Τι είναι πραγματικά το SHA-256

Ένα μαθηματικό αποτύπωμα που χωράει σε εξήντα τέσσερις χαρακτήρες και αλλάζει ολόκληρο αν μετακινηθεί έστω και ένα κόμμα από το πρωτότυπο κείμενο. Γιατί το ονομάζουμε ψηφιακή σφραγίδα με βουλοκέρι.

Η απλή ιδέα πίσω από το τεχνικό όνομα

Φανταστείτε ότι υπάρχει μια μηχανή με μία μόνο υποδοχή και μία μόνο οθόνη. Από την υποδοχή εισάγετε ένα κείμενο: μια λέξη, μια φράση, ένα ολόκληρο μυθιστόρημα. Στην οθόνη εμφανίζεται, λίγες στιγμές αργότερα, μια αλληλουχία ακριβώς εξήντα τεσσάρων χαρακτήρων. Αυτή την αλληλουχία, για τον επαγγελματία αναγνώστη την ονομάζουμε *hash* ή *κρυπτογραφική σύνοψη*· για τον γενικό αναγνώστη, μπορούμε να την ονομάσουμε προς το παρόν ένα μαθηματικό αποτύπωμα του κειμένου, όπως το δακτυλικό αποτύπωμα είναι για έναν άνθρωπο.

Εάν εισάγετε το ίδιο κείμενο δύο φορές, η μηχανή θα δείξει το ίδιο αποτύπωμα και τις δύο φορές. Εάν εισάγετε ένα κείμενο ελαφρώς διαφορετικό —ένα μόνο κόμμα μετατοπισμένο, ένα κεφαλαίο γράμμα που γίνεται πεζό— η μηχανή δείχνει ένα αποτύπωμα εντελώς διαφορετικό από το πρώτο. Όχι παρόμοιο: διαφορετικό. Αυτές οι δύο ιδιότητες μαζί —ο ντετερμινισμός και η ευαισθησία— είναι η απλή ιδέα. Όλα τα υπόλοιπα στο SHA-256 είναι ο μηχανισμός που τις κάνει να λειτουργούν σωστά.

Είναι καλό να πούμε από την αρχή τι δεν κάνει η μηχανή. Δεν κρυπτογραφεί το κείμενο. Δεν το αποκρύπτει. Δεν το αποθηκεύει. Η μηχανή κοιτάζει το κείμενο, υπολογίζει το αποτύπωμα και ξεχνά το κείμενο. Το αποτύπωμα δεν επιτρέπει την ανακατασκευή του κειμένου που το παρήγαγε· επιτρέπει μόνο, δεδομένου ενός υποψήφιου κειμένου, να ελέγξουμε αν συμπίπτει ή όχι με το πρωτότυπο. Γι' αυτό λέμε ότι είναι μια σύνοψη *μονής κατεύθυνσης*: πας, αλλά δεν επιστρέφεις.

Το hash δεν είναι το ίδιο με την κρυπτογράφηση

Η σύγχυση είναι συχνή και είναι καλό να την ξεκαθαρίσουμε: η κρυπτογράφηση και το hashing είναι διαφορετικές λειτουργίες. Η κρυπτογράφηση συνίσταται στον μετασχηματισμό ενός κειμένου έτσι ώστε μόνο ο κάτοχος του κλειδιού να μπορεί να το επαναφέρει στην αρχική του μορφή. Το hashing συνίσταται στην παραγωγή ενός αποτυπώματος του κειμένου από το οποίο το αρχικό κείμενο δεν μπορεί να ανακτηθεί ποτέ, ούτε με κλειδί ούτε χωρίς αυτό. Το πρώτο είναι αναστρέψιμο από σχεδιασμό· το δεύτερο, μη αναστρέψιμο από σχεδιασμό.

Η πρακτική συνέπεια έχει σημασία. Όταν μια εφαρμογή λέει «αποθηκεύουμε τον κωδικό πρόσβασής σας κρυπτογραφημένο», υπάρχει κάποιος που έχει το κλειδί για να τον αποκρυπτογραφήσει — η ίδια η εφαρμογή, σε κάθε περίπτωση. Όταν μια εφαρμογή λέει «αποθηκεύουμε τον κωδικό πρόσβασής σας με hashing», η ίδια η εφαρμογή δεν μπορεί να διαβάσει τον αρχικό κωδικό ακόμα κι αν το ήθελε· μπορεί μόνο να ελέγξει αν αυτό που γράφετε παράγει ξανά το ίδιο αποτύπωμα. Το δεύτερο μοντέλο, όταν γίνεται σωστά, είναι πολύ προτιμότερο από το πρώτο για την αποθήκευση κωδικών πρόσβασης. Στη συνέχεια θα δούμε γιατί το «σωστά» απαιτεί κάτι περισσότερο από το SHA-256 σκέτο.

Οι τέσσερις ιδιότητες που καθιστούν χρήσιμο ένα κρυπτογραφικό hash

Μια συνάρτηση hash που αξίζει το επίθετο *κρυπτογραφική* πληροί τέσσερις ιδιότητες:

1. **Ντετερμινισμός.** Η ίδια είσοδος παράγει πάντα το ίδιο αποτύπωμα.
2. **Φαινόμενο χιονοστιβάδας (Avalanche effect).** Μια μικρή αλλαγή στην είσοδο παράγει ένα εντελώς διαφορετικό αποτύπωμα, χωρίς ορατή ομοιότητα με το προηγούμενο.
3. **Αντίσταση στην αναστροφή.** Δεδομένου ενός αποτυπώματος, δεν είναι υπολογιστικά εφικτό να βρεθεί το κείμενο που το παρήγαγε.

4. **Αντίσταση σε συγκρούσεις.** Δεν είναι υπολογιστικά εφικτό να βρεθούν δύο διαφορετικά κείμενα που να παράγουν το ίδιο αποτύπωμα.

«Δεν είναι υπολογιστικά εφικτό» δεν σημαίνει «είναι μαθηματικά αδύνατο». Σημαίνει ότι το κόστος σε χρόνο, ενέργεια και χρήμα για να επιτευχθεί υπερβαίνει κατά τάξεις μεγέθους το σύνολο όλης της λογικά διαθέσιμης υπολογιστικής ικανότητας. Για το SHA-256, αυτό το όριο μετριέται σε χιλιάδες δισεκατομμύρια χρόνια ακόμα και για τις πιο αισιόδοξες προσεγγίσεις με εξειδικευμένο υλικό. Το οποίο, για τους πρακτικούς σκοπούς του αναγνώστη, είναι το ίδιο με το «δεν γίνεται».

Το SHA-256, συγκεκριμένα

Το όνομα τα λέει όλα. SHA είναι τα αρχικά του *Secure Hash Algorithm*: αλγόριθμος ασφαλούς κατακερματισμού. Ο αριθμός 256 υποδηλώνει το μέγεθος του αποτυπώματος σε bits: διακόσια πενήντα έξι bits, δηλαδή τριάντα δύο bytes, τα οποία εμφανίζονται σε δεκαεξαδική μορφή ως οι εξήντα τέσσερις χαρακτήρες που ο αναγνώστης ήδη γνωρίζει. Το πρότυπο δημοσιεύθηκε από το αμερικανικό NIST, τον οργανισμό που τυποποιεί αυτού του είδους τις συναρτήσεις, το 2001 ως μέρος της οικογένειας SHA-2· η τρέχουσα έκδοση του προτύπου, FIPS 180-4, είναι του 2015.

Για όσους δεν έχουν ακόμα υπόψη τους τι είναι τα bits και τα bytes:

```
1 bit    →  0 ή 1        (ένας διακόπτης: αναμμένος ή σβηστός)
1 byte   →  8 bits       (256 δυνατοί συνδυασμοί)
32 bytes →  256 bits     (το αποτύπωμα SHA-256)
```

Ο αριθμός 256 στο τέλος του ονόματος δηλώνει το μέγεθος του αποτυπώματος σε bits. Στο δεκαεξαδικό σύστημα —ένα σύστημα αρίθμησης με δεκαέξι σύμβολα αντί για δέκα— αυτά τα 256 bits χωρούν σε ακριβώς 64 χαρακτήρες. Αυτοί είναι οι 64 χαρακτήρες που βλέπετε στο κάτω μέρος κάθε Cuaderno.

Οι διαστάσεις αξίζουν μια στιγμή προσοχής. Διακόσια πενήντα έξι bits επιτρέπουν δύο εις την διακοσιοστή πεντηκοστή έκτη διαφορετικές τιμές: ένας αριθμός με εβδομήντα οκτώ δεκαδικά ψηφία, αρκετές τάξεις μεγέθους μεγαλύτερος από τον εκτιμώμενο αριθμό ατόμων στο παρατηρήσιμο σύμπαν. Κάθε κείμενο στον κόσμο —κάθε βιβλίο, κάθε email, κάθε μήνυμα— πέφτει πάνω σε μία από αυτές τις τιμές. Η πιθανότητα δύο διαφορετικά κείμενα να συμπίπτουν τυχαία είναι, για πρακτικούς σκοπούς, μη διακρίσιμη από το μηδέν.

Πώς φαίνεται σε κώδικα

Στην Zig, τη γλώσσα στην οποία γράφουμε τα κομμάτια που υποστηρίζουν το Solo2, ο υπολογισμός της σφραγίδας SHA-256 ενός κειμένου φαίνεται κάπως έτσι:

```
const std = @import("std");

const texto = "Cuadernos Lacre";
var resumen: [32]u8 = undefined;
std.crypto.hash.sha2.Sha256.hash(texto, &resumen, .{});
```

Μόλις ζητήσαμε από την τυπική βιβλιοθήκη της Zig να υπολογίσει το SHA-256 του κειμένου μέσα στα εισαγωγικά. Μετά την κλήση, η μεταβλητή *resumen* περιέχει τα τριάντα δύο bytes που αποτελούν τη σφραγίδα στην ακατέργαστη μορφή της· όταν εμφανίζονται στην οθόνη σε δεκαεξαδική μορφή, είναι οι εξήντα τέσσερις χαρακτήρες που εμφανίζονται στο κάτω μέρος αυτού του άρθρου. Εάν αλλάζαμε το *Cuadernos Lacre* σε *Cuadernos lacre* —ένα κεφαλαίο γράμμα λιγότερο— η σφραγίδα θα άλλαζε ολόκληρη. Αυτή είναι, σε πέντε γραμμές, η κεντρική ιδιότητα που υποστηρίζει τα υπόλοιπα. Για όποιον θέλει να δει πώς λειτουργεί εσωτερικά, στο τέλος του άρθρου περιλαμβάνουμε μια αναγνώσιμη έκδοση του αλγορίθμου με σχόλια βήμα προς βήμα.

Γιατί το ονομάζουμε σφραγίδα με βουλοκέρι

Στην ευρωπαϊκή αλληλογραφία από τον δέκατο πέμπτο έως τον δέκατο ένατο αιώνα, το βουλοκέρι σφράγιζε την επιστολή. Μια σταγόνα λιωμένο κερί, μια σφραγίδα που πιέζεται από πάνω, και η επιστολή έμενε μαρκαρισμένη με τρόπο ανεπανάληπτο. Δεν προστάτευε το περιεχόμενο από τον αποφασισμένο αδιάκριτο —το χαρτί μπορούσε να διαβαστεί στο φως, το βουλοκέρι μπορούσε να σπάσει— αλλά το αποδείκνυε. Οποιαδήποτε αλλοίωση του κλεισίματος ήταν ορατή στον παραλήπτη πριν καν ανοίξει το χαρτί. Το βουλοκέρι δεν εμπόδιζε τη ζημιά· την δήλωνε.

Το SHA-256 του σώματος κάθε Cuaderno επιτελεί την ίδια λειτουργία στην ψηφιακή του έκδοση. Εάν μια μόνο λέξη του άρθρου άλλαζε μεταξύ της στιγμής που δημοσιεύθηκε και της στιγμής που το διαβάζετε, η δεκαεξαδική σφραγίδα στο κάτω μέρος του κειμένου δεν θα συνέπιπτε πλέον με το SHA-256 του κειμένου που έχετε μπροστά σας. Οποιοσδήποτε αναγνώστης με πέντε γραμμές κώδικα θα μπορούσε να το ελέγξει. Η έκδοση δεν μπορεί να ξαναγράψει την ιστορία της χωρίς η σφραγίδα να την προδώσει. Δεν προστατεύει από τη ζημιά· την καθιστά επαληθεύσιμη.

Τι δεν είναι ένα hash

Τέσσερις χρήσεις ζητούνται μερικές φορές από το SHA-256 που δεν του αντιστοιχούν:

1. **Κρυπτογράφηση.** Ένα hash συνοψίζει· δεν αποκρύπτει. Εάν θέλετε το κείμενο να μην μπορεί να διαβαστεί, πρέπει να το κρυπτογραφήσετε, όχι να του κάνετε hashing.
2. **Αυθεντικοποίηση του συγγραφέα.** Ένα hash δεν λέει ποιος έγραψε το κείμενο, μόνο ποιο κείμενο υποβλήθηκε σε hashing. Για τη συσχέτιση πατρότητας χρειάζεται μια κρυπτογραφική υπογραφή πάνω από το hash, όχι το hash σκέτο.
3. **Αποθήκευση κωδικών πρόσβασης.** Εδώ υπάρχει μια παγίδα που είναι καλό να κατανοήσουμε. Το SHA-256 έχει σχεδιαστεί για να είναι πολύ γρήγορο —κάτι που είναι καλό για πολλά πράγματα, αλλά κακό για αυτό. Ένας εισβολέας με εξειδικευμένο υλικό μπορεί να δοκιμάσει δισεκατομμύρια κωδικούς πρόσβασης ανά δευτερόλεπτο έναντι ενός hash SHA-256 μέχρι να βρει τον δικό σας. Για την αποθήκευση κωδικών πρόσβασης πρέπει να χρησιμοποιούνται σκόπιμα αργές συναρτήσεις παραγωγής κλειδιών όπως οι Argon2, scrypt ή bcrypt, σε συνδυασμό με ένα *salt* (ένα μοναδικό τυχαίο δεδομένο ανά χρήστη, που εμποδίζει δύο άτομα με τον ίδιο κωδικό πρόσβασης να έχουν το ίδιο hash).
4. **Ανάγνωση του hash ως αναγνωριστικού του συγγραφέα.** Δεν είναι. Ένα hash αναγνωρίζει το περιεχόμενο. Εάν δύο άτομα κάνουν hashing τη λέξη *hola* με SHA-256, και οι δύο παίρνουν την ίδια σύνοψη — και αυτό είναι η κεντρική ιδιότητα, όχι ένα ελάττωμα: αν ήταν διαφορετικές συνόψεις, δεν θα μπορούσαμε να ελέγξουμε τη σύμπτωση μεταξύ αυτού που δημοσιεύθηκε και αυτού που παραλήφθηκε.

Πού εμφανίζεται το SHA-256 στην καθημερινότητά σας

Ακόμα κι αν δεν το βλέπετε, το SHA-256 υποστηρίζει ένα μεγάλο μέρος αυτών που χρησιμοποιείτε καθημερινά στο διαδίκτυο. Η αλυσίδα μπλοκ (blockchain) του Bitcoin χτίζεται συνδέοντας το SHA-256 κάθε μπλοκ με το επόμενο· η αλλοίωση ενός παρελθόντος μπλοκ αναγκάζει στον επανυπολογισμό ολόκληρης της μεταγενέστερης αλυσίδας. Το Git, το σύστημα με το οποίο γίνεται η διαχείριση εκδόσεων του κώδικα του μισού κόσμου, αναγνωρίζει κάθε commit από το SHA-256 (σε πρόσφατες εκδόσεις) ή από τον προκάτοχό του SHA-1 (σε παλαιότερες εκδόσεις) του πλήρους περιεχομένου του. Τα πιστοποιητικά HTTPS που επαληθεύουν την ταυτότητα ενός ιστότοπου όταν εισέρχεστε φέρουν ένα σχετιζόμενο αποτύπωμα SHA-256. Οι λήψεις λογισμικού συνοδεύονται συχνά από ένα SHA-256 που δημοσιεύεται από τον προγραμματιστή, ώστε να επαληθεύσετε ότι το αρχείο δεν αλλοιώθηκε στη διαδρομή. Και, όπως είπαμε, στο κάτω μέρος κάθε Cuadernos Lacre.

Για τον επαγγελματία αναγνώστη

Τέσσερις λειτουργικές υπενθυμίσεις για όποιον αποφασίζει ή ελέγχει συστήματα:

1. Το Hash δεν είναι κρυπτογράφηση. Εάν ένας πάροχος συγχέει τους δύο όρους στην τεχνική του τεκμηρίωση, είναι σκόπιμο να ρωτήσετε τι ακριβώς εννοεί.
2. Για την αποθήκευση κωδικών πρόσβασης δεν πρέπει ποτέ να χρησιμοποιείται το SHA-256 σκέτο. Το SHA-256 είναι πολύ γρήγορο για αυτή την εργασία (βλ. σημείο 3 του *Τι δεν είναι ένα hash*). Το τρέχον πρότυπο είναι το **Argon2id**: αργό από σχεδιασμό, παραμετροποιήσιμο ανάλογα με τη χωρητικότητα του διακομιστή, σε συνδυασμό με ένα διαφορετικό τυχαίο *salt* ανά χρήστη.
3. Για την ακεραιότητα εγγράφων —συμβόλαια, φάκελοι, αρχεία— το SHA-256 παραμένει το πρότυπο αναφοράς. Είναι αυτό που χρησιμοποιούν οι εγκεκριμένοι πάροχοι υπηρεσιών χρονοσήμανσης στην ΕΕ.
4. Για μακροχρόνια διατήρηση (δεκαετίες) είναι σκόπιμο να υπολογίζεται και να αρχειοθετείται επίσης ένα SHA-3 ή ένα SHA-512 μαζί με το SHA-256· η κρυπτογραφική σύνεση συνιστά να μην βασιζόμαστε σε μία μόνο συνάρτηση κατά τη διάρκεια αρχειοθετήσεων αιώνων.

Τεχνικά, αυτή η επαναληπτική δομή —όπου η ενδιάμεση κατάσταση διατηρείται μεταξύ των μπλοκ εισόδου— είναι γνωστή ως κατασκευή **Merkle-Damgård**, το πρότυπο στο οποίο βασίζονται οι SHA-1, SHA-2 (συμπεριλαμβανομένου

του SHA-256) και πολλές άλλες κλασικές συναρτήσεις κατακερματισμού. Ο SHA-3, αντίθετα, εγκαταλείπει το Merkle-Damgård υπέρ μιας διαφορετικής αρχιτεκτονικής που ονομάζεται *σφουγγάρι* (sponge).

Πώς λειτουργεί ο SHA-256, βήμα προς βήμα, με απλά λόγια

Φανταστείτε ότι έχετε στήσει το πιο περίπλοκο κύκλωμα ντόμινο στον κόσμο: χιλιάδες πλακίδια, δεκάδες διακλαδώσεις, μηχανικές γέφυρες και ράμπες που διασχίζουν ολόκληρο το δωμάτιο, τοποθετημένα προσεκτικά κομμάτι-κομμάτι.

Αν δώσετε ένα άγγιγμα στο πρώτο πλακίδιο, η αλυσίδα πέφτει σε μια ακριβή και επαναλήψιμη αλληλουχία. Ίδιο στήσιμο, ίδιο αρχικό άγγιγμα → πανομοιότυπο τελικό μοτίβο πεσμένων πλακιδίων, ξανά και ξανά.

Εδώ είναι το ενδιαφέρον μέρος: μετακινήστε **ένα μόνο πλακίδιο** μισό εκατοστό προς τη μία πλευρά πριν ξεκινήσετε και αγγίξτε το ξανά. Μια ράμπα που έπρεπε να ενεργοποιηθεί παραμένει αδρανής, μια γέφυρα δεν πέφτει, μια διαφορετική διακλάδωση πυροδοτείται. Το τελικό μοτίβο των πλακιδίων στο πάτωμα είναι εντελώς αγνώριστο σε σύγκριση με το πρώτο.

Ο SHA-256 είναι μαθηματικά αυτό το κύκλωμα. Το κείμενο που γράφετε είναι η αρχική θέση των πλακιδίων. Ο αλγόριθμος είναι το άγγιγμα που απελευθερώνει την καταρράκτη. Και το τελικό αποτέλεσμα —αυτό που ονομάζουμε *hash*— είναι η στατική φωτογραφία του δαπέδου όταν όλα έχουν σταματήσει. Αλλάξτε ένα μόνο κόμμα στο αρχικό κείμενο και η φωτογραφία θα είναι ριζικά διαφορετική. Τόσο απλά, και τόσο δραστικά.

Βήμα 1. Μετάφραση του κειμένου σε δυαδικά πλακίδια. Οι υπολογιστές δεν καταλαβαίνουν γράμματα· τα μεταφράζουν πρώτα σε αριθμούς (ASCII) και τους αριθμούς σε δυαδικούς (άσους και μηδενικά). Κάθε γράμμα γίνεται 8 λευκά ή μαύρα πλακίδια: το *A* είναι `01000001`, το *B* είναι `01000010`, το κενό είναι `00100000`. Ολόκληρο το κείμενό σας —μια λέξη, ένα συμβόλαιο, ένα μυθιστόρημα— γίνεται μια μακριά σειρά από λευκά και μαύρα πλακίδια.

Βήμα 2. Συμπλήρωση έως το τυπικό μέγεθος. Το κύκλωμα επεξεργάζεται τη σειρά σε *τμήματα* ακριβώς 512 πλακιδίων. Εάν το μήνυμά σας δεν φτάνει σε πολλαπλάσιο του 512, προστίθεται ένα πλακίδιο-δείκτης (αυτό με την τιμή `10000000`) αμέσως μετά το κείμενο και στη συνέχεια μηδενικά μέχρι να ολοκληρωθεί το τμήμα. Οι τελευταίες 64 θέσεις κάθε τμήματος προορίζονται για την καταγραφή του αρχικού μήκους του κειμένου. Έτσι, το κύκλωμα γνωρίζει πάντα πού τελείωσε το πραγματικό περιεχόμενο και πού άρχισε το γέμισμα.

Βήμα 3. Τοποθέτηση των οκτώ πλακιδίων-οδηγών. Πριν ξεκινήσουμε, τοποθετούμε στο τραπέζι **οκτώ πλακίδια-οδηγούς** (master tiles) σε μια ακριβή αρχική θέση. Αυτά τα οκτώ πλακίδια δεν είναι μυστικό: η αρχική τους τιμή καθορίζεται από έναν δημόσιο μαθηματικό κανόνα (τις τετραγωνικές ρίζες των πρώτων οκτώ πρώτων αριθμών —2, 3, 5, 7, 11, 13, 17, 19— και τα πρώτα bit του δεκαδικού μέρους κάθε ρίζας). Όλοι, σε οποιαδήποτε γωνιά του πλανήτη, ξεκινούν με τα ίδια οκτώ πλακίδια-οδηγούς στην ίδια θέση. Η μοίρα τους είναι να σπρώχνονται και να μεταμορφώνονται από τη χιονοστιβάδα.

Βήμα 4. Η μεγάλη χιονοστιβάδα: εξήντα τέσσερις γύροι ωθήσεων. Εδώ ξεκινά το θέαμα. Το πρώτο τμήμα 512 πλακιδίων του κειμένου σας συγκρούεται με τα οκτώ πλακίδια-οδηγούς. Αλλά δεν πέφτουν όλα μαζί: ο μηχανισμός εκτελεί **εξήντα τέσσερις διαδοχικούς γύρους**. Σε κάθε γύρο εκτελεί τρεις λειτουργίες με τα πλακίδια:

- **Το Καρουζέλ** (περιστροφή). Τα πλακίδια κινούνται κυκλικά: αυτά στα δεξιά περνούν στα αριστερά. Κανένα πλακίδιο δεν χάνεται ούτε προστίθεται· απλώς αναδιατάσσονται κάνοντας μια πλήρη στροφή στο καρουζέλ. Είναι ένας φθηνός και αναστρέψιμος τρόπος αναδιανομής των πληροφοριών.
- **Το Λογικό Χωνί** (XOR). Τα πλακίδια περνούν μέσα από ένα χωνί που τα συγκρίνει ανά δύο: αν και τα δύο έχουν το ίδιο χρώμα, βγαίνει ένα λευκό· αν είναι διαφορετικά, βγαίνει ένα μαύρο. Είναι η απλούστερη λειτουργία της δυαδικής λογικής, αλλά σε συνδυασμό με τις περιστροφές του καρουζέλ γίνεται εξαιρετικά ισχυρή για την ανάμειξη πληροφοριών χωρίς να τις χάσει.
- **Η Υπερχείλιση** (αρθρωτή πρόσθεση). Το αποτέλεσμα προστίθεται με ένα *πλακίδιο σταθερής ώθησης* που λαμβάνεται από μια δημόσια λίστα εξήντα τεσσάρων σταθερών (τις κυβικές ρίζες των πρώτων εξήντα τεσσάρων πρώτων αριθμών). Εάν το άθροισμα δημιουργήσει επιπλέον πλακίδια που δεν χωρούν στον προβλεπόμενο χώρο των 32 πλακιδίων, αυτά τα πλεονάζοντα πλακίδια απορρίπτονται. Το τραπέζι έχει χώρο μόνο για 32 πλακίδια, ούτε ένα παραπάνω.

Στο τέλος του γύρου εξήντα τέσσερα, κάθε ένα από τα πλακίδια του τμήματος του κειμένου σας έχει επηρεάσει τη θέση των οκτώ πλακιδίων-οδηγών. Η ενέργεια της ώθησης έχει ταξιδέψει σε ολόκληρο το κύκλωμα.

Βήμα 5. Προσθήκη του επόμενου τμήματος (χωρίς επαναφορά). Εάν το κείμενό σας ήταν μεγάλο και υπάρχει άλλο ένα τμήμα 512 πλακιδίων προς επεξεργασία, **το κύκλωμα δεν κάνει επαναφορά**. Τα οκτώ πλακίδια-οδηγοί παραμένουν ακριβώς όπως τα άφησε η πρώτη χιονοστιβάδα, και το δεύτερο τμήμα εκτοξεύεται εναντίον τους για να ενεργοποιήσει άλλους εξήντα τέσσερις γύρους. Είναι σαν να προσθέτετε ένα νέο δωμάτιο γεμάτο ντόμινο στο τέλος αυτού που μόλις έπεσε: η αταξία του πρώτου καθορίζει πλήρως το πώς θα πέσει το δεύτερο.

Βήμα 6. Λήψη της τελικής φωτογραφίας. Όταν δεν υπάρχουν πλέον τμήματα προς επεξεργασία, η χιονοστιβάδα σταματά. Κοιτάζουμε την τελική θέση στην οποία έχουν παραμείνει τα οκτώ πλακίδια-οδηγοί. Μεταφράζουμε τη διαμόρφωσή τους σε έναν κωδικό γραμμάτων και αριθμών στο δεκαεξαδικό σύστημα. Το αποτέλεσμα είναι μια σειρά ακριβώς εξήντα τεσσάρων χαρακτήρων: αυτή είναι η σφραγίδα SHA-256.

Τέσσερις ιδιότητες προκύπτουν από μόνες τους από τον τρόπο που είναι στημένο το κύκλωμα:

1. **Ντετερμινισμός.** Το ίδιο κείμενο παράγει πάντα την ίδια τελική φωτογραφία, σε οποιονδήποτε υπολογιστή στον κόσμο. Μηδενική τυχαιότητα, μηδενικές εκπλήξεις.
2. **Φαινόμενο χιονοστιβάδας.** Ένα κόμμα που προστέθηκε, ένα κεφαλαίο γράμμα που άλλαξε, ένας τόνος που ξεχάστηκε: η φωτογραφία καταλήγει εντελώς αγνώριστη. Αυτή είναι η ακραία ευαισθησία που ήδη περιγράψαμε στην αρχή.
3. **Μονόδρομη κατεύθυνση.** Δεδομένης της τελικής φωτογραφίας, δεν μπορείτε να ανακατασκευάσετε το αρχικό κείμενο. Οι περιστροφές, τα χωνιά και οι υπερχειλίσεις καταστρέφουν όλες τις κατευθυντήριες πληροφορίες σχετικά με το *από πού προήλθε κάθε bit* και διατηρούν μόνο το *τι προστέθηκε συνολικά*.
4. **Αντίσταση σε συγκρούσεις.** Σε είκοσι πέντε χρόνια δημόσιας κρυπτανάλυσης, κανείς δεν κατάφερε να βρει δύο διαφορετικά κείμενα των οποίων οι τελικές φωτογραφίες να συμπίπτουν. Και η δυσκολία να γίνει αυτό είναι πέρα από την υπολογιστική εμβέλεια οποιουδήποτε λογικά φανταστού πολιτισμού.

Το παράρτημα κώδικα που ακολουθεί υλοποιεί ακριβώς αυτά τα έξι βήματα στη Zig. Τώρα μπορείτε να το διαβάσετε γνωρίζοντας τι σημαίνει κάθε λειτουργία bit, αντί να αποδέχεστε τους χειρισμούς στα τυφλά.

Τεχνικό γλωσσάρι

Για τον αναγνώστη που θέλει να καταλάβει τι κάνει κάθε λειτουργία. Παρακάμψτε το ελεύθερα: το άρθρο γίνεται κατανοητό και χωρίς αυτό.

ASCII και Unicode — πώς τα γράμματα γίνονται αριθμοί. Οι υπολογιστές δεν βλέπουν γράμματα· βλέπουν αριθμούς. Ένα πρότυπο που ονομάζεται **ASCII** (*American Standard Code for Information Interchange*, του 1963) αντιστοιχίζει σε κάθε χαρακτήρα πληκτρολογίου έναν συγκεκριμένο αριθμό: το *A* είναι 65, το *B* είναι 66, το *a* είναι 97, το *0* είναι 48, το κενό είναι 32, το κόμμα είναι 44. Τα σύγχρονα συστήματα το επεκτείνουν με το **Unicode**, το οποίο εκχωρεί έναν αριθμό σε κάθε χαρακτήρα κάθε αλφαβήτου στον κόσμο: κυριλλικό, αραβικό, κινεζικό, ιαπωνικό, ακόμη και emoji. Όταν πληκτρολογείτε έναν χαρακτήρα ή ανοίγετε ένα αρχείο κειμένου, ο υπολογιστής διαβάζει τον υποκείμενο αριθμό, όχι το σχήμα στην οθόνη. Ο SHA-256 εργάζεται πάνω σε αυτούς τους αριθμούς, αντιμετωπίζοντας οποιοδήποτε κείμενο ως μια μακριά ακολουθία ψηφίων. Γι' αυτό μπορεί να σφραγίσει ένα άρθρο στα ισπανικά, ένα ποίημα στα ιαπωνικά και ένα δυαδικό αρχείο με τον ίδιο αλγόριθμο.
XOR — ο συγκριτής bit προς bit. Το XOR (προφέρεται «εξ-*ορ*», από το αγγλικό *exclusive or*, «αποκλειστικό ή») είναι μία από τις απλούστερες λειτουργίες που μπορεί να κάνει ένας υπολογιστής με δύο δυαδικούς αριθμούς. Συγκρίνει δύο bit θέση προς θέση και επιστρέφει: **1** εάν ακριβώς ένα από τα δύο είναι 1 (ένα αλλά όχι και τα δύο), **0** εάν και τα δύο είναι ίδια (και τα δύο 0 ή και τα δύο 1). Παράδειγμα: Το XOR των `1010` και `1100` είναι `0110`. Έχει μια αξιοσημείωτη ιδιότητα: είναι αναστρέψιμο —αν κάνετε XOR δύο φορές με το ίδιο κλειδί, επιστρέφετε στο αρχικό. Γι' αυτό είναι το «άλογο εργασίας» της κρυπτογραφίας: αναμιγνύει bit χωρίς να χάνει πληροφορίες, αλλά το αποτέλεσμα δεν αποκαλύπτει τίποτα για τις εισόδους αν δεν γνωρίζετε μία από αυτές.
Δεκαεξαδικό — μέτρηση στη βάση 16. Σχεδόν όλοι οι καθημερινοί αριθμοί χρησιμοποιούν δέκα ψηφία (0-9). Το δεκαεξαδικό χρησιμοποιεί δεκαέξι: τα συνηθισμένα 0-9 συν έξι γράμματα που αντιπροσωπεύουν τις ακόλουθες τιμές: `A` = 10, `B` = 11, `C` = 12, `D` = 13, `E` = 14, `F` = 15. Γιατί δεκαέξι; Επειδή οι υπολογιστές σκέφτονται σε ομάδες των τεσσάρων bit, και τα τέσσερα bit μπορούν να αναπαραστήσουν ακριβώς δεκαέξι διαφορετικές τιμές —έτσι, ένας δεκαεξαδικός χαρακτήρας αντιστοιχεί καθαρά σε τέσσερα bit. Ένα αποτύπωμα SHA-256 έχει μέγεθος 256 bit, τα οποία είναι ακριβώς **64 δεκαεξαδικοί χαρακτήρες**. Αν το γράφαμε στο συνηθισμένο δεκαδικό, θα καταλάμβανε περίπου 78 ψηφία και θα ήταν πιο άβολο. Η επιλογή είναι αισθητική και συμπαγής· ο υποκείμενος αριθμός είναι ο ίδιος.
Περιστροφή bit — το δυαδικό καρουζέλ. Φανταστείτε μια σειρά από επτά λαμπτήρες, μερικοί αναμμένοι (1) και άλλοι σβηστοί (0): `1 0 1 1 0 0 1`. Η περιστροφή προς τα δεξιά κατά μία θέση συνίσταται στο να πάρουμε τον λαμπτήρα στο τέρμα δεξιά, να τον μεταφέρουμε στο τέρμα αριστερά και να μετατοπίσουμε τους υπόλοιπους μία θέση προς τα δεξιά: `1 1`

`0 1 1 0 0`. Κανένας λαμπτήρας δεν χάνεται ούτε προστίθεται: απλώς χορεύουν κυκλικά. Ο SHA-256 χρησιμοποιεί την περιστροφή bit εκατοντάδες φορές σε κάθε υπολογισμό· είναι ένας φθηνός και χωρίς απώλειες τρόπος αναδιανομής των πληροφοριών εντός της κατάστασης.

Σταθερές *«χωρίς άσο στο μανίκι»* — γιατί προέρχονται από πρώτους αριθμούς. Τα οκτώ πλακίδια-οδηγοί και οι εξήντα τέσσερις σταθερές γύρου του SHA-256 δεν επιλέχθηκαν τυχαία. Προέρχονται από τις τετραγωνικές και κυβικές ρίζες των πρώτων πρώτων αριθμών. Γιατί; Επειδή οι σχεδιαστές τους ήθελαν σταθερές *«χωρίς τίποτα κρυμμένο»* (nothing-up-my-sleeve): τιμές των οποίων την προέλευση μπορεί ο καθένας να επαληθεύσει. Αν κάποιος σας έλεγε *«εμπιστευτείτε με: χρησιμοποιήστε αυτόν τον τυχαίο αριθμό 32 bit»*, θα υποψιαζόσασταν εύλογα μια κρυφή αδυναμία ή μια πίσω πόρτα. Αλλά οποιοσδήποτε με μια αριθμομηχανή μπορεί να ελέγξει ότι τα πρώτα 32 bit της τετραγωνικής ρίζας του 2 είναι `0x6a09e667`. Οι τιμές είναι μαθηματικές, δημόσιες και αναπαραγώγιμες: καμία κρυφή παγίδα δεν μπορεί να εισχωρήσει στη συνταγή.

Παράρτημα: Το SHA-256 σε αναγνώσιμο κώδικα

Αυτό το παράρτημα είναι για τον αναγνώστη που θέλει να δει τον αλγόριθμο από μέσα. Είναι μια διδακτική υλοποίηση στην Zig που ακολουθεί την προδιαγραφή FIPS 180-4. Δεν είναι η έκδοση που χρησιμοποιεί το Solo2 —η πραγματική βρίσκεται στο `std.crypto.hash.sha2.Sha256` της τυπικής βιβλιοθήκης της Zig, βελτιστοποιημένη και ελεγμένη—. Αλλά ο αλγόριθμος είναι ο ίδιος: αυτό που βλέπετε εδώ είναι, βήμα προς βήμα, αυτό που συμβαίνει όταν εκείνη η κλήση των πέντε χαρακτήρων εκτελεί τη δουλειά της.

```
const std = @import("std");

// SHA-256 — implementación didáctica.
// Sigue la especificación FIPS 180-4. Prioriza la claridad sobre la
// velocidad y la robustez frente a entradas hostiles. Para producción,
// usa std.crypto.hash.sha2.Sha256, que está optimizada y auditada.

// H0: las ocho palabras del estado inicial. Primeros 32 bits de la parte
// fraccionaria de las raíces cuadradas de los primeros ocho primos
// (2, 3, 5, 7, 11, 13, 17, 19).
const H0 = [_]u32{
    0x6a09e667, 0xbb67ae85, 0x3c6ef372, 0xa54ff53a,
    0x510e527f, 0x9b05688c, 0x1f83d9ab, 0x5be0cd19,
};

// K: 64 constantes de ronda. Primeros 32 bits de la parte fraccionaria
// de las raíces cúbicas de los primeros 64 primos.
const K = [_]u32{
    0x428a2f98, 0x71374491, 0xb5c0fbcf, 0xe9b5dba5, 0x3956c25b, 0x59f111f1, 0x923f82a4, 0xab1c5ed5,
    0xd807aa98, 0x12835b01, 0x243185be, 0x550c7dc3, 0x72be5d74, 0x80deb1fe, 0x9bdc06a7, 0xc19bf174,
    0xe49b69c1, 0xefbe4786, 0x0fc19dc6, 0x240ca1cc, 0x2de92c6f, 0x4a7484aa, 0x5cb0a9dc, 0x76f988da,
    0x983e5152, 0xa831c66d, 0xb00327c8, 0xbf597fc7, 0xc6e00bf3, 0xd5a79147, 0x06ca6351, 0x14292967,
    0x27b70a85, 0x2e1b2138, 0x4d2c6dfc, 0x53380d13, 0x650a7354, 0x766a0abb, 0x81c2c92e, 0x92722c85,
    0xa2bfe8a1, 0xa81a664b, 0xc24b8b70, 0xc76c51a3, 0xd192e819, 0xd6990624, 0xf40e3585, 0x106aa070,
    0x19a4c116, 0x1e376c08, 0x2748774c, 0x34b0bcb5, 0x391c0cb3, 0x4ed8aa4a, 0x5b9cca4f, 0x682e6ff3,
    0x748f82ee, 0x78a5636f, 0x84c87814, 0x8cc70208, 0x90befffa, 0xa4506ceb, 0xbef9a3f7, 0xc67178f2,
};

// Rotación circular a la derecha de un u32.
inline fn rotr(x: u32, n: u5) u32 {
    return std.math.rotr(u32, x, n);
}

// Lee 4 bytes consecutivos como un u32 big-endian.
inline fn readU32(b: []const u8) u32 {
    return @as(u32, b[0]) << 24 | @as(u32, b[1]) << 16 | @as(u32, b[2]) << 8 | @as(u32, b[3]);
}

// Escribe un u32 como 4 bytes consecutivos big-endian.
inline fn writeU32(b: []u8, v: u32) void {
    b[0] = @truncate(v >> 24);
    b[1] = @truncate(v >> 16);
    b[2] = @truncate(v >> 8);
    b[3] = @truncate(v);
}
```

```
// Compresión de un bloque de 64 bytes sobre el estado del hash. Sigue §6.2.2 de FIPS 180-4.
fn compress(state: *[8]u32, block: [16]u32) void {

    // 1. Expansión del schedule: 16 palabras → 64. Las nuevas se obtienen
    //    combinando cuatro anteriores con dos funciones de mezcla (s0 y s1)
    //    que usan rotación, XOR y desplazamiento. El "+%" es suma con
    //    truncado u32 (overflow-wrap), tal como exige el estándar.
    var w: [64]u32 = undefined;
    for (0..16) |i| w[i] = block[i];
    for (16..64) |i| {
        const s0 = rotr(w[i-15], 7) ^ rotr(w[i-15], 18) ^ (w[i-15] >> 3);
        const s1 = rotr(w[i-2], 17) ^ rotr(w[i-2], 19) ^ (w[i-2] >> 10);
        w[i] = w[i-16] +% s0 +% w[i-7] +% s1;
    }

    // 2. Variables de trabajo: copia del estado actual.
    var a = state[0]; var b = state[1]; var c = state[2]; var d = state[3];
    var e = state[4]; var f = state[5]; var g = state[6]; var h = state[7];

    // 3. 64 rondas de mezcla no lineal.
    //    S1, S0  : combinaciones rotacionales de 'e' y 'a'.
    //    ch      : "choose" — multiplexor bit a bit, elige entre f y g según e.
    //    maj     : "majority" — bit mayoritario entre a, b, c.
    //    t1 + t2 : se inyecta al top de la cascada cada ronda.
    for (0..64) |i| {
        const S1 = rotr(e, 6) ^ rotr(e, 11) ^ rotr(e, 25);
        const ch = (e & f) ^ (~e & g);
        const t1 = h +% S1 +% ch +% K[i] +% w[i];
        const S0 = rotr(a, 2) ^ rotr(a, 13) ^ rotr(a, 22);
        const maj = (a & b) ^ (a & c) ^ (b & c);
        const t2 = S0 +% maj;
        h = g; g = f; f = e; e = d +% t1;
        d = c; c = b; b = a; a = t1 +% t2;
    }

    // 4. Acumular las variables de trabajo en el estado.
    state[0] +%= a; state[1] +%= b; state[2] +%= c; state[3] +%= d;
    state[4] +%= e; state[5] +%= f; state[6] +%= g; state[7] +%= h;
}

// Hash completo: procesa el mensaje en bloques, padea el último, escribe el resumen.
pub fn sha256(msg: []const u8, out: *[32]u8) void {
    var state = H0;
    var block: [64]u8 = undefined;
    var block_w: [16]u32 = undefined;

    // Procesar bloques completos del mensaje original.
    var i: usize = 0;
    while (i + 64 <= msg.len) : (i += 64) {
        @memcpy(block[0..64], msg[i..i+64]);
        for (0..16) |j| block_w[j] = readU32(block[j*4..j*4+4]);
        compress(&state, block_w);
    }

    // Padding del último bloque: byte 0x80, después ceros, después la
    // longitud original (en bits) como u64 big-endian en los 8 últimos bytes.
    const remaining = msg.len - i;
    @memcpy(block[0..remaining], msg[i..]);
    block[remaining] = 0x80;
    const bit_len: u64 = @as(u64, msg.len) * 8;

    if (remaining + 1 + 8 <= 64) {
        // El padding cabe en el mismo bloque.
        for (remaining + 1..56) |k| block[k] = 0;
        var k: usize = 0;
        while (k < 8) : (k += 1) block[56 + k] = @truncate(bit_len >> @as(u6, @intCast((7 - k) * 8)));
        for (0..16) |j| block_w[j] = readU32(block[j*4..j*4+4]);
        compress(&state, block_w);
    } else {
        // El padding requiere un bloque adicional.
        for (remaining + 1..64) |k| block[k] = 0;
```

```
        for (0..16) |j| block_w[j] = readU32(block[j*4..j*4+4]);
        compress(&state, block_w);
        for (0..56) |k| block[k] = 0;
        var k: usize = 0;
        while (k < 8) : (k += 1) block[56 + k] = @truncate(bit_len >> @as(u6, @intCast((7 - k) * 8)));
        for (0..16) |j| block_w[j] = readU32(block[j*4..j*4+4]);
        compress(&state, block_w);
    }

    // Escribir el estado final como 32 bytes big-endian.
    for (0..8) |j| writeU32(out[j*4..j*4+4], state[j]);
}

// Ejemplo de uso.
pub fn main() void {
    var resumen: [32]u8 = undefined;
    sha256("Cuadernos Lacre", &resumen);
    for (resumen) |byte| std.debug.print("{x:0>2}", .{byte});
    std.debug.print("\n", .{});
    // Imprime: ae6bdea6bbf5476889e0651a31f3dc1612fc61497477e21a95cabae2a6886c3e
}
```

Οποιαδήποτε επανεγγραφή σε άλλη γλώσσα που ακολουθεί την ίδια δομή —αρχικές σταθερές, επέκταση του προγράμματος (schedule expansion), εξήντα τέσσερις γύροι, συσσώρευση— παράγει το ίδιο αποτέλεσμα. Ο αλγόριθμος δεν έχει μυστικά: η αξία του έγκειται στο ότι οι ιδιότητες που αναφέρονται παραπάνω συνεχίζουν να ισχύουν μετά από δύο δεκαετίες δημόσιας κρυπτανάλυσης από χιλιάδες μάτια.

Εάν επιστρέψετε στο κάτω μέρος αυτού του άρθρου, θα δείτε μια δεκαεξαδική σφραγίδα εξήντα τεσσάρων χαρακτήρων. Είναι το SHA-256 του κειμένου που μόλις διαβάσατε, σε αυτή τη γλώσσα. Εάν μεταφράζαμε το άρθρο, η σφραγίδα θα ήταν άλλη· εάν άλλαζε μια λέξη της ελληνικής έκδοσης, η ελληνική σφραγίδα θα άλλαζε. Η σφραγίδα δεν προστατεύει το περιεχόμενο —γι' αυτό υπάρχουν άλλα εργαλεία— αλλά το προσδιορίζει μοναδικά. Και αυτό, όσο ταπεινό κι αν ακούγεται, αρκεί ώστε κανένα βήμα της εκδοτικής αλυσίδας να μην μπορεί να αλλοιώσει τα λεγόμενα χωρίς να γίνει αντιληπτό. Τα υπόλοιπα —κρυπτογράφηση, υπογραφή, ταυτοποίηση— χτίζονται πάνω σε αυτή την απλή ιδέα.

Πηγές και περαιτέρω μελέτη

- NIST — *FIPS PUB 180-4: Secure Hash Standard (SHS)*, Αύγουστος 2015. Επίσημη προδιαγραφή της οικογένειας SHA-2, συμπεριλαμβανομένου του SHA-256.
- RFC 6234 — *US Secure Hash Algorithms (SHA and SHA-based HMAC and HKDF)*, IETF, Μάιος 2011. Κανονιστική έκδοση για υλοποιητές.
- Ferguson, N.; Schneier, B.; Kohno, T. — *Cryptography Engineering: Design Principles and Practical Applications* (Wiley, 2010). Τα κεφάλαια 5 και 6 καλύπτουν τις συναρτήσεις hash και τις νόμιμες και παράνομες χρήσεις τους.
- Nakamoto, S. — Bitcoin: A Peer-to-Peer Electronic Cash System (2008). Πρακτικό παράδειγμα χρήσης του SHA-256 για τη σύνδεση μπλοκ σε μια αμετάβλητη δομή.
- Κανονισμός (ΕΕ) 910/2014 (eIDAS) — πλαίσιο των εγκεκριμένων παρόχων υπηρεσιών χρονοσήμανσης. Το SHA-256 είναι η συνάρτηση αναφοράς για τις εγκεκριμένες ηλεκτρονικές υπογραφές και σφραγίδες που εκδίδονται στην ΕΕ.
- Υλοποίηση αναφοράς στην Zig: `std.crypto.hash.sha2.Sha256` στο επίσημο αποθετήριο της γλώσσας (*github.com/ziglang/zig* → `lib/std/crypto/sha2.zig`). Είναι η βελτιστοποιημένη και ελεγμένη έκδοση που χρησιμοποιεί στην πραγματικότητα το Solo2. Χρήσιμο για σύγκριση με τη διδακτική υλοποίηση του παραρτήματος.

← ΠροηγούμενοCUADERNOS_LIST_SCHREMS_TITLEΕπόμενο →CUADERNOS_LIST_KILLSWITCH_TITLE

Πρόσφατα αναγνώσματα

- CUADERNOS_LIST_PREGUNTAS_TITLE
- CUADERNOS_LIST_SELFHOST_TITLE
- CUADERNOS_LIST_IDENTIDAD_TITLE

Πάρτε αυτό το άρθρο μαζί σας όπου το χρειάζεστε.

↓ Markdown ↓ Απλό κείμενο ↓ PDF

Το αρχείο θα ληφθεί στη συσκευή σας. Από εκεί μπορείτε να το αποθηκεύσετε, να το εισαγάγετε στο Solo2 ή να το μοιραστείτε όπου θέλετε. Το Cuadernos δεν αποφασίζει τον προορισμό για εσάς.

Σφραγίδα από βουλοκέρι · SHA-256 7724e482068bfeeaa92175b56e07bd6163f84f7a3bf8b9df7e7747bdf48dbf10

Cuadernos Lacre · Μια έκδοση της Menzuri Gestión S.L. ·
γραμμένη από τον R.Eugenio · επιμελημένη από την ομάδα του Solo2.

Αυτός ο ιστότοπος δεν χρησιμοποιεί cookies και δεν φορτώνει πόρους από τρίτους. Χρησιμοποιεί έναν ανώνυμο μετρητή επισκέψεων με δική μας φιλοξενία (Umami, στον ευρωπαϊκό μας διακομιστή) και το ελάχιστο JavaScript που απαιτείται για την προτίμησή σας σε φωτεινό/σκοτεινό θέμα. Χωρίς ιχνηλάτες, χωρίς προφίλ, χωρίς κοινή χρήση δεδομένων. Αν θέλετε να μας ακολουθήσετε: RSS.