

# HA A MÁSODIK FAKTOR TOTP, AKKOR MINDEN RENDBEN...

## Egy hiedelem vizsgálata

*Pfeiffer Szilárd – [mailbox@pfeifferszilard.hu](mailto:mailbox@pfeifferszilard.hu)*

*Balsai Péter – [balsai.peter@mithrandir.hu](mailto:balsai.peter@mithrandir.hu)*

### Absztrakt

Tanulmányunkban az Ügyfélkapu+ kétlépcsős azonosításában alkalmazott időalapú egyszer használatos jelszó (TOTP) megoldást vizsgáljuk. Célunk annak bemutatása, hogy még egy egyszerűnek tűnő technológia is hordozhat biztonsági kockázatokat. Ismertetjük a TOTP előállításához szükséges fő paramétereket, valamint az ezekkel szemben támasztható biztonsági és használhatósági követelményeket. Elemzésünk során kitérünk arra is, hogy az Ügyfélkapu+ és a legnépszerűbb kliensalkalmazások mennyiben felelnek meg ezeknek. Rámutatunk, hogy a TOTP jelenléte önmagában nem garantál teljes körű biztonságot, szélsőséges esetekben pedig csupán annak illúzióját kelti.

**Kulcsszavak:** ügyfélkapu+, mfa, 2fa, otp, hotp, totp, hmac, freeotp

### Abstract

**If the Second Factor is TOTP, then Everything is Fine...**

#### Testing a belief

In this paper, we examine the use of time-based one-time passwords (TOTP) in the two-factor authentication mechanism of Ügyfélkapu+. Our goal is to demonstrate that even seemingly simple technologies can pose significant security risks. We outline the key parameters required for TOTP generation, as well as the related requirements for security and usability. We also assess how Ügyfélkapu+ and the most popular client applications meet these requirements. Our findings highlight that the presence of TOTP alone does not guarantee comprehensive security and, in extreme cases, may merely create the illusion of it.

**Keywords:** Ügyfélkapu+, MFA, 2FA, OTP, HOTP, TOTP, HMAC, FreeOTP

### 1. Bevezetés

Az informatikai biztonság teljes története még egy fél évszázadra sem tekint vissza, mégis legendák lengik körül. Amikor egy hiedelem kellően régóta áll fenn, kellően sokan hisznek benne, és elég nagy súllyal, tekintéllyel, erővel hirdetik, akkor dogmává válik. Egy a dogmák

közül, hogy ha két-, vagy többfaktoros azonosítást (2FA<sup>1</sup> / MFA<sup>2</sup>) alkalmazunk, az egyszer használatos jelszavak (one-time password, OTP) – amik jellemzően időalapúak (TOTP<sup>3</sup>) – használata minden körülmények között jobb, mint ha a megoldást nem alkalmazzuk.

Ez azonban nem feltétlenül igaz, mivel a TOTP biztonságát alapvetően határozza meg paramétereinek biztonsága, illetve az OTP-t kezelő kliens szoftver fejlesztése során alkalmazott biztonságtudatosság mértéke. Mindezek vizsgálatára apropóul az Ügyfélkapu szolgáltatás 2FA kiegészítése (Ügyfélkapu+) szolgált. Pontosabban az apropót a második faktor (TOTP) kezelésére használt alkalmazás ([FreeOTP](#)) biztonsági figyelmeztetése<sup>4</sup> adta[1]. A figyelmeztetést az Ügyfélkapu+ regisztráció során kapott QR-kód<sup>5</sup> – ami a TOTP-paramétereket tartalmazza – beolvasása során kaptuk. Ennek hatására kezdtünk vizsgálni:

1. A FreeOTP forráskódjának tanulmányozása révén megállapítottuk, hogy több TOTP-paramétertípus nem megfelelő értékei esetén is kaphatunk figyelmeztetést.
2. A QR-kódból a TOTP-paraméterek kinyertük, és előállítottunk hasonló QR-kódokat, amik csak egy-egy paraméterben térnek el az eredetitől, így a hibüzenet pontos oka megállapíthatóvá vált.

Mielőtt azonban a megállapításokra kitérnénk, tisztáznunk kell, mik a TOTP paraméterei és azok milyen biztonsági jelentőséggel bírnak.

## 2. Többtényezős hitelesítés

A 2FA / MFA alapvetése, hogy a felhasználónév-jelszó párossal történő azonosítás nem kellően biztonságos, mivel a jelszó megszerzése/kitudódása kompromittálja a felhasználói fiókot. Márpedig a jelszavak többfelől is támadhatóak, akár egyszerre is.

- **Kliens oldalon:** mind az asztali, mind a mobil készülékek biztonsága hagy(hat) kívánnivalót maga után – érheti támadás a jelszavakat, akár tárolás<sup>6</sup>, akár használat<sup>7</sup> közben. Rossz példa a tárolásra jelszavak szöveges dokumentumban való mentése, vagy egy hasonló célú cetli a monitoron.

---

1 two-factor authentication (2FA)

2 multi-factor authentication (MFA)

3 one-time password (OTP)

4 ha a TOTP paraméterek kriptográfiai szempontból gyengék

5 quick-response (QR) code

6 data at rest

7 data in use

- **Szerver oldalon:** a tárolás – szélsőséges esetben szöveges formában, vagy gyenge kriptográfiai hasítófüggvénnyel<sup>8</sup>, esetleg só/bors<sup>9</sup> alkalmazása nélkül kódolva történhet. Ezen túl használat közben (pl.: memória analizáló támadó szoftverek) is jelentkeznek kockázatok. Probléma, hogy az alkalmazott megoldások jóságára nincs befolyásunk és a megvalósítás ellenőrzésére sincs módunk.
- **Az átvitel során:** lehallgatással megszerezhetőek a szöveges formában továbbított jelszavak (ez az általános gyakorlat). Ennek a lehetőségét a csatornatitkosítás – például TLS (korábban SSL) – elterjedése és használata csökkentti.

Végiggondolva nem meglepő, hogy az informatikai biztonság fejlődő gyakorlata, valamilyen más vagy kiegészítő biztonsági lépés bevezetését találta a jelszavak helyett/mellett szükségesnek, melyre számos megoldás született[2]. Ezek egyike lett az OTP.

### 2.1. Egyszer használatos jelszavak

Egy OTP előállításához szükséges kód viszonylag egyszerű, könnyen implementálható. Kiindulópontja egy titok (kulcs) – amit mind az azonosító-, mind az azonosított fél ismer –, amiből aztán az OTP-t származtatjuk. A származtatás egy kriptográfiai hasítófüggvény-alapú üzenetellenőrző kód<sup>10</sup> (HMAC) segítségével történik, melynek bemenete a titok és egy plusz paraméter, ami a kimenet változékonyságát biztosítja. Az eltérést ez egyes módszerek között ez a plusz paraméter adja.

### 2.2. Hasítófüggvény és időalapú OTP

A legkézenfekvőbb megoldás egy számláló alkalmazása (HOTP<sup>11</sup> – RFC 4226[3]), amit minden használat után megváltoztatunk – pl. növeljük eggyel – így biztosítva az OTP nehéz megjósolhatóságát (tehát nem egyediségét!). A HMAC kimenetét – ami a hash függvénytől függően lehet 160–512 bit (49–155 decimális számjegy) – nem közvetlenül alkalmazzuk. A sok karakter helyes beírásának megkövetelése ugyanis a használatot ellehetetlenítené, ezért a HMAC kimenetének egy dinamikusan csonkolt változatával (32 bit – 10 számjegy) dolgozunk.

A HOTP hátránya, hogy nem csak a titkot kell tárolnunk, hanem a számlálót is, amit a két oldalon szinkronban kell a tartani. Aszinkronitás esetén az azonosítás sikertelen lesz. Kézenfekvő megoldás a tárolás, és az egymással szinkronban tartásának elkerülésére,

<sup>8</sup> cryptographic hash

<sup>9</sup> cryptographic salt / pepper

<sup>10</sup> hash-based message authentication code (HMAC)

<sup>11</sup> HMAC-based One-Time Password (HOTP)

a számláló helyett az idő alkalmazása (TOTP<sup>12</sup> – RFC 6238[4]). A teljesen azonos idő persze csak illúzió, de ebben a felállásban elég ha kellő tűréssel azonos.

### 2.3. HOTP-algoritmus paraméterek

Az inicializáció során kapott QR-kódban lévő URI[5] a következő HOTP-algoritmus paramétereket (alapértelmezések félkövéren kiemelve) tartalmazza:

1. **titok (kulcs):** véletlen bitsorozat
  - a. mindkét fél ismeri
  - b. a HMAC egyik bemenete
2. **hasítófüggvény:** a HMAC paramétere (pl.: SHA-1, SHA-256, ...)
3. **időkorlát:** ezen belül az OTP nem változik (pl.: 30, vagy 60 másodperc), biztosítva
  - a. a megadáshoz egy időkeretet
  - b. sebességkorlátozás<sup>13</sup> egy esetleges támadás esetén
4. **számjegyek száma:** a HMAC dinamikusan rövidített változatát 10 – a számjegyek számának (pl.: 6–8) megfelelő – hatványával maradékosan osztjuk

## 3. Biztonsági problémák

### 3.1. Titok

Két jelentőséggel bíró ismérve van:

1. **Hossz:** ami meghatározó egy nyers erővel történő támadás esetén. Ha a támadónak sikerül megszereznie néhány TOTP-értéket a hozzájuk tartozó idővel együtt, akkor kisebb méretű titok esetében lehetőség nyílna a titok minden értékének végigpróbálására. Mivel a HMAC számolása nem igazán erőforrás-igényes, illetve azt a titok hossza érdemben nem befolyásolja, nincs praktikus ok kisebb kulchosszak alkalmazására, főként hogy a titok méretének egy bittel történő növelése a nyers erővel történő törés teljesítményigényét duplázza.
2. **Minőség:** azaz a megfelelő entrópia. Ma általunk nem ismert olyan támadás, ami lehetővé tenné a TOTP feltörését 80 bit méretű kulcs esetén, de a klasszikus mérnöki szakmákban megszokott túlbiztosítás itt is indokolt. A mérnökök az anyaghibák (és használati előírások be nem tartása), az informatikusok az entrópia csökkenését okozó hibák miatt kell, hogy túlméretezzenek. Adott esetben a kulcs hossza ugyan megfelel az elvárásoknak (pl.: >128 bit), de az entrópia csökkenése

---

<sup>12</sup> Time-based One-Time Password (TOTP)

<sup>13</sup> rate-limiting

(pl.: hibás véletlenszám-generátor) miatt annak tényleges biztonsága<sup>14</sup> a vártnál/kívántnál kisebb lehet. A környezet változása – mint a processzorok sebességének folyamatnövekedése, hatékonyabb algoritmusok, vagy technológiák megjelenése – szintén jelent kockázatot.

Fentiek miatt logikus, hogy a vonatkozó RFC már 2005-ben elvárta (MUST) a 128 bitnyi kulcshosszt, és ajánlotta (RECOMMEND) a 160 bites kulcsok alkalmazását<sup>15</sup>. Az [NIST SP](#)<sup>16</sup> 800-57 első változata már 2006-ban 80 bitnyi kulcshossz meglétét írta elő<sup>17</sup> 2010 előtti használat esetén, míg 2030-ig 112, utána pedig 128 bitet vár el. Ezen értékek az aktuális kiadásban (Rev. 5[6]) változatlanok. Kulcsok esetén a kulcshossz és a bitben vett biztonság azonos, vagyis az elvárás a minimum 112 bites kulcsok használata. Ha egy szolgáltatást 2030-at követően is terveznek használni (ennek 2022-ben, mikor a közigazgatás számára<sup>18</sup> az Ügyfélkapu+ megvalósították realitásnak kellett lennie), célszerűbb lett volna a 80 bit helyett már induláskor legalább 128, de inkább 160 bites, vagy még hosszabb kulcsokat használni, betartva az ajánlást és elkerülendő a “közeli” nemmegfelelőséget.

### 3.2. Hasítófüggvény

A TOTP leírása csak az SHA-1 algoritmus használatát taglalja, bár az SHA-2 algoritmus ismertetője már az RFC megjelenése előtt 3 évvel szerepelt egy kiadványban (FIPS 180-2). A HOTP leírása nem csak az SHA-1, de az SHA-2 algoritmusok használatát is engedi (MAY), igaz nem kívánja meg az SHA-2 preferálását az SHA-1 algoritmussal szemben. Ezek fényében nem meglepő, hogy az URI-formátum definiálásakor is az SHA-1 lett az alapértelmezett érték.

Tudható ugyanakkor, hogy az SHA-1 kapcsán 2017-ben napvilágot látott egy támadás (SHattered attack), ami kifejezetten hatékony, emiatt közvetlenül már nem használjuk ezt a kriptográfiai hasítófüggvényt (pl.: digitális aláírás). Ám a mai tudásunk szerint ez nem jelent kockázatot, amennyiben közvetve, a HMAC-algoritmus részeként alkalmazzuk, mivel HMAC nem érzékeny az olyan ütközéses támadásokra, mint a SHattered. Ezzel együtt az NIST vonatkozó dokumentuma (SP 800-131A Rev. 3[7]) 2030-ig elavultnak<sup>19</sup> tekin-

---

14 security strength

15 A szerzők egyikének tájékoztatása szerint egy sejtés, melyet alaposan nem vizsgáltak, is okozta, hogy a 160 bit lett az ajánlott, mert ez a hossz már annak helyessége esetén is elégségesnek tűnt..

16 Special Publication

17 5.6.2 Defining Appropriate Algorithm Suites

18 egy közigazgatás emberöltőkben (ma már 50-100 évben kell(ene), hogy gondolkodzon).

19 deprecated

ti, ezt követően nem engedélyezi<sup>20</sup> az SHA-1 alkalmazását HMAC esetén, tehát 2022-ben egy nehezen magyarázható döntés volt a közigazgatásban a “közeli” tiltás miatt az SHA-1 használata.

### 3.3. Időkorlát

Ez határozza meg, hogy egy adott TOTP-kód meddig érvényes. Ez idő alatt az érték nem változhat, mivel csak így valósítható meg az ellenőrzés. Ez úgy biztosítható, hogy az TOTP értékének számításakor az aktuális idő<sup>21</sup> – az időkorlát értékének megfelelő – alsó egész részét használjuk a HMAC bemenetével.

A valóságban a két fél órája egyaránt lehet pontatlan, és a hálózati átvitel ideje sem kalkulálható. Ennek kezelésére az RFC értelmében az ellenőrző fél korábbi és későbbi értékeket is elfogadhat, bár ajánlott (RECOMMEND), hogy az időablak mérete ne legyen nagyobb egynél (az időablak 1 értéke is azt jelenti, hogy 3 db olyan TOTP van, amit az ellenőrzőnek el kell fogadni, ami 2-es időablak esetén már 5 db lenne).

A biztonságot javító előírás az RFC-ben, hogy a sikeres azonosítást után az adott TOTP-érték többször már nem fogadható el<sup>22</sup>.

### 3.4. Számjegyek száma

A számjegyek száma nagy mértékben befolyásolja a TOTP támadhatóságát. Ha kisebb a számjegyek száma akkor kevesebb azon variációk száma is, amit egy támadónak ki kell próbálnia egy sikeres támadáshoz. Az időkorlát szintén módosítja a támadó lehetőségeit; az alapértelmezett értékek mellett (6 számjegy, 30 s) például 1 millió variációt kell végigpróbálni, ami ~33 ezer próbálkozást jelent másodpercenként. Ha a szomszédos időablakokban érvényes OTP is elfogadandó az harmadolja az értéket, tehát ~11 ezerszer kell próbálkozni a támadónak másodpercenként.

Belátható, hogy a próbálkozások sebességének korlátozása mindenképpen indokolt az ellenőrző oldalán, és az időkorlát is inkább csökkentendő, míg a számjegyek száma inkább növelendő. A számjegyek számának növelésének van a humán korlátja, mivel az az elgépelek valószínűségét növeli, tehát bizonyos számjegyszám felett egyre több ember számára nehézkesen használható (az ismételt, de inkább a többszöri próbálkozásnak is hasznos elférnie az időablakban).

---

<sup>20</sup> disallowed

<sup>21</sup> az 1970. 01. 01. 00:00 óta eltelt másodpercek száma

<sup>22</sup> Az előadás óta felfedeztük, hogy az Ügyfélkapu+ TOTP-szerű szolgáltatása ezt sem tartja be.

## 4. Ügyfélkapu+

A rendszer a TOTP beállításakor az alábbi paramétereket tartalmazó QR-kódot mutat fel:

- **Titok:** 80 bit
- **HMAC:** SHA-1
- **Időkorlát:** 30 s
- **Számjegyek száma:** 6

Látható, hogy olyan esetben, ahol a paraméternek van alapértelmezett érték, ott a szolgáltatás azt használja, illetve a titok hossza 80 bit.

Az eddigiek alapján két beállítás lehet problémás.

1. **Titok hossza:** A 80 bites érték nem jelent általunk ismert kihasználható sérülékenységet, de az NIST megfeleléségi szempontból problémás lehet, amennyiben az Ügyfélkapu+-ra az valamilyen okból vonatkozik, illetve sem az RFC ajánlásainak, sem a mérnöki alapvetésként szanált túlméretezésnek nem felel meg.
2. **HMAC:** az NIST-ajánlás, illetve a mérnöki túltervezés kapcsán hasonló a helyzet.

A vizsgálat eddig nem publikált része nem csak a szolgáltatás által ajánlott paraméterekre, de az Android platform legnépszerűbb – valamint a [NISZ saját fejlesztésű](#) – TOTP alkalmazásainak képességeire is kiterjedt. A tanulsága kettős:

1. A legnépszerűbb alkalmazások implementációja is hiányos.
  - a [Google Authenticator](#) nem támogatja az alapértelmezettől való eltérést az időkorlátnál, a [Microsoft Authenticator](#) pedig egyik paraméter esetében sem.
  - A Microsoft eszközt használva a HMAC okozta kockázat nem mitigálható.
  - A NISZ alkalmazása teljes körű támogatást ad, ennek használata mellett minden paraméter maximális biztonságú lehet.
2. Az esetleg kockázatot hordozó beállításokra egyedül a FreeOTP figyelmeztet, de magyarázat nélkül teszi.

## 5. Konklúzió

Az Ügyfélkapu+, illetve tágabb értelemben véve a TOTP mint biztonsági szolgáltatás vizsgálata több tanulsággal is járt.

1. A klasszikus mérnöki szakmák gondolkodásmódja a szabványokhoz, ajánlásokhoz való viszonya, a túlbiztosításhoz, a jövőállósághoz való igénye nem mutatkozik meg az informatika, illetve az informatikai biztonság területén. Erre adekvát példa a hajlandóság arra, hogy biztonsági funkciókat rontsunk (HMAC), vagy implementációs hiányosságokat tűrjünk (időkorlát, számjegyek száma) még a

legnagyobb gyártók (Google, Microsoft) alkalmazásai esetén is. A kompatibilitás (felesleges) fenntartása miatt, olyan algoritmusokat, paramétereket alkalmazunk, melyek nem jövőállóak (pl.: SHA-1), vagy felesleges kockázatot hordoznak (80 bites titok), úgy, hogy ezt sem teljesítményproblémák, sem a felhasználói igények nem indokolják, sőt még a vonatkozó ajánlásokkal is ellentétesek.

2. A nem hiába emlegetett biztonságtudatosság támogatása sem történik meg, lévén a TOTP-alkalmazások – egy kivételével (FreeOTP) – nem figyelmeztetik felhasználóikat az alkalmazott algoritmusok, paraméterek esetleges hiányosságaira, potenciálisan hozzájárulva egy hamis biztonságérzet kialakulásához.
3. Feleslegesen történnek implementációk, hiszen adottak olyan szabadon, kötetmek és költségek nélkül használható szoftverek, melynél adott esetben a márkaváltás<sup>23</sup> is megoldható, ha szükséges, de javítások, fejlesztések mindenképp eszközölhetőek. Jelen esetben ez a szoftver a FreeOTP, ami nemcsak hogy szabad szoftver, de teljes funkcionalitást ad, és egy neves gyártó (RedHat) áll mögötte.

## 6. Hivatkozások

- [1] P. Balsai and S. Pfeiffer, “Az ügyfélkapu plusz technológiája és biztonsága,” Zenodo, Jan. 2025.
- [2] S. Pfeiffer, “Miért ne becsüljük le a kisbetűs jelszavakat?,” Bitport. Dec. 2025.
- [3] D. M’Raihi, M. Bellare, F. Hoornaert, D. Naccache, and O. Ranen, “HOTP: An hmac-based one-time password algorithm,” RFC Editor, Dec. 2005.
- [4] D. M’Raihi, S. Machani, M. Pei, and J. Rydell, “TOTP: Time-Based one-time password algorithm,” RFC Editor, May 2011.
- [5] İ. Y. Eroğlu, “Usage specification of the otpauth URI format for TOTP and HOTP token generators,” IETF Datatracker.
- [6] E. Barker, “Recommendation for key management;,” National Institute of Standards and Technology, Gaithersburg, MD, May 2020.
- [7] E. Barker and A. Roginsky, “SP 800-131A Rev. 3, Transitioning the Use of cryptographic algorithms and key lengths,” CSRC.

---

23 rebranding