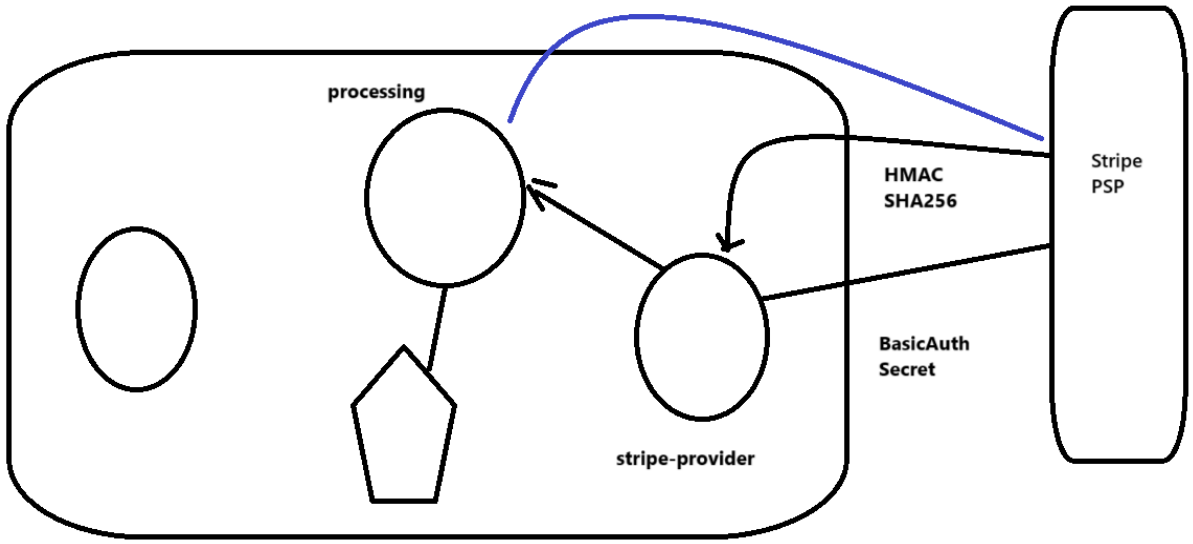
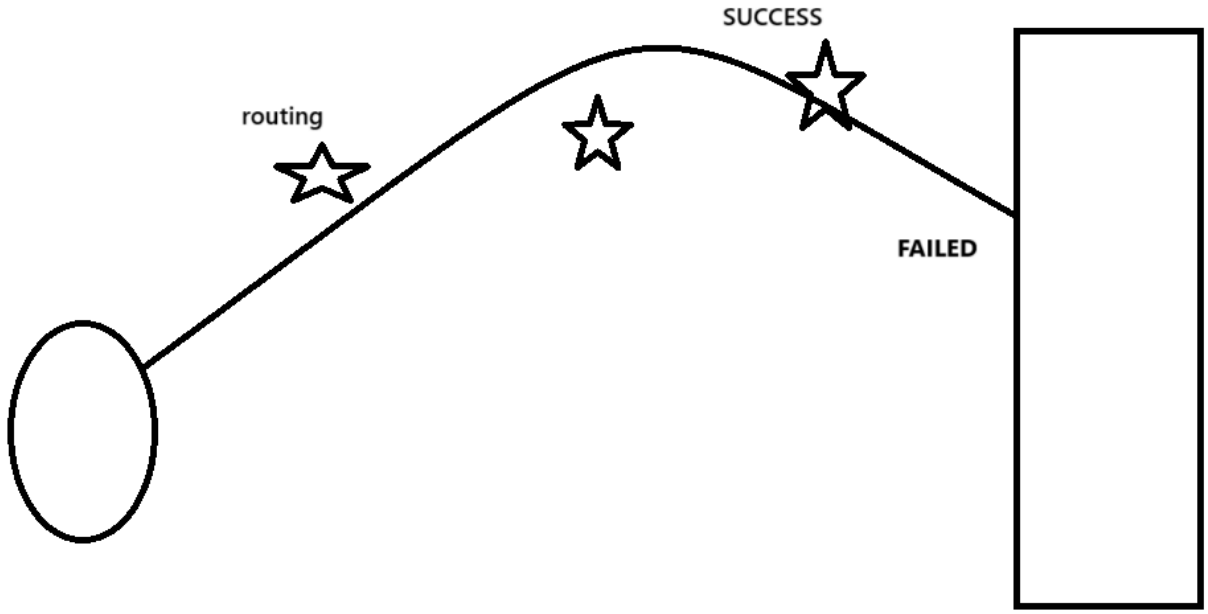
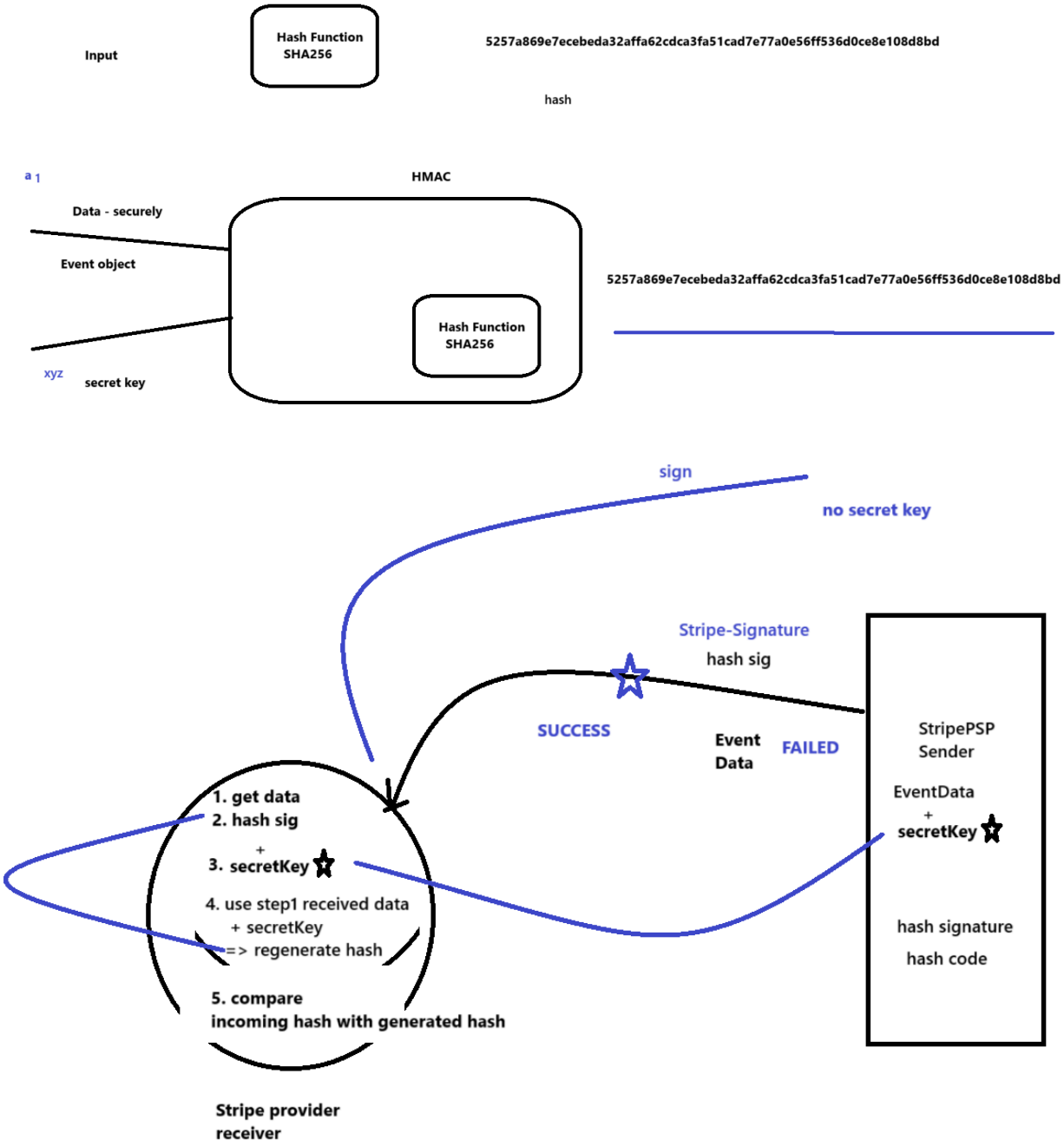
DIAGRAM


processing
HMAC
SHA256
Stripe
PSP
BasicAuth
Secret
stripe-provider


SUCCESS
routing
FAILED



LIVE NOTES

W6D4: Notification Web hook processing.

Stripe communicates status update to us by sending events/notifications/webhooks

We need to expose API call, which Stripe is going to invoke.
    RestController

For interpreting success
    Depend upon these webhooks
    failure can also happen via webhook

We will receive
checkout.session.completed
checkout.session.async_payment_succeeded

    "status": "COMPLETED"
    "payment_status": "paid"

    Successful
    updating our SUCCESS

checkout.session.async_payment_failed
    Update the status as FAILED in DB
    errorCode & errorMessage

When someone pays you, it creates a checkout.session.completed event. Set up an endpoint on your server to accept, process, and confirm receipt of these events.

1. Our GoLIVE implementation approach
    we will expose stripe notification webhook / endpoint from stripe-provider service.
Accordingly for valid notifications events stripe-provider will invoke processing-service.
    - process-service updates the DB for success/failed

Stripe-CLI

1. Code endpoint /stripe/webhook
2. stripe listen --forward-to localhost:8082/stripe/webhook
3. Further testing - coding

One round of testing where using stripe-cli, we send notification to this endpoint.
    Call Stripe createSession, get url, make successful payment on the url, & check, if we

are receiving notification..
	processing-service

with every notification, we will receive event object.
https://docs.stripe.com/api/events/object

For different events its possible, that we will receive different object structure.

How do we define Java object to handle such dynamic object structure
we need to convert json string into java object so that we can process it.

StripeEvent:
	id
	object
	StripeEventData data
	type

StripeEventData {
	??? object
}

Implement security check before processing stripe notification

https://docs.stripe.com/webhooks

- For running notifications on PROD (also on aws env - if you have ssl https) - then configure your webhook endpoint in Stripe Dashboard.
- For local testing (when you don't have valid ssl), then use stripe-cli

for ensuring security on notification endpoint.
	stripe is offering 2 approaches
	1. using their libraries (recommended)
	2. manual verification

Stripe have already written some java code which will ensure security. And this code is available in Jar. We need to add this jar in our classpath & work with it.

HmacSHA256:
1. Authentication
    (verifying that request is coming from stripe PSP)
2. Data integrity
    (data is not tampered. whatever stripe psp has sent, same is received at our end.)

Hmac - SHA256 (hash function)
hash-based message authentication code (HMAC) hash generation

At receiver end:
    we will again calculate hmacsha256 hash signature. Compare this with incoming signature.
    if both are same, means authentication & data integrity is ensured.
    Means data is coming from stripe psp & is not tampered over the wire. Whatever stripe psp has pass, same data is being received at our end.

Secret key is shared only between sender & receiver. Nobody else knows it.

hmacsha256
    for the given input, it will always generate same hash output.
    for any change in input, new output will be generated.

Authentication is ensured.
    bcoz only stripe (sender) has this secretkey. no one in world can generate sign which will be generated by our sender.

in event handling:

1. Read Data @RequestBody
1.1 how to get incoming signature @RequestHeader
2. Get the signing secret - from stripe cli

Manually / stripe library

3. use HmacSHA256 logic to regenerate new signature
4. our generated signature match with incoming coming

https://docs.stripe.com/api

```
<dependency>
  <groupId>com.stripe</groupId>
  <artifactId>stripe-java</artifactId>
  <version>28.4.0</version>
</dependency>
```

handle generic Exception

```
return ResponseEntity.badRequest().build();
return ResponseEntity.ok().build();
```

```
try {
	Webhook.constructEvent(
		eventBody, signatureHeader, endpointSecret);
		log.info("Successfully verified webhook signature");
} catch (Exception e) {
	log.error("Invalid signature");
	return ResponseEntity.badRequest().build();
}
```

if exception comes means invalid signature. return 400
if no exception means signature is valid.. continue further processing.