

ক্রিপ্টোগ্রাফি (Cryptography) কী?

ক্রিপ্টোগ্রাফি হলো তথ্য বা বার্তাকে এমনভাবে পরিবর্তন করার বিজ্ঞান ও প্রযুক্তি, যাতে শুধুমাত্র নির্দিষ্ট ব্যক্তি বা পক্ষই সেই তথ্য বুঝতে পারে।

সরাসরি বললে — **তথ্য গোপন করার কৌশল**।

প্রাচীনকালেও রাজারা বার্তা লুকিয়ে পাঠানোর জন্য ক্রিপ্টোগ্রাফির ব্যবহার করতেন, আর এখন ইন্টারনেট, ব্যাংকিং, মেসেজিং, সবকিছুতেই এর ব্যবহার হয়।

এনক্রিপশন (Encryption) কী?

এনক্রিপশন হলো, কোনো স্পষ্ট তথ্য (plaintext) কে এমনভাবে কোডিং করা, যাতে সেটা অপরিচিত বা অননুমোদিত ব্যক্তির কাছে অর্থহীন হয়ে যায়।

মানে, তথ্যটাকে এমনভাবে **গোপন** করা হয় যাতে কেউ চাইলেও সহজে পড়তে না পারে।

উদাহরণ:

আমরা যদি "HELLO" শব্দটাকে এনক্রিপ্ট করি, সেটা হতে পারে "XK47@#L" — মানে আর বোঝার উপায় নেই।

ডিক্রিপশন (Decryption) কী?

ডিক্রিপশন হলো এনক্রিপ্ট করা তথ্যটাকে আবার আগের মূল অবস্থায় (plaintext এ) ফিরিয়ে আনার প্রক্রিয়া।

ডিক্রিপশন করতে সাধারণত **একটি চাবি (key)** লাগে, যেটা ছাড়া কেউ তথ্যটি উদ্ধার করতে পারে না।

ক্রিপ্টোগ্রাফি কীভাবে কাজ করে?

ক্রিপ্টোগ্রাফিতে সাধারণত দুইটা ধাপ থাকে:

1. **এনক্রিপশন:**
ব্যবহারকারী বার্তাকে **এনক্রিপশন অ্যালগরিদম** এবং **কী (key)** ব্যবহার করে কোড করে ফেলে।
2. **ডিক্রিপশন:**
যাকে বার্তাটি পাঠানো হয়েছে, সে আবার সেই কী (বা কখনো ভিন্ন কী) ব্যবহার করে বার্তাটিকে স্বাভাবিক অবস্থায় নিয়ে আসে।

এখানে সবচেয়ে গুরুত্বপূর্ণ বিষয় হলো —

Key ছাড়া এনক্রিপ্টেড মেসেজ পড়া **প্রায় অসম্ভব** (অন্তত আধুনিক ক্রিপ্টোগ্রাফিতে)।

ক্রিপ্টোগ্রাফির প্রকারভেদ (Prokar):

১. সিমেন্ট্রিক ক্রিপ্টোগ্রাফি (Symmetric Cryptography):

- একই কী দিয়ে এনক্রিপশন এবং ডিক্রিপশন হয়।
- দ্রুত, সহজ — তবে চাবি আদান-প্রদান করতে ঝুঁকি থাকে।
- উদাহরণ: AES (Advanced Encryption Standard)

২. অ্যাসিমেন্ট্রিক ক্রিপ্টোগ্রাফি (Asymmetric Cryptography):

- এখানে দুটি কী থাকে — Public Key এবং Private Key।
- Public Key দিয়ে এনক্রিপশন হয়, আর Private Key দিয়ে ডিক্রিপশন।
- নিরাপত্তা বেশি।
- উদাহরণ: RSA, ECC (Elliptic Curve Cryptography)

এখন কোনটা বেশি ব্যবহার হয়?

☒ আজকের দিনে Symmetric + Asymmetric দুটোর **কম্বিনেশন** ব্যবহৃত হয় নিরাপত্তা নিশ্চিত করার জন্য।

- ডেটা ট্রান্সমিশন** এর সময় Asymmetric Cryptography দিয়ে Secure Key আদান-প্রদান হয়।
- তারপর সেই Key দিয়ে Symmetric Encryption হয় কারণ সেটা অনেক ফাস্ট।

উদাহরণ:

- ☒ SSL/TLS (যেটা দিয়ে আমাদের ব্রাউজার ও ওয়েবসাইটের মধ্যে সিকিউর কানেকশন হয়)
- ☒ Bank Transaction Security
- ☒ Messaging Apps (WhatsApp, Signal ইত্যাদিতে End-to-End Encryption)

সংক্ষেপে:

বিষয়	ব্যাখ্যা
ক্রিপ্টোগ্রাফি	তথ্য গোপন করার বিজ্ঞান

এনক্রিপশন	তথ্যকে অজানা কোডে রূপান্তর করা
ডিক্রিপশন	কোডকে আগের অবস্থায় ফিরিয়ে আনা
প্রধান ধরন	Symmetric, Asymmetric
বর্তমান প্রচলিত ব্যবহার	SSL/TLS, Banking, Secure Communication

RSA কী?

RSA হলো এক ধরনের **Asymmetric Cryptography Algorithm**।

এখানে দুটি কী ব্যবহার হয় —

- একটি **Public Key** (সবার জন্য উন্মুক্ত)
- একটি **Private Key** (গোপন রাখা হয়)

কাজের ধরণ:

- Public Key দিয়ে মেসেজ **encrypt** করা হয়।
- Private Key দিয়ে সেই মেসেজ **decrypt** করা হয়।

RSA নাম এসেছে তিনজন বিজ্ঞানীর নাম থেকে —

Rivest, Shamir, Adleman।

RSA ব্যবহারের প্রধান উদ্দেশ্য:

- ☒ ডেটা নিরাপদ রাখা
- ☒ ডিজিটাল স্বাক্ষর (Digital Signature)
- ☒ Key Exchange (দুই পক্ষের মধ্যে সিকিউর কী আদান-প্রদান)

SHA কী?

SHA মানে হলো **Secure Hash Algorithm**।

এটা আসলে এনক্রিপশন না — এটা হলো **হ্যাশিং** প্রসেস।

হ্যাশিং কী?

হ্যাশিং মানে, যেকোনো দৈর্ঘ্যের তথ্যকে **একটি নির্দিষ্ট দৈর্ঘ্যের ফিক্সড সাইজের কোডে** রূপান্তর করা।
এবং এই হ্যাশ থেকে মূল তথ্য উদ্ধার করা সম্ভব না (One-Way Function)।

SHA মূলত ব্যবহার হয়:

- পাসওয়ার্ড স্টোর করার জন্য

- ফাইল ইন্টিগ্রিটি চেক করার জন্য
- ডিজিটাল সিগনেচার তৈরির জন্য

SHA কত প্রকার?

SHA এর অনেকগুলো ভার্সন আছে। এগুলো হলো:

Algorithm Name	Hash Size (Bit)	বর্তমানে ব্যবহার
SHA-1	160-bit	না, দুর্বল হয়েছে
SHA-2 (SHA-224, SHA-256, SHA-384, SHA-512)	224, 256, 384, 512-bit	<input checked="" type="checkbox"/> ব্যবহৃত হচ্ছে
SHA-3 (SHA3-224, SHA3-256, SHA3-384, SHA3-512)	224, 256, 384, 512-bit	<input checked="" type="checkbox"/> নতুন ও শক্তিশালী

সংক্ষেপে:

- **SHA-1:** পুরাতন, এখন আর নিরাপদ নয়।
- **SHA-2:** বর্তমানে সবচেয়ে বেশি ব্যবহৃত। (বিশেষ করে SHA-256)
- **SHA-3:** আরও নতুন জেনারেশন, খুব শক্তিশালী, ভবিষ্যতে SHA-2 কে রিপ্লেস করতে পারে।

SHA কত Bit এর হয়?

SHA-2 এবং SHA-3 দুইটাই বিভিন্ন বিট সাইজে আসে:

টাইপ	বিট সাইজ
SHA-224 / SHA3-224	224-bit
SHA-256 / SHA3-256	256-bit
SHA-384 / SHA3-384	384-bit
SHA-512 / SHA3-512	512-bit

বর্তমানে সবচেয়ে বেশি ব্যবহৃত —

- ☒ **SHA-256** (256-bit security) — যেমনঃ Bitcoin, SSL Certificates ইত্যাদিতে।
 - ☒ **SHA-512** — আরো বেশি শক্তিশালী নিরাপত্তার জন্য।
-

● RSA এবং SHA কীভাবে আলাদা?

দিক	RSA	SHA
উদ্দেশ্য	এনক্রিপশন ও ডিক্রিপশন	হ্যাশিং ও ইন্টিগ্রিটি চেক
কী থাকে?	Public Key & Private Key	কোন কী লাগে না
রূপান্তর	দুই দিকে (Encrypt-Decryption)	এক দিকে (Hash only)
ব্যবহার	ডেটা এনক্রিপশন, Key Exchange, Digital Signature	Password Hash, Data Verification, Digital Signature Hashing

☑ সংক্ষিপ্ত সারাংশ:

- **RSA** = ডেটা লুকিয়ে রাখা + চাবির সাহায্যে উদ্ধার করা।
- **SHA** = ডেটাকে একমুখী চিরস্থায়ী কোডে রূপান্তর করা।

🔗 একদম বাস্তব উদাহরণ:

- যখন তুমি কোন ওয়েবসাইটে লগইন করো, তোমার পাসওয়ার্ডটি সরাসরি সংরক্ষণ করা হয় না। সেটা প্রথমে **SHA-256** বা **SHA-512** দিয়ে হ্যাশ করা হয়, তারপর সেই হ্যাশ সংরক্ষিত থাকে।
- যখন তুমি কোনো মেসেজ এনক্রিপ্ট করে পাঠাও (যেমন Bank Transaction), তখন **RSA** দিয়ে Key Exchange হয় আর এরপর ডেটা এনক্রিপ্ট করা হয়।

● SSL/TLS কী?

SSL (Secure Sockets Layer) এবং **TLS** (Transport Layer Security) হলো ইন্টারনেটের মাধ্যমে তথ্য নিরাপদে আদান-প্রদানের প্রযুক্তি।

মানে, তুমি যখন কোনো ওয়েবসাইট ভিজিট করো (যেমন ব্যাংক, ই-মেইল, ই-কমার্স), তখন SSL/TLS তোমার ব্রাউজার আর সার্ভারের মধ্যে ডেটা সিকিউর রাখে যাতে মাঝপথে কেউ চুরি করতে না পারে।

📌 **SSL পুরনো ভার্সন**

📌 **TLS নতুন এবং উন্নত ভার্সন**

আজকাল আমরা যা দেখি HTTPS ওয়েবসাইটে — আসলে সেটা **TLS** ব্যবহার করে।

● SSL/TLS কীভাবে কাজ করে? (সহজ ভাষায় Step-by-Step)

১. Client Hello:

- ব্রাউজার (client) সার্ভারকে বলে: "আমি TLS কানেকশন শুরু করতে চাই। এই হলো আমার সাপোর্ট করা Encryption Algorithms (Cipher Suites)।"

২. Server Hello:

- সার্ভার তার সার্টিফিকেট (SSL/TLS Certificate) পাঠায়। এর মধ্যে তার Public Key থাকে।
- সার্ভার বলে: "ঠিক আছে, এই অ্যালগরিদম ব্যবহার করবো।"

3. Key Exchange:

- ক্লায়েন্ট একটা "Pre-Master Secret" তৈরি করে, সেটা সার্ভারের Public Key দিয়ে **Encrypt** করে সার্ভারে পাঠায়।
- সার্ভার তার Private Key দিয়ে সেটা **Decrypt** করে।

4. Session Key তৈরি:

- দুজন একই Secret থেকে একটা "Session Key" তৈরি করে।
- এরপর থেকে দুই পক্ষ নিজেদের মধ্যে **Symmetric Encryption** (যেমন AES) দিয়ে ডেটা ট্রান্সফার করে।

5. Secure Communication শুরু:

- এখন ব্রাউজার আর সার্ভার পরস্পরের সাথে গোপন এনক্রিপ্টেডভাবে কথা বলে।

🔗 SSL/TLS-এ কোন ধরনের এনক্রিপশন ব্যবহৃত হয়?

SSL/TLS এর মধ্যে দুই ধরনের এনক্রিপশন ব্যবহার হয়:

১. Asymmetric Encryption (Public Key, Private Key দিয়ে)

- শুরুতেই যখন কী আদান-প্রদান হয় (Key Exchange) তখন এটা ব্যবহৃত হয়।
- উদাহরণ: **RSA**, **ECC** (Elliptic Curve Cryptography)

২. Symmetric Encryption (Session Key দিয়ে)

- পুরো ডেটা ট্রান্সফার করার সময় এটা ব্যবহৃত হয়।
- উদাহরণ: **AES** (Advanced Encryption Standard), **ChaCha20**

🔔 কেন দুই ধরনের Encryption?

কারণ Asymmetric Encryption (RSA) ধীরগতি সম্পন্ন। তাই শুরুতেই সিকিউর কী শেয়ার করে, এরপর Symmetric Encryption (AES) দিয়ে দ্রুত ডেটা পাঠানো হয়।

🔗 Encryption নিয়ে আরো details:

১. Symmetric Encryption:

- একই চাবি দিয়ে এনক্রিপ্ট এবং ডিক্রিপ্ট হয়।
- খুব দ্রুত এবং কম রিসোর্সে কাজ করে।
- উদাহরণ: AES-128, AES-256

২. Asymmetric Encryption:

- Public Key দিয়ে এনক্রিপ্ট, Private Key দিয়ে ডিক্রিপ্ট।
- নিরাপত্তা বেশি, কিন্তু প্রসেসিং ধীর।
- উদাহরণ: RSA-2048, RSA-4096, ECC (P-256, P-384)

৩. Hashing (Integrity Checking):

- SSL/TLS হ্যাশ ফাংশনও ব্যবহার করে মেসেজের ইন্টিগ্রিটি ঠিক আছে কিনা যাচাই করতে।
- উদাহরণ: SHA-256

বর্তমানে SSL/TLS এর অবস্থা:

SSL/TLS ভার্সন


স্ট্যাটাস

SSL 2.0 / SSL 3.0 ☒ সম্পূর্ণ বন্ধ (দুর্বল)

TLS 1.0 / 1.1 ☒ এখন বন্ধ করার পরামর্শ দেওয়া হয়

TLS 1.2 ☒ স্ট্যান্ডার্ড, সবচেয়ে বেশি ব্যবহৃত

TLS 1.3 ☒ আরো উন্নত, দ্রুত এবং নিরাপদ (2024 সালে অনেক জায়গায় TLS 1.3 ব্যবহার হচ্ছে)

 এখনকার ওয়েবসাইট বা ব্যাংকিং সিস্টেমে TLS 1.2 বা TLS 1.3 ব্যবহার হয়। SSL পুরনো হয়ে গেছে।

☒ সংক্ষেপে:

- **SSL/TLS:** ইন্টারনেটে সিকিউর কমিউনিকেশনের জন্য ব্যবহৃত হয়।
- প্রথমে **Asymmetric Encryption** দিয়ে Secure Key Exchange হয়।
- পরে **Symmetric Encryption** দিয়ে দ্রুত এবং নিরাপদ ডেটা ট্রান্সফার হয়।
- SSL এখন পুরাতন, **TLS 1.2/1.3** চালু।
- **SHA-256** বা আরও ভালো Hash Algorithm দিয়ে Integrity Protect করা হয়।

একদম বাস্তব জীবন উদাহরণ:

- তোমার যখন মোবাইলে ব্যাংকের অ্যাপ খুলো, তখন TLS এর মাধ্যমে সিকিউর কানেকশন হয়।

- Google, Facebook, Gmail — সবাই HTTPS (মানে TLS) ব্যবহার করে।
- Online shopping বা Payment Gateway SSL/TLS ছাড়া কাজই করবে না।

🔗 Hash Algorithm কী?

Hash Algorithm বা হ্যাশ অ্যালগরিদম হলো এমন একটি গণিতভিত্তিক প্রসেস, যা যেকোনো মাপের (length) ইনপুট ডেটাকে → একটি নির্দিষ্ট মাপের (fixed-length) ছোট কোডে রূপান্তর করে।

এই ছোট কোডকেই বলে Hash বা Hash Value।

👉 সহজ ভাষায়:

"বড়সড় ডেটাকে ছোট একটি 'ডিজিটাল ফিঙ্গারপ্রিন্ট' এ রূপান্তর করা হয় — যেটা থেকে আসল ডেটা উদ্ধার করা যায় না।"

🔗 Hash Algorithm-এর বৈশিষ্ট্য (Properties)

একটি ভালো Hash Algorithm এর মধ্যে কিছু গুরুত্বপূর্ণ বৈশিষ্ট্য থাকতে হয়:

বৈশিষ্ট্য	ব্যাখ্যা
1. Deterministic	একই ইনপুট সবসময় একই হ্যাশ দিবে
2. Fast Computation	খুব দ্রুত হ্যাশ তৈরি করতে পারবে
3. Pre-image Resistance	হ্যাশ দেখে আসল ডেটা খুঁজে বের করা অসম্ভব
4. Small Change = Big Difference	ইনপুটে সামান্য পরিবর্তনেও সম্পূর্ণ ভিন্ন হ্যাশ তৈরি হবে
5. Collision Resistant	দুইটা ভিন্ন ইনপুট যেন একই হ্যাশ তৈরি না করে

🔗 Hash Algorithm-এর ব্যবহার (Uses)

Hash Algorithm রোজকার সিকিউরিটি ও প্রযুক্তিতে অনেক কাজে লাগে, যেমন:

- ☒ Password Protection (পাসওয়ার্ড সংরক্ষণ)
 - ☒ Data Integrity Check (ফাইল পরিবর্তন হয়েছে কিনা চেক)
 - ☒ Digital Signature Verification
 - ☒ Blockchain (Bitcoin, Ethereum ইত্যাদি)
 - ☒ SSL/TLS Handshake-এ Message Authentication
 - ☒ Malware detection (File Hash Match করে)
-

● জনপ্রিয় Hash Algorithms

Algorithm	Hash Size	ব্যবহার	অবস্থা
MD5	128-bit	পুরনো password hash, file verification	✗ দুর্বল, ব্যবহার নিষিদ্ধ
SHA-1	160-bit	পুরনো SSL cert, Git	✗ দুর্বল, ব্যবহার না করার পরামর্শ
SHA-2 (SHA-224, SHA-256, SHA-384, SHA-512)	224, 256, 384, 512-bit	Modern password hashing, SSL/TLS	☑ সবচেয়ে ব্যবহৃত
SHA-3	224, 256, 384, 512-bit	Future-proof cryptography	☑ শক্তিশালী, নতুন
BLAKE2	256-bit বা 512-bit	ফাস্ট hashing, modern apps	☑ বিকল্প হিসেবে জনপ্রিয়

● SHA-2 Family উদাহরণ:

- **SHA-224** → 224-bit hash
- **SHA-256** → 256-bit hash ☑ (সবচেয়ে জনপ্রিয়)
- **SHA-384** → 384-bit hash
- **SHA-512** → 512-bit hash (অত্যন্ত শক্তিশালী)

● Hash কেমন দেখতে হয়?

উদাহরণ:

আমরা যদি "Hello World" টেক্সটটি SHA-256 দিয়ে হ্যাশ করি, তাহলে বের হবে:

```
arduino
CopyEdit
SHA-256("Hello World") =
a591a6d40bf420404a011733cfb7b190d62c65bf0bcda32b57b277d9ad9f146e
```

একটা বড় ইনপুট থেকেও সবসময় নির্দিষ্ট সাইজের একটা কোড তৈরি হবে!

● Hash Function বাস্তব উদাহরণ:

বাস্তব ঘটনা

ফাইল
ডাউনলোড

ওয়েবসাইট
লগইন

Hash Function কিভাবে কাজ করে

ফাইলের সাথে দেওয়া Hash মিলিয়ে দেখা হয় ডাউনলোডের সময় কোনো পরিবর্তন হয়েছে কিনা

পাসওয়ার্ড সরাসরি সংরক্ষণ না করে, হ্যাশ করে ডাটাবেজে রাখা হয়

বাস্তব ঘটনা

Hash Function কিভাবে কাজ করে

Blockchain

প্রতিটি ব্লকের মধ্যে আগের ব্লকের হ্যাশ রাখা হয়, তাই ব্লক পরিবর্তন করলে পুরো চেইন নষ্ট হয়ে যায়

🔑 সংক্ষেপে মনে রাখার টিপস:

- Hash = Digital Fingerprint
- Hashing = One-way Process (ফিরে যাওয়া সম্ভব না)
- SHA-256 এখন সবচেয়ে বেশি ব্যবহৃত
- MD5, SHA-1 এখন দুর্বল ও ঝুঁকিপূর্ণ

Hash করা ডেটা কীভাবে ডিক্রিপ্ট করে? নাকি শুধু স্টোর করা হয়?"

চলো একদম ক্লিয়ার করে বলি বাংলা ভাষায়:

🔍 উত্তর:

👉 Hash করা ডেটা কখনোই ডিক্রিপ্ট করা যায় না।

👉 Hash একদিকের (One-Way) প্রসেস।

👉 মানে, তুমি ইনপুট দিয়ে Hash তৈরি করতে পারবে — কিন্তু সেই Hash থেকে আসল ইনপুট উদ্ধার করা সম্ভব না।

📌 Hash করা ডেটা কেবল স্টোর বা ভেরিফাই করার জন্য ব্যবহৃত হয়।

🔍 বাস্তব উদাহরণ দিয়ে বুঝি:

ধরো, তোমার পাসওয়ার্ড হলো `MySecret123`

যখন তুমি রেজিস্ট্রেশন করো, তখন সার্ভার তোমার পাসওয়ার্ডটা Hash করে ডাটাবেজে রাখে — ধরো, SHA-256 দিয়ে:

```
arduino
CopyEdit
SHA-256("MySecret123") = abcd1234efgh5678....
```

ডাটাবেজে এই হ্যাশটা স্টোর করা হয়, আসল পাসওয়ার্ড না।

● এখন তুমি লগইন করতে গেলে কী হয়?

১. তুমি আবার পাসওয়ার্ড লেখো (MySecret123)
২. সার্ভার সেই পাসওয়ার্ডটাকে আবার Hash করে। ৩. নতুন হ্যাশ আর ডাটাবেজের পুরনো হ্যাশ মিলিয়ে দেখে। ৪. যদি মিলে যায় → লগইন সফল! 🎉

● কিন্তু কখনোই ডাটাবেজে রাখা Hash থেকে আসল পাসওয়ার্ড উদ্ধার করা হয় না বা যায় না!

🔍 তাহলে যদি হ্যাশের আসল ডেটা দরকার হয়?

হ্যাশ কখনো আসল ডেটা ফেরত দেয় না।

তবে কেউ brute force বা dictionary attack দিয়ে অনুমান করার চেষ্টা করতে পারে (যেমন বারবার পাসওয়ার্ড ট্রাই করে দেখা) — তবে সেটা সহজ নয়, বিশেষ করে যদি:

- পাসওয়ার্ড কঠিন হয়
- সল্টিং (Salting) করা থাকে
- ভালো Hash Algorithm (SHA-256/SHA-3) ব্যবহার হয়

🔍 সংক্ষেপে মনে রাখো:

বিষয়	Hash	Encryption
কাজ	একদিকের কোড তৈরি	দুইদিকের (Encrypt ↔ Decrypt)
ডেটা ফেরত আসে?	❌ না	✅ হ্যাঁ (key দিয়ে)
প্রধান উদ্দেশ্য	Integrity Check, Password Storage	ডেটা গোপন রাখা

✅ এক লাইনে উত্তর:

"Hash = একদিকের রাস্তা (One-way street) — স্টোর করে, ডিক্রিপ্ট হয় না।"

"Salting", "Pepper", "Rainbow Table Attack"!

চলো সব একদম সহজ ভাষায়, Example দিয়ে বুঝি বাংলায়:

🔍 ১. Salting কী?

Salting মানে হলো —

পাসওয়ার্ডের সাথে **একটু Extra Random Data** যোগ করে তারপর Hash করা।

👉 উদ্দেশ্য হলো, যাতে দুইজন ইউজার যদি একি পাসওয়ার্ড ব্যবহার করে, তাহলেও তাদের হ্যাশ একরকম না হয়!

◇ Example:

ধরো, দুইজন ইউজার:

User	Password
A	MySecret123
B	MySecret123

যদি সোজাসুজি SHA-256 দিয়ে hash করি, তাহলে দুইজনের hashed password হবে একই! 😬

এটা বিপদজনক!

☒ তাই Salting করা হয়।

ধরো:

User	Password	Salt	Final Data	Hash Result
A	MySecret123	abc123	MySecret123abc123	Hashed_A
B	MySecret123	xyz789	MySecret123xyz789	Hashed_B

এখন দুইজনের hash আলাদা হবে! 🔥

◇ Key Point:

Salting = Password + Random Salt → তারপর Hash করা।

এটা Auto manage করে bcrypt, Argon2, scrypt এর মতো মডার্ন লাইব্রেরি।

🌀 ২. Pepper কী?

Pepper হলো —

একটা **Secret Value** যেটা Application Level এ রাখা হয় (Database এ রাখা হয় না)।

👉 এটা Password এর সাথে মিশানো হয় Hash করার আগে।

◇ Example:

- ধরো, Site Admin আগে থেকেই একটা গোপন **Pepper Key** ঠিক করে রেখেছে:
"SECRET_KEY_987".
- এখন User password হলো: "MySecret123"

তাহলে:

```
pgsql
CopyEdit
Final Password to Hash = Password + Pepper
                        = "MySecret123SECRET_KEY_987"
```

তারপর এইটা Hash করে ডাটাবেজে রাখে।

ফায়দা:

- যদি কেউ ডাটাবেজ চুরি করে, তাহলেও তাকে আলাদা করে সেই গোপন Pepper Key বের করতে হবে! 🔥
 - অনেক বেশি Security বাড়ায়।
-

🌀 ৩. Rainbow Table Attack কী?

Rainbow Table Attack হলো —

হ্যাকাররা আগে থেকেই **লক্ষ লক্ষ Common Password এর Hash** বানিয়ে রাখে একটা বড় টেবিলে। তারপর যদি তারা কোন hashed password পায়, তখন তারা ওই table থেকে মিলিয়ে Matching Hash খুঁজে পায়! 🤖

◇ Example:

Password SHA-256 Hash

123456 abcdef...

password ghijkl...

admin mnopqr...

● এখন যদি কোন ডাটাবেজ থেকে শুধু Hash চুরি হয়, হ্যাকার চেক করবে এই রেডিমেড টেবিল দিয়ে — যদি Match পায়, তাহলে আসল Password বের করে ফেলবে!

◇ Salting এর ভূমিকা:

Salting থাকলে Rainbow Table Attack **ব্যর্থ** হয়! ✕

কারণ:

- Salt যোগ করার ফলে, একেকটা পাসওয়ার্ডের Hash একেকরকম হবে।
- হ্যাকারদের আগের তৈরি করা Rainbow Table কাজ করবে না।

✦ সব মিলে Short Summary:

বিষয়	সংজ্ঞা	কাজ
Salting	Random Value যোগ করে Hash করা	Hash আলাদা হয়, Secure হয়
Pepper	Secret Key দিয়ে Hash আরও শক্তিশালী করা	Hacker শুধু Hash পেলেও কিছু করতে না পারা
Rainbow Table Attack	পূর্ব-তৈরি Hash টেবিল দিয়ে পাসওয়ার্ড বের করা চেষ্টা	Salting দিয়ে রক্ষা সম্ভব

💧 Practical জীবন থেকে Example:

- Banking Apps → Always use Salting + Pepper + bcrypt hashing
- Facebook, Google → Password Hash stored with Salt
- Github → Secure Password Storage Standards follow করে