ADUCID

PersonalCode™

This document dives deep into the PersonalCode™ design, that is part of ADUCID Universal Digital Identity solution. Cutting-edge technology is supported by several patents that are explained in detail throughout the paper.

PersonalCode™: How it differs from other security codes?

PersonalCode™ technology brings a combination of competitive **advantages** in the domains of **usability and security:**

- **Two-layered keyboard** enables the user experience with the first layer and enhances security with entropy in the second layer.
- User has a **single PersonalCode™** across all devices for authentication to all of their online accounts.
- **PersonalCode™ is not stored anywhere** – not in the user's device, nor the target service systems. If an attacker compromises a target system, the PersonalCode™ itself is not compromised.
- **Each service provider** that uses the PersonalCode™ knowledge factor **verifies a different value** - no two services technically store the same PersonalCode™. Compromising one system does not compromise another system.

Biometrics: Is it appropriate as an end-to-end second factor?

How to secure biometrics? This is a seriously difficult task. Processed biometric information is static. It is used, and therefore also shared, between different systems. It can thus be acquired from other sources that are out of scope of an identity system. It can also be locally imitated with quite simple techniques. In case of remote authentication, identity can be imitated using increasingly more sophisticated deepfake algorithms.

Is recovery possible? No! Once biometrics are compromised, they cannot be revoked or used again. The source of biometric data is the person herself and cannot be changed.

In practice, identity proofing and binding cannot be achieved. Current systems cannot pair a specific user with pecific **local biometrics**. Touch ID, as an example, can be shared between users. The security scope of the fingerprint goes way beyond the security scope of any Identity Provider (IdP). Moreover, there is no way to pair specific cryptomaterial on the phone with a specific finger that belongs to a particular user. On the other hand, using **remote biometric** verification raises serious security concerns.

PersonalCode^{TM}: A secure alternative to password?

Let's compare the PersonalCode^{TM} knowledge factor with legacy passwords. Several factors drive the motivation behind the development and design of the PersonalCode^{TM}:

People use simple passwords and usually duplicate one password multiple times. It is difficult for users to remember long and randomized secrets that differ between independent systems. The result is, people in practice write the secrets down, or store them in password manager software, which changes the nature of the "knowledge factor." In this case, it is no longer a knowledge factor but a possession factor.

Attackers can compromise stored secrets. A fraudulent service provider can, for example, get the user's password by phishing and this way, get control of the knowledge factor. Another problem is that the secret is typically shared between systems. Most users reuse passwords between different systems, and the target systems have no way of mitigating this. It commonly occurs in a situation that the user has a password for a government IdP and utilizes it for a poorly secured blog. An attacker can compromise this password by attacking a blog, and subsequently compromising the IdP's knowledge factor. Note that this is very common in practice.

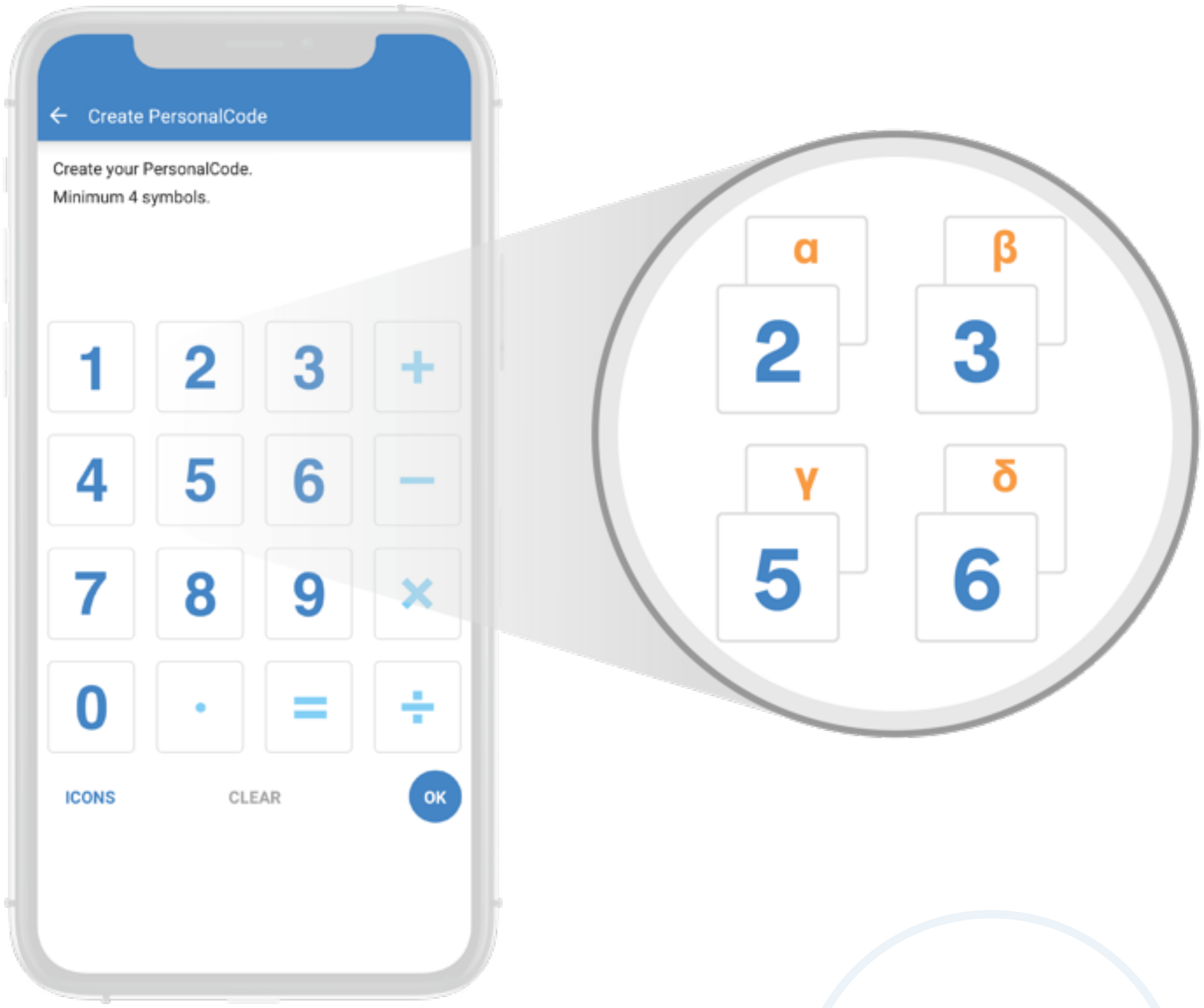
The problem of the dictionary attack. People can only remember very short sequences, and, in practice, users choose words that add meaning to make it easier to remember. The patterns that arise from the given meaning significantly reduces the desired entropy and attackers can map passwords from its hashed value.

Design

Picture Keyboard

The **Picture Keyboard** is a **patented technological method that is used as a user interface for entering a knowledge factor on the user's device.** The design goal is to significantly increase the level of entropy and prevent "dictionary attacks". The method functions on the principle of layering. Even if two users have the same PersonalCode^{TM}, the values in the second layer are actually different from what the user enters. The positioning of individual keys is different for every authentication to mitigate the risk of physically compromising the code. The keyboard is layered as presented in the diagram below.

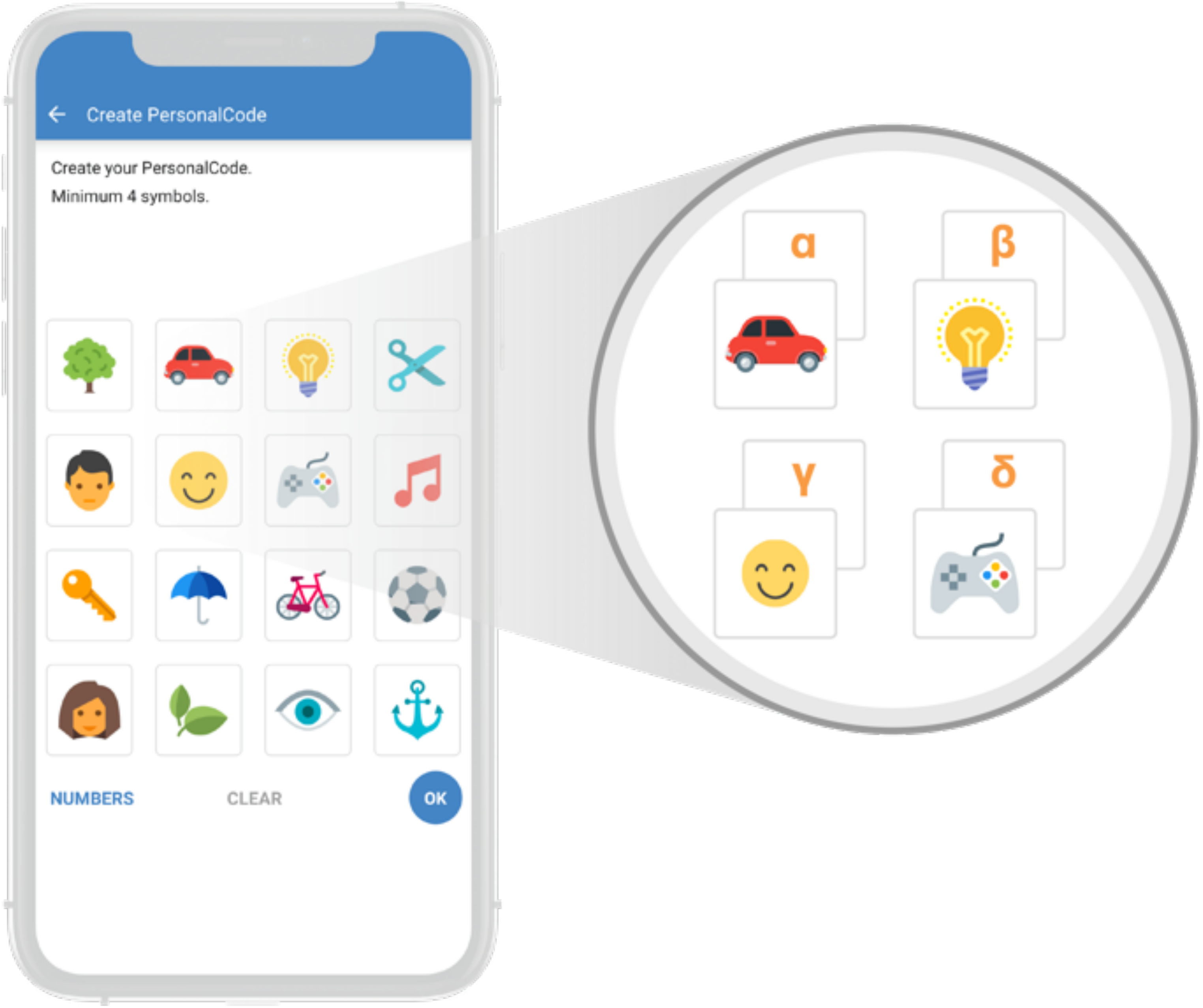
Diagram 1: Picture Keyboard – layers explained



Notice in the diagram that **each key (2, 3, 5,..) is associated with a second value (α, β, γ,..)**. These values are randomly generated and have significant entropy (20-byte value for each key in the ADUCID Picture Keyboard implementation). Numbers that the user selects are not actually used in authentication – in fact, they are not stored as numbers. Instead, they are pictures/images and can be represented in any way. The user presses the keys "2356", in this case, and a sequence comprised of the values "αβγδ" is created – let's call it a **sequence Z**.

PersonalCode^{TM} **keyboard** can be applied in the form of **numbers** (*Diagram 1*), or **icons** for better memorability, as shown below in *Diagram 2*.

Diagram 2: Picture Keyboard – icons



Create your PersonalCode.
Minimum 4 symbols.

Dual Authentication & Vector Authentication Introduction

Dual Authentication method enables authentication by a "**secret**", that is not stored anywhere, which is a principle that is used by the PersonalCode^{TM} technology. For this reason, it cannot be in principle brute-forced even if the attacker has access to the authentication server.

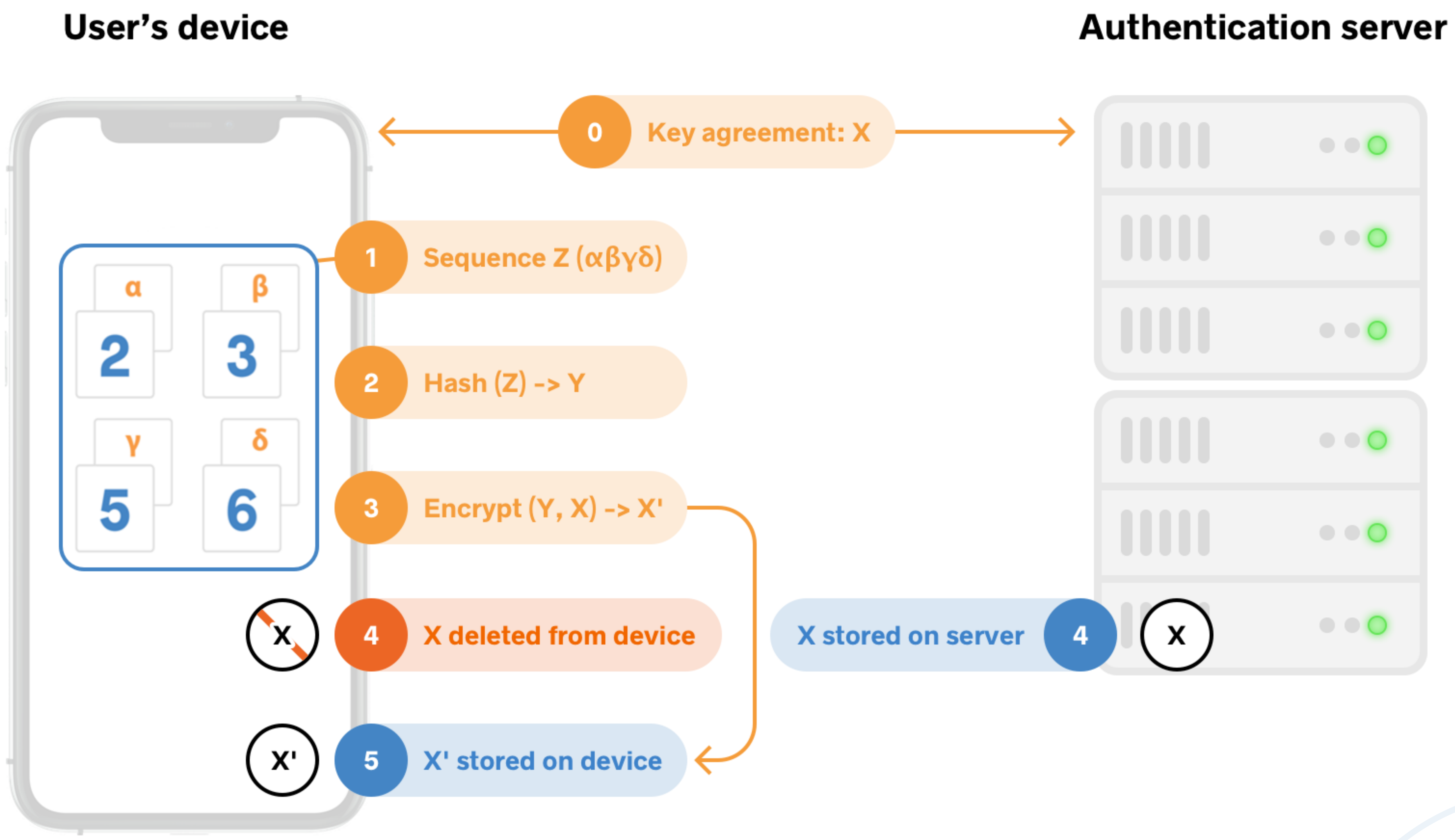
Vector Authentication is a security mechanism that enables advanced additional end-to-end security measures for a previously authenticated communication. This is also used by the PersonalCode^{TM} in the ADUCID technology implementation. By using the Vector Authentication method, PersonalCode^{TM} is authenticated end-to-end with the target service by using:

- Picture Keyboard to enter the code,
- Dual Authentication method to ensure knowledge-factor authentication without storing the "**secret**" anywhere.

PersonalCode^{TM} Creation

The following diagram and steps below describe interactions and a process of creating the PersonalCode^{TM}.

Diagram 3: PersonalCode Creation



Creation of the PersonalCode^{TM} is done in the following steps:

0 **Primary authentication of a possession factor.** Possession factor is represented by a digital identity app Peig on user's device. Primary authentication creates a mutually shared **Key Agreement: X** between Peig and an authentication server.

1 User is prompted to **enter the PersonalCode^{TM} on the Picture Keyboard**, the result of which is a sequence **Z (αβγδ)**, as shown in *Diagram 1*.

2 Sequence **Z** is immediately hashed; the resulting value **hash(Z) -> Y**

3 **X** is encrypted with **Y: encrypt(Y, X) -> X'**

4 **X** is **stored on the server and deleted on the user's device.**

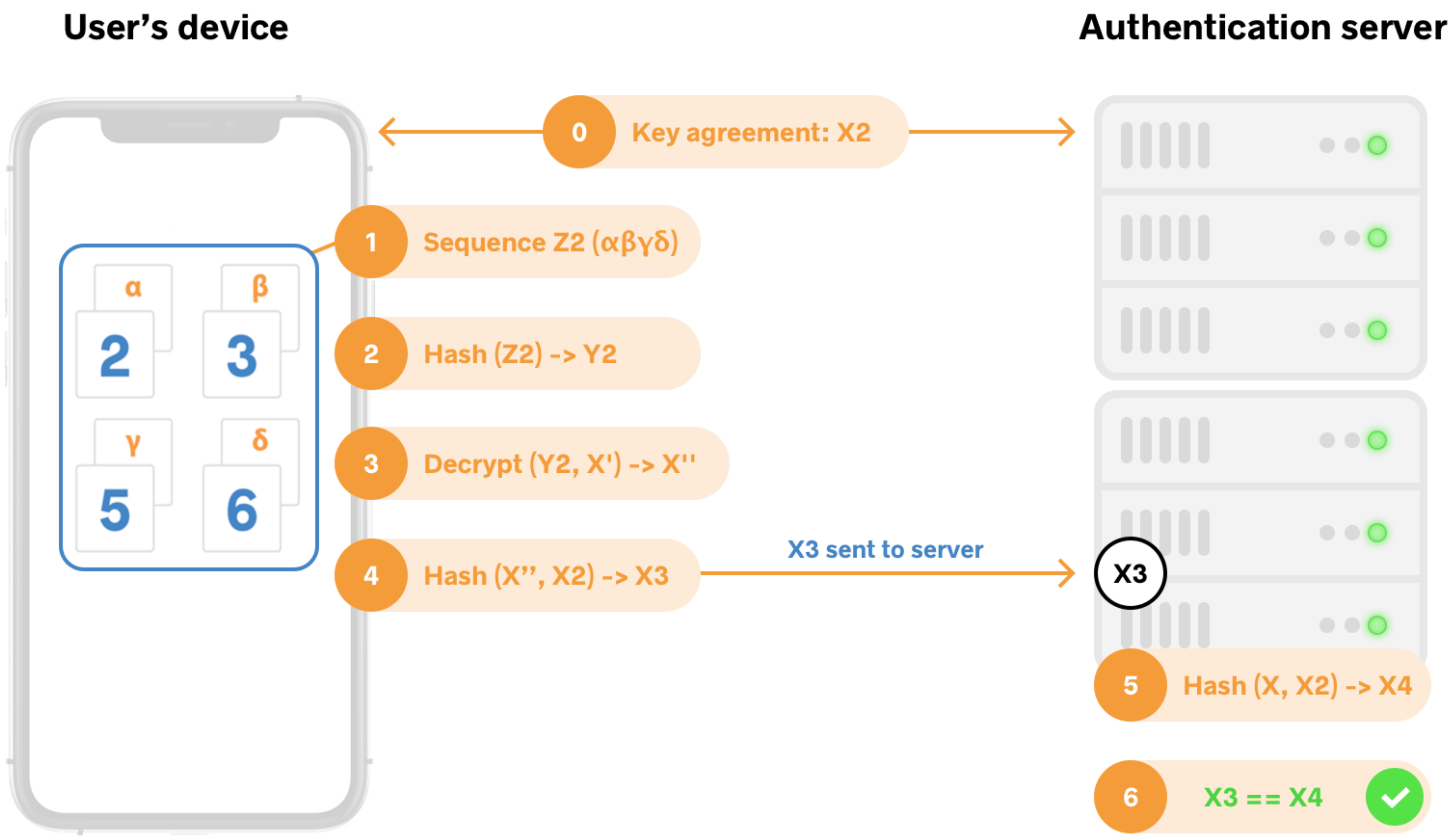
5 **X'** is **stored on the user's device.**

The user has created their PersonalCode^{TM} to be used for verification.

PersonalCode™ Verification

The following diagram and steps below describe interactions and a process of verifying the PersonalCode™.

Diagram 4: PersonalCode™ Verification



Verification of the PersonalCode™ is done in the following steps:

0 As in creation, the requirement is a previous and successful mutual **primary authentication of a possession factor**, the result of which is a **Key Agreement: X2**, which is only used once.

1 User enters some PersonalCode™ sequence, the result of which is **Z2**. It is not known whether the entered sequence is the correct one.

2 As in creation, **sequence Z2** is hashed: **hash(Z2) -> Y2**. It is not known whether **Y2** is equal to the **Y** that was used in creation.

3 **X'** is decrypted with **Y2: decrypt(Y2, X') -> X''**

4 The value of **X''** and **Key Agreement: X2** are together hashed and sent to the authentication server as part of Vector Authentication: **hash(X'', X2) -> X3**

5 On the authentication server **X4** is created from saved X (from creation) and **X2: hash(X, X2) -> X4**

6 **X3 == X4** is compared on the server.

If the values are equal, the correct PersonalCode™ was entered.

One PersonalCode^{TM} on multiple user devices

User can have multiple eID devices with the same PersonalCode^{TM}, which they can use to access all online services and assets in an entire ecosystem.

Easy solution for critical situations. If one of user's devices is lost or stolen, the device is broken, or if the user simply buys a new one, they can immediately deactivate or migrate to one of their other devices without downtime or losing access to online assets. The PersonalCode^{TM} enables this in the following ways:

- **Security** – all user's devices use the same PersonalCode^{TM}. In case the PersonalCode^{TM} is used correctly on one device, but compromised on the other, physical device theft can be detected.
- **There is only one PersonalCode^{TM} for all systems.** The user doesn't have to worry about creating it, changing it or managing its security. When the PersonalCodeTM is changed in one system, it is automatically changed for all authentication relations in other systems as well.

All of the above apply for both centralized and distributed ecosystem designs.

PersonalCode™ in a Distributed Ecosystem

Security advantage of the PersonalCode^{TM} design can be further scaled in a Distributed Ecosystem design. Additional security benefits arise from the **Dual Authentication** and **Vector Authentication** method design.

Dual Authentication

The secret pairs X' & X that are described above are not global. Instead, they are created between a client app and an authentication server. In a distributed model, where users verify the PersonalCode^{TM} directly with the target service, the values X' & X for a single user always correspond only to one target service. In extraordinary cases when the PersonalCode^{TM} on the target service is compromised (e.g. attacker has access to service providers databases, or attacker is a service provider himself who willingly compromises the knowledge factor database), other services are not affected.

Vector Authentication

The Vector Authentication method, which is used by the PersonalCode^{TM} design, can be used for additional security mechanisms that improve the security of the whole authentication system. In the ADUCID technology, these include data channel binding, anti-copy detection, eliminating MITM in identity proofing etc. The vector authentication method enables flexibility of authentication security scaling.