Vedlegg

Privacy Pass ble introdusert av Davidson m.fl. [DGS+18] i 2018, opprinnelig som en løsning for å redusere bruken av CAPTCHA-tester på internett. Arbeidet ble gjort i samarbeid med skyleverandøren Cloudflare, som også har tatt initiativ til standardisering av løsningen.

Del A av dette vedlegget gir en kort introduksjon til det matematiske grunnlaget til løsningen. I del B introduserer vi protokollen, og i del C kommer vi med forslag til parametre, med grunnlag i et utkast til standardisering av protokollen.

A Bakgrunn og notasjon

A.1 Diskrete logaritmer

La $\mathcal{G}$ være en gruppe av primsk orden p, og la g være en generator for $\mathcal{G}$. Da kan alle elementer X i $\mathcal{G}$ kunne skrives som $X = g^x$ for heltall $0 \leq x < p$. Dersom man er gitt $g, \mathcal{G}, X$ ønsker vi at det skal være vanskelig å finne x, den *diskrete logaritmen* til X. Sikkerheten til den følgende protokollen er basert på dette problemet.

A.2 Hashfunksjoner

La $\mathtt{H}$ være en funksjon hvor definisjonsmendgen er alle tekststrenger av vilkårlig lengde (argumenter) og verdimengen er gruppen $\mathcal{G}$ (funksjonsverdier). Vi sier at $\mathtt{H}$ er en hashfunksjon dersom

1. alle argumenter x går til tilfeldige elementer y i $\mathcal{G}$
2. gitt en verdi y så er det vanskelig å finne et argument x slik at $y = \mathtt{H}(x)$
3. det er vanskelig å finne to argumenter x_1 og x_2 slik at $\mathtt{H}(x_1) = \mathtt{H}(x_2)$

Vi vil videre anta at funksjonsverdier $y = \mathtt{H}(x)$ binder oss til verdien x: Dersom vi publiserer y, vil det være vanskelig å senere finne en annen x' slik at $y = \mathtt{H}(x')$.

A.3 Korrekthetsbevis

La $K = g^k$ og $H = h^k$, hvor g og h er generatorer for $\mathcal{G}$. Verdiene K og H er offentlige, mens $1 \leq k < p$ er et hemmelig heltall. Dersom man ønsker å bevise at man vet verdien av k uten å gjøre den offentlig, samt bevise at den diskrete logaritmen til K og H er den samme, gitt g og h, kan man beregne følgende korrekthetsbevis:

- Trekk et tilfeldig tall $1 \leq r < p$ og beregn $X = g^r$ og $Y = h^r$
- La $c = \mathtt{H}(g, h, \mathcal{G}, K, H, X, Y)$ og beregn $z = r - ck \mod p$
- Publiser beviset $\pi = (c, z)$

Da kan man verifisere beviset på følgende måte:

- Beregn $c' = \mathtt{H}(g, h, \mathcal{G}, K, H, g^z K^c, h^z H^c)$
- Godta dersom $c \stackrel{?}{=} c'$, ellers avslå.

Vi merker oss at sannsynligheten for at beviset ikke er korrekt er $1/p$, som er veldig liten når p er veldig stor. Vi bemerker også at vi ikke kan lære noe om k fra selve beviset, så dersom k er en hemmelig nøkkel vil den forbli hemmelig.

B Protokoll

Vi deler opp protkollen vår i flere steg. Beskrivelsene er konsistente med standardiseringsdokumentene for *Oblivious Pseudorandom Functions using Prime-Order Groups* [DSW20] og *Privacy Pass* [DS19].

B.1 Setup

Uviklerne av Smittestopp 2.0 bestemmer parametre for protokollen, for eksempel som angitt i del C, som gjøres offentlig tilgjengelig. De offentlige parametrene består av en gruppe $\mathcal{G}$ av primisk orden p sammen med en generator g, samt en hashfunksjon $\mathtt{H}$ som tar inn vilkårlige strenger og sender dem til elementer i $\mathcal{G}$.

Utviklerne genererer så en mengde hemmelige nøkler $1 \leq k_i < p$, hvor i er en teller som starter på $i = 0$ inntil to uker før Smittestopp 2.0 lanseres og som økes med 1 for hvert døgn som går. De beregner så $K_i = g^{k_i}$ for hver i. Hver k_i deles mellom Helsenorge og Smittestopp-backend, mens K_i gjøres offentlig tilgjengelig sammen med parametrene oppgitt ovenfor. Denne prosessen kan oppdateres fortløpende, for å sørge for at nye nøkkler alltid er tilgjengelig i systemet. Spesielt vil parametrene og nøklene gjøres tilgjengelig for appen.

Oppsummering:

- Parametrene $\mathcal{G}, g, p, \mathtt{H}, \{K_i\}$ er offentlig tilgjengelig
- Nøklene $\{k_i\}$ er bare kjent for Helsenorge og Smittestopp

B.2 Smittevarsling

Bruker: Ved positiv test. En bruker har testet positivt og skal gjøre sine smittenøkler tilgjengelig. Først logger brukeren inn på Helsenorge via appen. Brukerens telefon genererer så en tilfeldig tekststreng t av en gitt lengde, for eksempel 256 bit, samt et tilfeldig tall $1 \leq r < p$. Til slutt beregner man $T = \mathtt{H}(t)$, og sender $P = T^r$ til Helsenorge.

Helsenorge: Når smitte meldes. Når en bruker logger inn på Helsenorge for å melde smitte, vil Helsenorge motta identifikasjon `id` sammen med et element P. Det sjekkes først at P er i gruppen $\mathcal{G}$ og at P er ulik 0 og 1. Helsenorge sender så `id` til MSIS for å få bekreftet at brukeren har testet positivt. Dersom

ikke, så avsluttes prosessen. Dersom brukeren har smittet positivt så beregner Helsenorge $Q = P^{k_i}$ hvor i er tidligste dato for smittsomhet. Helsenorge beregner også et korrekthetsbevis $\pi = (c, z)$ for at Q og K_i er beregnet med den samme hemmelige eksponenten k_i, med hensyn til P og g. Til slutt sendes Q og π, samt indeks i, tilbake til brukeren.

Brukeren: Ved brekreftelse fra Helsenorge. Når brukeren mottar Q, π og i fra Helsenorge, sjekker han først at i er en indeks som korresponderer til en av de siste 14 dagene, og at π er et gyldig bevis. Dersom ja; beregn $W = Q^{1/r}$. Til slutt sender han t, W og i, samt alle smittenøklene for dagene med indeks i og høyere, til Smittestopp-backend.

Smittestopp-backend: Ved mottatt token. Når en bruker melder i fra om smitte vil serveren motta t, W og i, samt en mengde smittenøkler. Serveren sjekker om i er en indeks som korresponderer til en av de siste 14 dagene, at verdien t ikke er mottatt tidligere, og beregner så $W' = \mathtt{H}(t)^{k_i}$. Dersom $W = W'$, i er gyldig og t er ny, så vil serveren godta dette som token. Deretter sjekker serveren at alle smittenøklene er fra dager med indeks i og høyere, og godtar de som er gyldige. Disse vil så distribueres til alle brukere for å varsle brukere som har vert i kontakt med den som er smittet. Verdien t lagres så for å hindre at brukeren gjenbruker t senere.

Oppsummering:

1. Brukeren velger t og r, og sender $T = \mathtt{H}(t)^r$ til Helsenorge.
2. Helsenorge beregner $Q = P^{k_i}$ og $\pi = (c, z)$, og sender (Q, π, i) til brukeren.
3. Brukeren sjekker beviset, beregner $W = Q^{1/r}$ og sender (t, W, i), samt smittenøkler, til Smittestopp-backend.
4. Smittestopp-backend beregner $W' = H(t)^{k_i}$ og sjekker at W' og W er like. Så lagres t til senere, og man publiserer gyldige smittenøkler.

C Parametre og implementasjon

I tråd med avsnitt 4.1 i standardiseringsutkastet [DSW20] foreslår vi at systemet instansieres med den elliptiske kurven Curve25519 som gruppe og SHA-512 som hashfunksjonen `H`. Det refererte dokumentet inneholder ytterligere detaljer. Dan Bernstein har laget en referanseimplementasjon [Ber05] av Curve25519, og publisert koden som offentlig eiendom.

Det finnes en offentlig tilgjengelig implementasjon av Privacy Pass tilpasset sin opprinnelige bruk [DGS+20]. Koden er publisert under en BSD 3-Clause-lisens, noe som gjør det mulig å bruke og modifisere koden, også til kommersiell bruk.

Disse anbefalingene er ment å gi et utgangspunkt for å kunne komme raskt i gang. Dersom det er mulig å bruke den koden vi refererer til veldig direkte er

det for eksempel ikke et problem å bruke den elliptiske kurven P-256 i stedet, slik de allerede har gjort.

Referanser

[Ber05] Daniel J. Bernstein. Curve25519: high-speed elliptic curve cryptography, 2005. https://cr.yp.to/ecdh.html.

[DGS+18] Alex Davidson, Ian Goldberg, Nick Sullivan, George Tankersley, and Filippo Valsorda. Privacy pass: Bypassing internet challenges anonymously. *Proc. Priv. Enhancing Technol.*, 2018(3):164–180, 2018.

[DGS+20] Alex Davidson, Ian Goldberg, Nick Sullivan, George Tankersley, and Filippo Valsorda. Privacy pass (nettside), 2020. https://privacypass.github.io.

[DS19] A. Davidson and N. Sullivan. The Privacy Pass Protocol. Internet-Draft, Cloudflare, November 2019.

[DSW20] A. Davidson, N. Sullivan, and C. Wood. Oblivious Pseudorandom Functions (OPRFs) using Prime-Order Groups. Internet-Draft, Cloudflare, July 2020.