haventec

HAVENTEC AUTHENTICATE

INTRODUCTION

We are living in an era of truly remarkable change. Industries and enterprises are being transformed by new technologies. Incumbents are being disrupted. Disruptors are redefining markets and transforming customer expectations. With more than half the world's population now online, organisations seeking to remain relevant are relying on digital technologies to underpin access to, and meaningful engagement with, their markets. This rapidly expanding connectivity is in turn amplifying the impact of bad actors. The increasing sophistication and frequency of cyber-attacks, as well as the increasing impact of basic human error, are heightening concerns about privacy. Violations of privacy are driving a new focus from regulators around the world, each mandating new rules and data rights to both protect consumers and sanction increasing criminal penalties for data breaches.

Global economies have been able to thrive based on the existence of trust between transactors in an exchange. The future of open government and the open enterprise is heavily dependent on the existence of digital trust. However, current technologies are providing inadequate protection to the privacy, misuse, or theft of sensitive data. This is the acute and unresolved challenge facing the digital world. The core digital stakeholders (consumers / enterprises / regulators) are all seeking a solution to the inherent weaknesses of the current digital trust model. Addressing the weaknesses will require redesigning the foundations of digital trust which involves:

1. revisiting the need to centrally store sensitive data;
2. changing the way applications interact with that data;
3. changing the way individuals, the enterprise and third-parties access that data; and
4. providing individuals with more control over how their personal information is used and shared.

At the core of the Haventec proposition is a remarkably simple idea, beautifully executed – protecting digital identities from harm, and keeping 'data that matters' safe – the key foundations for digital trust. Haventec have developed a suite of technologies that eliminate the need to centrally store sensitive credentials and data, providing the ultimate preparedness for when the enterprise is breached. Due to the decentralised nature of these innovative technologies, the sensitive data and credentials currently stored centrally are not there.

Haventec delivers a haven for sensitive credentials and data.

1. For consumers and citizens, it provides them the capacity to extract value from their personal digital transaction data while maintaining their individual right of privacy.

2. For governments and regulators, it provides a credible alternative to the current approaches adopted for protecting the sensitive data of individuals, promoting the safeguarding of the broader interests of society by sanction for mass data and credential breach.

3. For digital suppliers and data custodians, it allows them to extend controls to their customers, placing them in charge of how their data is shared and used. It eliminates the risk of mass credential and data breaches and provides additional controls to the enterprise to better manage regulation of individual privacy that is in the best interests of the business, its shareholders, the Boards of Directors, and their customers, thereby strengthening trust.

WHAT IS AUTHENTICATE

Haventec's Authenticate platform protects digital identities from harm. Authenticate breaks access to your identity into multiple parts, separating them across multiple locations. Two sets of keys are changed and re-encrypted for every Authenticate transaction.

When using Authenticate to access a digital service, the user simply enters a secret [this secret is never stored anywhere]. Authenticate includes password options and a true password-free option – one that eliminates operational risk as it does not require a central credentials store to be maintained, unlike other "password-free" offers in the market that simply substitute a password for another factor, but continue to store the password centrally.

Authenticate's architecture is quantum computer resistant, a matter that will become more pressing in relevance during the next four years.

HOW DOES IT WORK?

Haventec Authenticate's rolling key encryption offers a simplified and fully customisable experience with multi-factor encryption with every authentication.

Identity and account fraud prevention including protection against:

- Password cracking;
- Phishing & Pharming attacks;
- Social Engineering attacks;
- Shoulder Surfing attacks; and
- Mass account breaches.

To view a video demonstration on how Haventec Authenticate works, please click on this link: https://youtu.be/O3TTjNL6E1I

AUTHENTICATE API OVERVIEW

Haventec's Authenticate decentralised identity platform provides a suite of API's for securing digital identities. The Authenticate APIs are organised as eight functional controllers that enable developers to easily leverage and integrate the Authenticate capabilities into their application user experience flows.

1. authentication-controller : Authentication Controller
2. device-controller : Device Controller
3. iam-controller : IAM Controller
4. jwt-controller : Jwt Controller
5. open-id-controller : Open ID Controller
6. otp-controller : Otp Controller
7. self-service-controller : Self Service Controller
8. user-controller : User Controller

For more detailed information on Haventec Authenticate APIs, please click on this link: https://api.haventec.com/authenticate/v1-2/api-docs.html

AUTHENTICATE PROCESSES

REGISTER NEW USER

This following description outlines the technical algorithmic flow implemented in the register new user process:

1. The user will enter a username (U1) and 4-digit pin (P1).
2. The user's device will generate a 512-bit random salt, which will be stored as Base64 encoded string on the device specific local storage (S_1).
3. The user's device will append the 4-digit pin with the Base64 encoded 512-bit salt and hash it using SHA-512 resulting in a Base64 encoded hash known as the hashed pin (HP_1).
4. The username (U1) and hashed pin (HP1) will be sent to Authenticate. The client device will gather a set of device metadata (e.g. installed fonts, OS and Browser version) and send that along with the user entered information.
5. Authenticate will generate an asymmetric keypair (Consisting of a Public Key (PK_1) and Private Key (PV_1)) (type ECC P-384 Curve).
6. Authenticate will generate a Unique Device ID ($UDID_1$) and link it to the username (U1) within the database store.
7. Authenticate will link the Public Key (PK_1) to the Unique Device ID ($UDID_1$) and device metadata within the database store.
8. Authenticate will create a reversible obfuscation of the Private Key (PV_1) using an obfuscation function and the hashed pin (HP_1). The resultant output of the obfuscation function is known as a delta (D1) of the Private Key.
9. Authenticate will generate a symmetric key (Consisting of an Initialisation Vector

(IV_1) and key (K_1)) (type AES-256).

10. Authenticate will link the Initialisation Vector (IV_1) and key (K_1) to the Unique Device ID ($UDID_1$) within the database store.
11. Authenticate will encrypt the Base64 encoded version of delta (D1) using the Initialisation Vector (IV_1) and key (K_1) using the AES-256 algorithm to form a Base64 encoded string known as the encrypted delta (or AuthKey (AK_1)).
12. Authenticate will generate a HMAC key (HK_1)
13. Authenticate will append the PEM encoded public key (PK_1) to the AuthKey (AK_1) and hash it using the HMAC algorithm (type HMAC-256) and HMAC key (HK_1), the resultant output of the HMAC algorithm is a Base64 encoded string known as the HMAC hash ($HMAC_{hash1}$)
14. Authenticate will link the HMAC hash ($HMAC_{hash1}$) and HMAC key (HK_1) to the to the Unique Device ID ($UDID_1$) within the database store.
15. Authenticate will return the Unique Device ID ($UDID_1$) and AuthKey (AK_1) to the user.
16. The user's device will link the Unique Device ID ($UDID_1$) and AuthKey (AK_1) to the username (U_1) within a device specific local storage.

At this stage, the registration transaction is complete.

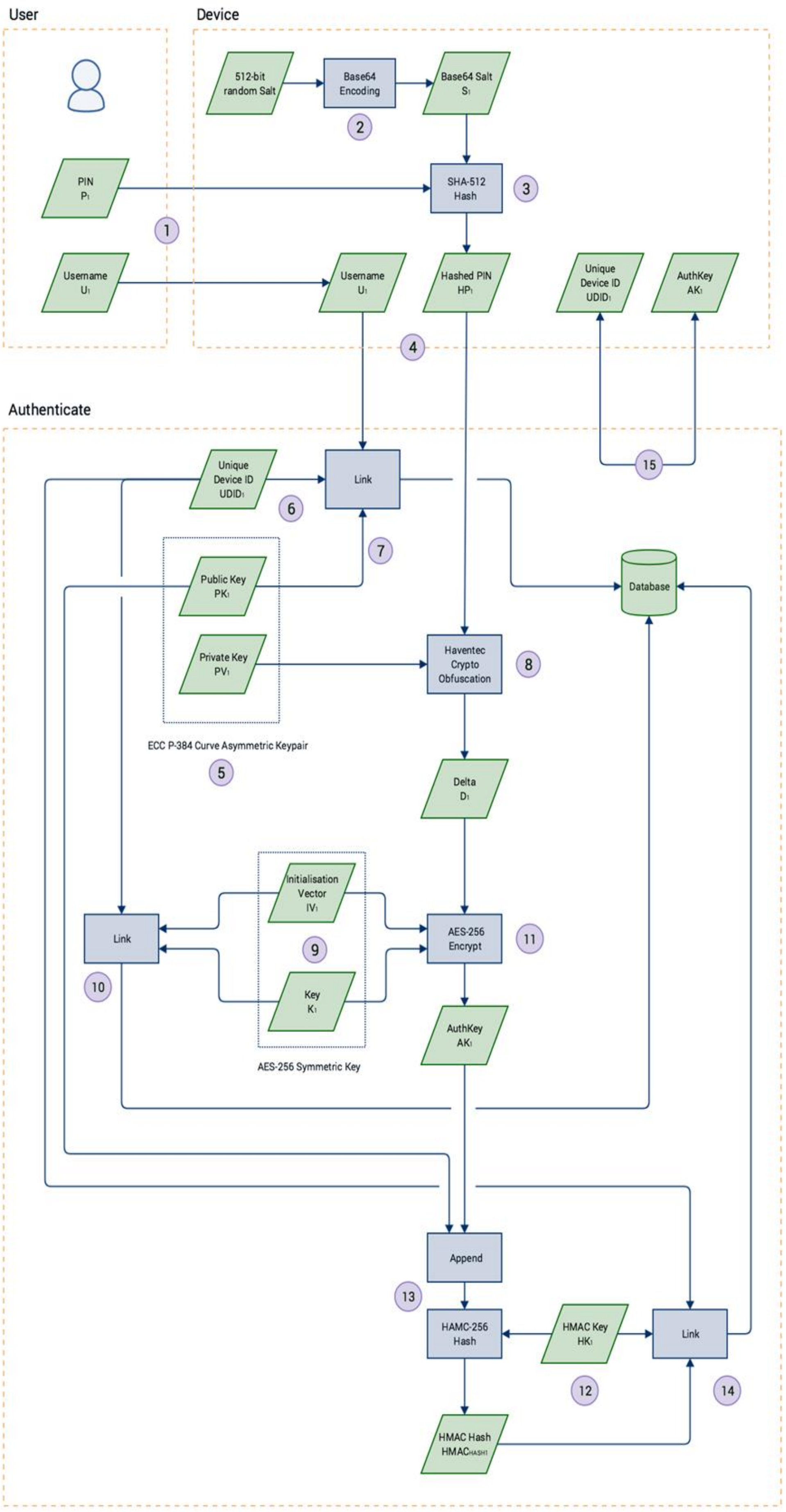


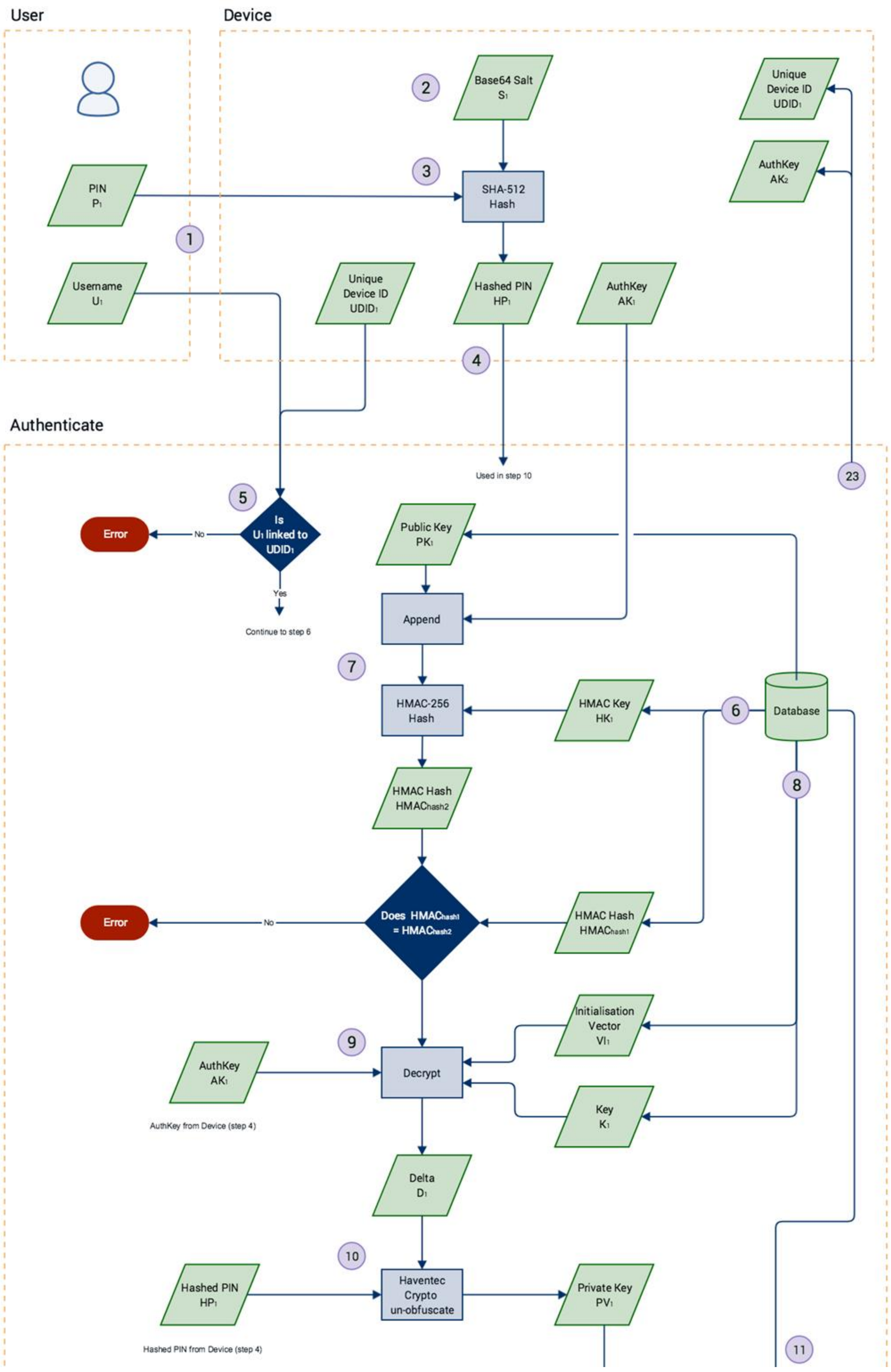
Figure 1: Register New User Flow

AUTHENTICATE REGISTERED USER

This following description outlines the technical algorithmic flow implemented in the authenticate registered user process:

1. The user will enter a username (U1) and 4-digit pin and the client device will gather device metadata (Similar to the registration step).
2. The user's device will retrieve the Base64 encoded 512-bit random salt from local storage (S1)
3. The user's device will append the 4-digit pin with the Base64 encoded 512-bit salt and hash it using SHA-512 resulting in a Base64 encoded hash known as the hashed pin (HP1).
4. The user's device will send the username (U1), Unique Device ID (UDID1), AuthKey (AK1), hashed pin (HP1) and device metadata to Authenticate.
5. Authenticate will verify the provided Unique Device ID (UDID1) and device metadata is linked to the username (U1), if incorrect, it will halt the transaction with an error.
6. Authenticate will retrieve the HMAC key (HK1) and HMAC hash (HMAChash1) based on the Unique Device ID (UDID1) from the database store.
7. Authenticate will retrieve the public key (PK1) and append it to the AuthKey (AK1) and hash it using the retrieved HMAC key (HK1) using the HMAC-256 algorithm, the resultant HMAC hash calculation is compared to the stored HMAC hash (HMAChash1) value. If they don't match, it will halt the transaction with an error.
8. Authenticate will retrieve the Initialisation Vector (IV1) and key (K1) based on the Unique Device ID (UDID1) from the database store.
9. Authenticate will decrypt the AuthKey (AK1) to get the delta (D1) using the Initialisation Vector (IV1) and key (K1)
10. Authenticate will un-obfuscate the delta (D1) using the hashed pin (HP1) to form the original private key (PV1)
11. Authenticate will retrieve the public key (PK1) based on the Unique Device ID (UDID1) from the database store.
12. The private key (PV1) is matched against the public key (PK1) by signing a random constant string using the private key, and verifying with the public key (PK1). If the signing check is valid, a successful authentication message has occurred. If the signing check is invalid, the transaction is halted and an error is sent to the user.
13. Assuming a successful authentication has occurred, Authenticate will remove all keys (public key, symmetric keys, HMAC key and hash) from the database store.
14. Authenticate will then follow steps 5-15 to in registration process above and generate a new set of keys and link them to the Unique Device ID (UDID1) in the database store.

15. Authenticate will generate a new asymmetric keypair (Consisting of a Public Key (PK2) and Private Key (PV2)) (type ECC P-384 Curve).
16. Authenticate will link the new Public Key (PK2) to the Unique Device ID (UDID1) within a database store.
17. Authenticate will create a reversible obfuscation of the Private Key (PV2) using an obfuscation function and the hashed pin (HP1) which is stored in memory. The resultant output of the obfuscation function is known as a delta (D2) of the Private Key.
18. Authenticate will generate a new symmetric key (Consisting of an Initialisation Vector (IV2) and key (K2)) (type AES-256).
19. Authenticate will link the Initialisation Vector (IV2) and key (K2) to the Unique Device ID (UDID1) within the database store.
20. Authenticate will encrypt the Base64 encoded version of delta (D2) using the Initialisation Vector (IV2) and key (K2) using the AES-256 algorithm to form a Base64 encoded string known as the encrypted delta (or AuthKey (AK2)).
21. Authenticate will generate a new HMAC key (HK2)
22. Authenticate will append the PEM encoded public key (PK2) to the AuthKey (AK2) and hash it using the HMAC algorithm (type HMAC-256) and HMAC key (HK2), the resultant output of the HMAC algorithm is a Base64 encoded string known as the HMAC hash (HMAChash2)
23. Authenticate will link the HMAC hash (HMAChash2) and HMAC key (HK2) to the to the Unique Device ID (UDID1) within the database store.
24. Authenticate will return the Unique Device ID (UDID1) and AuthKey (AK2) to the user.
25. The user's device will link the Unique Device ID (UDID1) and AuthKey (AK2) to the username (U1) within a device specific local storage.


User
Device
PIN P_1
Username U_1
1
2
Base64 Salt S_1
3
SHA-512 Hash
Unique Device ID $UDID_1$
Hashed PIN HP_1
AuthKey AK_1
Unique Device ID $UDID_1$
AuthKey AK_2
4
Used in step 10
23
Authenticate
5
Is U_1 linked to $UDID_1$
No
Error
Yes
Continue to step 6
Public Key PK_1
Append
7
HMAC-256 Hash
HMAC Key HK_1
6
Database
8
HMAC Hash $HMAC_{hash2}$
Does $HMAC_{hash1}$ = $HMAC_{hash2}$
No
Error
HMAC Hash $HMAC_{hash1}$
9
Initialisation Vector VI_1
AuthKey AK_1
Decrypt
Key K_1
AuthKey from Device (step 4)
Delta D_1
10
Hashed PIN HP_1
Haventec Crypto un-obfuscate
Private Key PV_1
Hashed PIN from Device (step 4)
11

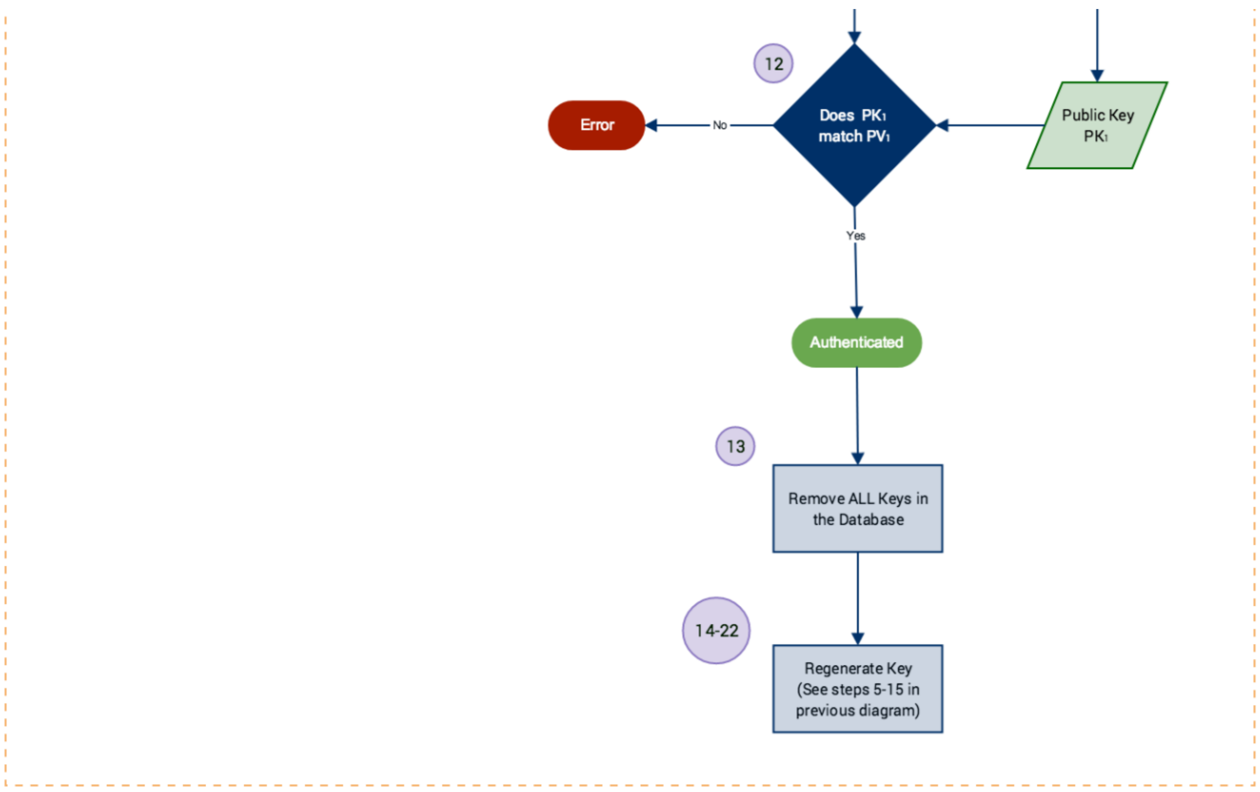

12
Does PK1 match PV1
Public Key PK1
No
Error
Yes
Authenticated
13
Remove ALL Keys in the Database
14-22
Regenerate Key (See steps 5-15 in previous diagram)

Figure 2: Authenticate Registered User Flow

NOTABLE FEATURES

- Authenticate only stores one part of the three-part puzzle required to authenticate the user – a set of cryptographic keys that, if stolen, are of no value.

- All keys stored in the Authenticate database (public key, symmetric key, HMAC key and hash) are rotated every single valid transaction.

- The Unique Device ID (UDID) stays constant for the life of the device and is not cycled

- Once a 512-bit random salt is generated on the client device, it is not cycled on every transaction. It is also not regenerated on a change of pin.

FREQUENTLY ASKED QUESTIONS

Why is the delta encrypted using a symmetric key?

To prevent client-side AuthKey (D1) brute force attempts to calculate the original private key.

What is the purpose of the HMAC key calculation?

To prevent the AuthKey and Public key from being tampered with to leak information. If tampered with, the hash calculation will be different and the transaction will be rejected.

What happens if the incorrect pin is entered several times?

The device is locked after a defined number of incorrect attempts. This device locking threshold limit is a fully configurable parameter.

Why is the PIN hashed on the client?

To prevent the plaintext PIN being viewable in Authenticate memory, further protecting the PIN from theft and misuse.

TALK WITH US ABOUT SECURING YOUR SENSITIVE DATA

info@haventec.com

haventec.com