minimalizowania, uproścmy nieco to wyrażenie

$$n \left(\sqrt{(\lambda t + t - \lambda - 1)n\delta + \frac{n}{t}} \right) \leq n \left(\sqrt{2\lambda t n\delta + \frac{n}{t}} \right).$$

Teraz możemy znaleźć minimum względem naszego parametru t .

$$\frac{\partial}{\partial t} n \left(\sqrt{(2\lambda t)n\delta} + \frac{n}{t} \right) = \frac{\sqrt{2\delta\lambda n^3}}{2t} - \frac{n^2}{t^2}$$

Minimum znajduje się w punkcie, gdzie pochodna ma wartość zero.

$$\begin{aligned} \frac{\sqrt{2\delta\lambda n^3}}{2t} - \frac{n^2}{t^2} &= 0 \\ t &= \frac{2n^2}{\sqrt{2\delta\lambda n^3}} = O\left(n^{\frac{1}{3}}\right) \end{aligned}$$

Zatem podstawiając t minimalizujące ograniczenie górne do wzoru z twierdzenie 10 [ABP17] otrzymujemy

$$\Pi_{cc}^{\parallel}(RSG_{\lambda}^{\bar{n}}) = O\left(n \left(\sqrt{(\lambda n^{\frac{1}{3}} + n^{\frac{1}{3}} - \lambda - 1)n\delta} + \frac{n}{n^{\frac{1}{3}}} \right)\right) = O\left(n^{1.\bar{6}}\right)$$

□



Riffle Scrambler



Implementacja



Instalacja i wdrożenie

W tym rozdziale należy omówić zawartość pakietu instalacyjnego oraz założenia co do środowiska, w którym realizowany system będzie instalowany. Należy przedstawić procedurę instalacji i wdrożenia systemu. Czynności instalacyjne powinny być szczegółowo rozpisane na kroki. Procedura wdrożenia powinna obejmować konfigurację platformy sprzętowej, OS (np. konfiguracje niezbędnych sterowników) oraz konfigurację wdrażanego systemu, m.in. tworzenia niezbędnych kont użytkowników. Procedura instalacji powinna prowadzić od stanu, w którym nie są zainstalowane żadne składniki systemu, do stanu w którym system jest gotowy do pracy i oczekuje na akcje typowego użytkownika.



Podsumowanie

W podsumowanie należy określić stan zakończonych prac projektowych i implementacyjnych. Zaznaczyć, które z zakładanych funkcjonalności systemu udało się zrealizować. Omówić aspekty pielęgnacji systemu w środowisku wdrożeniowym. Wskazać dalsze możliwe kierunki rozwoju systemu, np. dodawanie nowych komponentów realizujących nowe funkcje.

W podsumowaniu należy podkreślić nowatorskie rozwiązania zastosowane w projekcie i implementacji (niebanalne algorytmy, nowe technologie, itp.).



Bibliografia

- [1] J. Alwen, J. Blocki, K. Pietrzak. Depth-robust graphs and their cumulative memory complexity. *Annual International Conference on the Theory and Applications of Cryptographic Techniques*, strony 3–32. Springer, 2017.
- [2] K. Gotfryd, P. Lorek, F. Zagórski. Rifflescrambler—a memory-hard password storing function. *European Symposium on Research in Computer Security*, strony 309–328. Springer, 2018.



Zawartość płyty CD

W tym rozdziale należy krótko omówić zawartość dołączonej płyty CD.

