Szyfrowanie end-to-end, wyjaśnione naprawdę

Co mówią dostawcy, gdy mówią E2EE, a czego nie mówią. Dydaktyczne wyjaśnienie mechanizmu i jego ograniczeń, bez marketingowej otoczki.

Aby było jasne: WhatsApp mówi, że Twoje wiadomości są szyfrowane end-to-end. To prawda — i to nie wystarczy. Jeśli kopia zapasowa trafia do iCloud lub Google Drive bez dodatkowego szyfrowania, szyfrowanie zostaje złamane na Twoim własnym telefonie. Pytanie operacyjne nie brzmi, czy to jest szyfrowane, ale gdzie znajdują się klucze.

Co naprawdę oznacza szyfrowanie

Szyfrowanie wiadomości polega na przekształceniu jej w coś, co dla każdego, kto nie posiada określonej informacji zwanej kluczem, wygląda jak szum. Operacja ta odbywa się na urządzeniu nadawcy, a przy użyciu odpowiedniego klucza zostaje odwrócona na urządzeniu odbiorcy. W międzyczasie wiadomość podróżuje jako ciąg bajtów bez widocznego znaczenia. To prosta idea. Reszta artykułu zajmuje się niuansami, które czynią z niej, zależnie od przypadku, realną gwarancję lub tylko marketingową etykietę.

Przymiotnik *end-to-end* — po polsku *od końca do końca*, w skrócie E2EE — dodaje precyzji. Szyfrowanie nie jest wykonywane po to, aby serwer pośredniczący mógł je odczytać i dostarczyć. Jest wykonywane po to, aby tylko oba końce — urządzenie nadawcy i urządzenie odbiorcy — posiadały klucz. Każdy serwer, przez który przechodzi wiadomość, widzi szum, a nie wiadomość. Na tym polega techniczna różnica w stosunku do szyfrowania *w transicie*, w którym treść podróżuje zaszyfrowana od jednego serwera do drugiego, ale każdy serwer, przez który przechodzi, odszyfrowuje ją w celu dalszego przesłania, tymczasowo przywracając tekst jawny.

Paradoks współdzielonego sekretu

Istnieje oczywisty problem. Aby dwie osoby mogły szyfrować i odszyfrowywać wiadomości między sobą, obie potrzebują tego samego klucza. Ale jak uzgodnić ten klucz, jeśli wszystko, co sobie przesyłają, z definicji przechodzi przez kanał, na którym ktoś mógłby podsłuchiwać? Uzgodnienie klucza w tym samym kanale, w którym będzie on później używany, wydaje się niemożliwe: jeśli atakujący usłyszy go podczas uzgadniania, będzie mógł odszyfrować wszystko, co nastąpi później. Przez dziesięciolecia klasyczna kryptografia rozwiązywała to w twardy sposób: klucze były przekazywane osobiście, przed rozpoczęciem użytkowania, podczas fizycznych spotkań. Ambasadorzy nosili teczki z kluczami wszytymi w podszewkę płaszcza.

We współczesnej poczcie elektronicznej takie rozwiązanie się nie skaluje. Gdybyśmy musieli iść fizycznie do domu każdej osoby, z którą zamierzamy komunikować się w sposób zaszyfrowany, nie zdążylibyśmy z nikim porozmawiać. Pytanie postawione pięćdziesiąt lat temu przez społeczność kryptograficzną brzmiało: czy jest możliwe, aby dwie osoby, które się nie znają i które dzielą tylko kanał publiczny, uzgodniły w tym samym kanale publicznym sekret, którego nikt podsłuchujący kanał nie może poznać?

Elegancja Diffie-Hellman

W 1976 roku dwaj matematycy, Whitfield Diffie i Martin Hellman, udowodnili coś pozornie niemożliwego: że dwie osoby, rozmawiające tylko przez kanał publiczny — kanał, na którym każdy może usłyszeć wszystko, co mówią — mogą uzgodnić tajne hasło, nie dając żadnemu słuchaczowi możliwości jego odkrycia. Brzmi jak magia. To nie magia: to matematyka. Wymiana kluczy Diffie-Hellman, jak jest od tego czasu znana, stanowi podstawę praktycznie całej szyfrowanej komunikacji w internecie, a pół wieku intensywnego użytkowania i światowej analizy akademickiej potwierdzają jej solidność. Kto chce poznać intuicję wizualną lub matematykę, może czytać dalej. Kto woli ufać, że to działa, może również kontynuować, nie tracąc wątku artykułu.

Dla tych, którzy chcą to sobie wyobrazić, istnieje znana analogia z kolorami. Wyobraź sobie, że Alicja i Bruno uzgadniają publicznie kolor bazowy — powiedzmy żółty — na oczach podsłuchującej ich Ewy. Każde z nich wybiera prywatnie drugi, sekretny kolor i miesza swój sekret z żółtym. Alicja otrzymuje specyficzny pomarańczowy, Bruno otrzymuje specyficzny zielony. Wymieniają się wynikami na oczach Ewy. Teraz każde z nich miesza otrzymany kolor z własnym sekretem i oboje dochodzą do tego samego koloru końcowego, ponieważ kolejność mieszania nie ma znaczenia. Ewa widziała żółty i dwie mieszanki pośrednie, ale nie sekrety; bez żadnego z sekretów nie może dojść do koloru końcowego. Prawdziwa matematyka zastępuje kolory potęgowaniem w grupach modularnych lub na krzywych eliptycznych, ale idea jest ta sama: współdzielony sekret jest budowany publicznie, bez możliwości jego zrekonstruowania przez kogokolwiek w kanale.

W arytmetyce, dla tych, którzy wolą zobaczyć mechanizm: Alicja wybiera tajną liczbę a, Bruno wybiera b. Wymieniają g^a i g^b otwarcie w kanale. Alicja oblicza $(g^b)^a$, a Bruno oblicza $(g^a)^b$; oboje dochodzą do tego samego g^{ab}. Ewa widzi g, g^a i g^b przechodzące przez kanał, ale odzyskanie a z g^a — tak zwany problem logarytmu dyskretnego — wymaga czasu obliczeniowego astronomicznie dłuższego niż wiek wszechświata, gdy g zostanie wybrane w odpowiedniej grupie matematycznej.

Dla tych, którzy chcą to sprawdzić na małych liczbach. Wymianę Diffie-Hellman można prześledzić w całości na liczbach na tyle małych, by wykonać obliczenia ręcznie. Kto woli nie wchodzić w arytmetykę, może pominąć ten blok bez utraty wątku artykułu; kto chce zobaczyć działanie mechanizmu krok po kroku, znajdzie to tutaj. **Zasady publiczne**, które każdy może przeczytać: liczba pierwsza $p = 11$ (w prawdziwym Diffie-Hellman ma około trzystu cyfr; używamy jedenastu, aby obliczenia zmieściły się na jednej stronie), baza $g = 2$, oraz konwencja, że cała arytmetyka wykonywana jest *modulo p* — obliczasz, dzielisz przez p i zatrzymujesz resztę, jak zegar z jedenastoma pozycjami, który wraca do zera po przekroczeniu dziesiątki. **Prywatne wybory**, po jednym dla każdego i nigdy nieudostępniane: Alicja wybiera $a = 4$. Bruno wybiera $b = 7$.

Krok 1. Alicja oblicza $2^4 = 16$, następnie $16 \bmod 11 = \mathbf{5}$. Wysyła piątkę. Ewa ją zapisuje.

Krok 2. Bruno oblicza $2^7 = 128$, następnie $128 \bmod 11 = \mathbf{7}$. Wysyła siódemkę. Ewa również ją zapisuje. Po dwóch wysyłkach notatnik Ewy zawiera cztery dane: $p = 11$, $g = 2$, A = 5, B = 7. Brakuje jej wspólnej liczby, którą Alicja i Bruno właśnie zamierzają wyprowadzić — a której Ewa nie będzie w stanie zrekonstruować.

Krok 3. Alicja bierze siódemkę, którą wysłał jej Bruno, i podnosi ją do swojej prywatnej potęgi $a = 4$. Aby uniknąć operowania na $7^4 = 2401$, obliczenia wykonuje się w częściach, stosując modulo na każdym kroku:

$7^2 = 49$

$49 \bmod 11 = 5$

$7^4 = (7^2)^2 = 5^2 = 25$

$25 \bmod 11 = 3$

Alicja otrzymuje liczbę **3**.

Krok 4. Bruno bierze piątkę, którą wysłała mu Alicja, i podnosi ją do swojej prywatnej potęgi $b = 7$. Ponownie w częściach:

$5^2 = 25 \bmod 11 = 3$

$5^4 = (5^2)^2 = 3^2 = 9 \bmod 11 = 9$

$5^6 = 5^4 \times 5^2 = 9 \times 3 = 27 \bmod 11 = 5$

Ostatecznie $5^7 = 5^6 \times 5 = 5 \times 5 = 25 \bmod 11 = \mathbf{3}$.

Bruno również otrzymuje **3**.

Oboje dotarli do tej samej liczby, 3, pracując równolegle. Żadne z nich w żadnym momencie nie wysłało swojej prywatnej potęgi. Alicja nie wie, że $b = 7$; Bruno nie wie, że $a = 4$. Każde z nich użyło wartości publicznej wysłanej przez drugie, połączonej z własną prywatną potęgą, i spotkali się w tym samym celu. **Dlaczego dochodzą do tej samej liczby?** Co każde z nich obliczyło: Alicja, $(g^b)^a = 2^{7 \times 4} = 2^{28} \bmod 11$. Bruno, $(g^a)^b = 2^{4 \times 7} = 2^{28} \bmod 11$. To ta sama wielkość, ponieważ kolejność mnożenia potęg nie ma znaczenia ($7 \times 4 = 4 \times 7$). Każde dotarło do tego samego celu inną drogą.

A co z Ewą? Ma w swoim notatniku $p = 11$, $g = 2$, A = 5, B = 7 i chciałaby uzyskać 3. Aby to obliczyć, musiałaby znać a lub b — ale żadne z nich nie podróżowało przez kanał. Jedynym jej sposobem jest zadać sobie pytanie: «dla jakiej potęgi a zachodzi $2^a \bmod 11 = 5$?». Przy tak małym p może wypróbować 0, 1, 2, 3, 4... i znaleźć to w mniej niż minutę. Ale po zastąpieniu 11 liczbą pierwszą o trzystu cyfrach, przestrzeń możliwych potęg ma więcej elementów niż jest atomów w obserwowalnym wszechświecie. **Obecnie nie istnieje żaden znany ludzkości algorytm, który mógłby przeszukać tę przestrzeń w czasie krótszym niż miliardy lat.** To tak zwany *problem logarytmu dyskretnego*: łatwy w przód, obliczeniowo niemożliwy wstecz. I to jest powód, dla którego szyfrowanie opiera się, nawet jeśli Ewa śledziła całą rozmowę litera po literze.

Trzy proste składniki — arytmetyka zegarowa, potęgowanie i przemienność mnożenia ($a \cdot b = b \cdot a$) — połączone ze sobą tworzą protokół, od którego połowa ludzkości zależy każdego dnia w swojej prywatnej komunikacji. Żaden z tych trzech elementów z osobna nie wydaje się wyjątkowy. Decydujące jest ich złożenie.

Od Diffie-Hellman do protokołu Signal

Szyfrowanie end-to-end, z którego korzystają dzisiejsze profesjonalne aplikacje do przesyłania wiadomości, opiera się prawie bez wyjątku na eleganckiej i wzmocnionej wersji wymiany Diffie-Hellman. Protokołem odniesienia jest Signal, zaprojektowany przez Trevora Perrina i Moxie Marlinspike'a w latach 2013-2016. Łączy on dwie kluczowe idee. Pierwszą jest wymiana kluczy na krzywych eliptycznych (X25519), która generuje początkowy współdzielony sekret między dwoma urządzeniami. Drugą jest tak zwany Double Ratchet — podwójna zapadka — który automatycznie odnawia klucze przy każdej wiadomości, dzięki czemu przejęcie urządzenia dzisiaj nie pozwala na odszyfrowanie wiadomości z przeszłości ani wiadomości z przyszłości po obróceniu zapadki.

W Zig wymiana X25519, która generuje współdzielony sekret między dwoma urządzeniami, mieści się w sześciu liniach, przy użyciu biblioteki standardowej:

```
const std = @import("std");
const X25519 = std.crypto.dh.X25519;
```

```
// Alicia y Bruno generan cada uno un par (privada, pública).
const par_alicia = X25519.KeyPair.generate(io);
const par_bruno  = X25519.KeyPair.generate(io);

// Cada parte recibe la clave pública de la otra y deriva el mismo secreto.
const secreto_alicia = X25519.scalarmult(par_alicia.secret_key, par_bruno.public_key) catch unreachable;
const secreto_bruno  = X25519.scalarmult(par_bruno.secret_key,  par_alicia.public_key) catch unreachable;
// secreto_alicia == secreto_bruno  (32 bytes)
```

Co dzieje się w tych sześciu liniach: Klucze publiczne podróżują otwarcie. Klucze prywatne nigdy nie opuszczają danego urządzenia. Każda ze stron wyprowadza ze swojego klucza prywatnego i klucza publicznego drugiej strony ten sam trzydziestodwubajtowy sekret, którego nikt w kanale nie może odzyskać. Sekret ten służy później jako ziarno do szyfrowania wymienianych wiadomości. Double Ratchet protokołu Signal dodaje stałą rotację tego materiału, aby przejęcie klucza w danym momencie nie zagroziło reszcie rozmowy.

A co dokładnie kryje się wewnątrz `std.crypto.dh.X25519`? Żadna ukryta magia. To dwie krótkie funkcje, które można przeczytać w całości w samej bibliotece standardowej języka Zig. Pierwsza z nich wyprowadza klucz publiczny z prywatnego — owo «g^a» z wymiany:

```
pub fn recoverPublicKey(secret_key: [secret_length]u8) IdentityElementError![public_length]u8 {
    const q = try Curve.basePoint.clampedMul(secret_key);
    return q.toBytes();
}
```

W języku artykułu: klucz prywatny jest «mnożony» — w sensie eliptycznym, a nie elementarnej arytmetyki — przez punkt bazowy krzywej Curve25519, a wynik jest serializowany do trzydziestu dwóch bajtów. Operacja `clampedMul` to wzmocniona wersja tego mnożenia skalarnego: zawiera zabezpieczenia, które społeczność kryptograficzna dodawała przez lata, aby oprzeć się znanym rodzinom ataków. Dwie linijki ciała funkcji.

Druga funkcja łączy Twój klucz prywatny z kluczem publicznym, który przesyła Ci druga strona. To «$(g^b)^a$» z wymiany, które generuje trzydziestodwubajtowy współdzielony sekret, którego żadne z was nigdy nie transmitowało:

```
pub fn scalarmult(secret_key: [secret_length]u8, public_key: [public_length]u8) IdentityElementError![shared_length]u8 {
    const q = try Curve.fromBytes(public_key).clampedMul(secret_key);
    return q.toBytes();
}
```

Kolejne dwie linijki. Otrzymany klucz publiczny jest interpretowany jako punkt na krzywej i «mnożony» przez własny klucz prywatny. Dzięki przemienności operacji na krzywej — analogicznej do przemienności mnożenia potęg, którą widzieliśmy w przykładzie numerycznym — obie strony kończą z tym samym zserializowanym punktem: dokładnie tym współdzielonym sekretem, o którym mowa w artykule.

To wszystko. To, co w aplikacji wygląda jak magia, to w rzeczywistości dwie funkcje po trzy linijki każda. Techniczna złożoność koncentruje się w jednej operacji, `clampedMul`, która jest napisana w dalszej części tej samej biblioteki standardowej, rewidowana od dziesięcioleci przez międzynarodową społeczność kryptograficzną i dostępna dla każdego, kto chce ją przeczytać litera po literze. Nie ma tu czarnej skrzynki — ani w naszej aplikacji, ani w bibliotece standardowej Zig. Jest kod open source, który człowiek może zrozumieć, wybierając tempo, w jakim chce się w niego zagłębić.

Co chroni szyfrowanie end-to-end

To, co E2EE chroni dobrze, przy założeniu poprawnej implementacji, to treść wiadomości w transicie. Serwer pośredniczący, który odbiera i przesyła dalej zaszyfrowane dane, zobaczy ciąg niezrozumiałych bajtów. Atakujący z dostępem do kabla, routera czy punktu dostępowego wi-fi zobaczy to samo. Dostawca usługi, który przechowuje kopie ruchu, nie będzie mógł go odczytać post-factum. Rząd, który nakaże operatorowi usługi wydanie treści, otrzyma te same niezrozumiałe bajty, które serwer posiadał na początku.

To, w kategoriach praktycznych, bardzo dużo. To różnica między napisaniem listu wewnątrz nieprzezroczystej koperty a napisaniem go na pocztówce. Oba docierają. Tylko jeden chroni treść przed listonoszem.

Czego nie chroni szyfrowanie end-to-end

Warto wiedzieć to równie dobrze. E2EE nie chroni metadanych: serwer nadal wie, że użytkownik A wysyła dane do użytkownika B, o której godzinie, z jaką częstotliwością i skąd, nawet jeśli nie wie, co mówi. Te metadane, jak już argumentowaliśmy w *Szyfrowanie to nie prywatność*, są często bardziej ujawniające niż treść. Wiedza o tym, że ktoś dzwonił do kancelarii adwokackiej specjalizującej się w rozwodach w piątek o 22:00 na trzydzieści minut, opowiada historię, której treść rozmowy nigdy nie opowiedziała. To ta sama sytuacja, co widok osoby wchodzącej i wychodzącej kilka razy z kliniki onkologicznej: nie trzeba słyszeć nic z tego, o czym mówi się w środku, aby wyobrazić sobie, co się dzieje. Pojedynczy, odosobniony metadany może nic nie znaczyć; kilka skrzyżowanych ze sobą rysuje coś zbyt podobnego do prawdy. E2EE nie chroni końców: jeśli urządzenie odbiorcy zostanie zainfekowane przez złośliwe oprogramowanie, wiadomość zostanie normalnie odszyfrowana dla tego odbiorcy, a złośliwe oprogramowanie ją odczyta. E2EE nie chroni przed tożsamością samego rozmówcy: jeśli Alicja wierzy, że rozmawia z Brunonem, ale atakujący wstawia się na początku (tzw. *man in the middle*), a protokół nie obejmuje niezależnej weryfikacji, obie strony kończą rozmawiając z intruzem, myśląc, że rozmawiają ze sobą.

Jest czwarta rzecz, którą warto sformułować bez dwuznaczności. E2EE nie przeszkadza dostawcy, który twierdzi, że go oferuje, w przechowywaniu dodatkowo kopii niezaszyfrowanej wiadomości we własnych systemach. Stwierdzenie «moje wiadomości są szyfrowane end-to-end» oraz stwierdzenie «dostawca nie przechowuje moich treści» nie są tożsame. Aplikacja może spełniać pierwsze, naruszając jednocześnie

drugie; widzieliśmy to w nagłówkach prasowych wielokrotnie od 2018 roku. Użytkownik, o ile kod klienta nie jest weryfikowalny, nie ma technicznej możliwości odróżnienia jednego przypadku od drugiego bez ekspertyzy. Najbardziej znany przypadek wśród szerokiej publiczności: WhatsApp szyfruje wiadomości end-to-end w transicie, ale jeśli użytkownik aktywuje kopię zapasową w iCloud lub Google Drive bez dodatkowego szyfrowania, kopia ta jest przechowywana w formie czytelnej w infrastrukturze strony trzeciej, a szyfrowanie zostaje przełamane na końcu samego użytkownika.

Pytanie, którego operator nie chce słyszeć

Aplikacja, która twierdzi, że szyfruje end-to-end, może technicznie zrobić jedną z trzech rzeczy w odniesieniu do kluczy:

1. **Klucze rezydują tylko na urządzeniach.** Są generowane i rezydują wyłącznie na urządzeniach użytkowników; operator ich nie zna ani nie przechowuje. To przypadek optymalny.
2. **Operator może mieć dostęp, jeśli zechce.** Operator posiada klucze użytkowników (lub może je wygenerować według uznania) i przechowuje je w swoich bazach danych. Jeśli zechce lub zostanie do tego zmuszony, może odczytać treść. Tak jest w przypadku większości usług „chmurowych”.
3. **Operator nie może mieć dostępu z założenia, ale kontroluje dostęp.** Operator nie posiada kluczy, ale ma kontrolę nad aplikacją, która je generuje. W razie przymusu może wysłać złośliwą aktualizację, która przechwyci klucze lub treść przed zaszyfrowaniem. Tak jest w przypadku wielu komercyjnych usług E2EE.

Pytanie operacyjne nie brzmi zatem, czy coś jest zaszyfrowane, ale kto ma kontrolę nad urządzeniem i oprogramowaniem zarządzającym kluczami. W Solo2 klucze znajdują się wyłącznie w Twoim Skarbcu (IndexedDB zaszyfrowana Twoim hasłem), a oprogramowanie to weryfikowalny kod open source.

Dla czytelnika profesjonalnego

Szyfrowanie end-to-end to narzędzie cyfrowej suwerenności. Ale jak każde narzędzie, jego skuteczność zależy od ręki, która je dzierży, i gruntu, na którym stoi.

1. Gdzie są generowane klucze kryptograficzne i gdzie fizycznie rezydują? Jeśli operator ma do nich dostęp (nawet tymczasowo, nawet pod pozorem odzyskiwania), E2EE jest tylko nominalne.
2. Czy istnieje niezależna weryfikacja rozmówcy (numery bezpieczeństwa, kody QR, porównanie poza pasmem), która zapobiega atakowi man-in-the-middle podczas nawiązywania połączenia?
3. Czy kod klienta można audytować — jest otwarty, opublikowany, powtarzalny — czy też wymaga zaufania do słowa dostawcy o tym, co klient faktycznie robi?
4. Jakie metadane generuje i przechowuje usługa, i jak długo? Nawet jeśli treść jest nieprzejrzysta, metadane mogą zrekonstruować dużą część poufnych informacji.

Te cztery pytania nie wymagają zaawansowanych informacji technicznych; wymagają informacji, na które każdy uczciwy operator może odpowiedzieć w swojej publicznej dokumentacji. Jakość i precyzja odpowiedzi mówią o produkcie równie dużo, co sama odpowiedź.

Szyfrowanie end-to-end, dobrze wykonane, jest jedną z najdoskonalszych konstrukcji, jakie współczesna kryptografia przekazała do codziennej praktyki. Oryginalna idea — że dwie osoby mogą uzgodnić sekret w otwartym kanale — należy do Whitfield Diffie i Martin Hellman, 1976; pół wieku później nadal żyjemy w jej konsekwencji. Ale, jak w przypadku każdej obietnicy technicznej, jej wartość zależy od rzeczywistego spełnienia, a nie od etykiety. Pytanie uczciwego profesjonalisty nie brzmi «czy to jest zaszyfrowane?», lecz «kto ma klucze?». Odpowiedzi niosą ze sobą różne konsekwencje. Warto je znać.

Źródła i dodatkowa lektura

- Diffie, W.; Hellman, M. — *New Directions in Cryptography*, IEEE Transactions on Information Theory, listopad 1976. Fundamentalny artykuł o kryptografii klucza publicznego.
- Perrin, T.; Marlinspike, M. — *The Double Ratchet Algorithm*, publiczna specyfikacja Open Whisper Systems, rewizja 2016. Podstawa protokołu Signal i jego przemysłowych pochodnych.
- RFC 7748 — Elliptic Curves for Security (IETF, styczeń 2016). Normatywna specyfikacja krzywych X25519 i X448 używanych w nowoczesnych wymianach kluczy.
- Ferguson, N.; Schneier, B.; Kohno, T. — *Cryptography Engineering: Design Principles and Practical Applications* (Wiley, 2010). Rozdziały o wymianie kluczy i protokołach uwierzytelnionego szyfrowania.
- Rozporządzenie (UE) 2024/1183 w sprawie europeckich ram tożsamości cyfrowej (eIDAS 2) — ustanawia ramy, w których niezależna weryfikacja rozmówcy zyskuje wsparcie instytucjonalne, a rozróżnienie między szyfrowaniem nominalnym a rzeczywistym ma różne konsekwencje prawne.

← PoprzedniKill switch i instytucjonalne przejęcieNastępny →Model biznesowy jako sygnał zaufania

Ostatnie lektury

- Analiza · 18 maja 2026 Prywatność rzeczywista vs pozorna: pytania, które warto sobie zadać
- Analiza · 18 maja 2026 Self-hosting jako praktyka zawodowa
- Koncepcja · 18 maja 2026 24 słowa: czym jest tożsamość kryptograficzna

Zabierz ten artykuł tam, gdzie go potrzebujesz.

↓ Markdown ↓ Zwykły tekst ↓ PDF

Plik zostanie pobrany na Twoje urządzenie. Stamtąd możesz go zapisać, zaimportować do Solo2 lub udostępnić w dowolnym miejscu. Cuadernos nie decyduje o miejscu docelowym za Ciebie.

Pieczęć lakowa · SHA-256 d61bcaa648809bd0ab25c585901f1d5878944c3e1c37dcf31db7a301704f5747

Cuadernos Lacre · Publikacja Menzuri Gestión S.L. ·
napisana przez R.Eugenio · redagowana przez zespół Solo2.