Password-Based Credentials with Security against Server Compromise

Dennis Dayanikli and Anja Lehmann

Hasso-Plattner-Institute, University of Potsdam
{dennis.dayanikli, anja.lehmann}@hpi.de

Abstract. Password-based credentials (PBCs), introduced by Zhang et al. (NDSS'20), provide an elegant solution to secure, yet convenient user authentication. Therein the user establishes a strong cryptographic access credential with the server. To avoid the assumption of secure storage on the user side, the user does not store the credential directly, but only a password-protected version of it. The ingenuity of PBCs is that the password-based credential cannot be offline attacked, offering essentially the same strong security as standard key-based authentication. This security relies on a secret key of the server that is needed to verify whether an authentication token derived from a password-based credential and password is correct. However, the work by Zhang et al. assumes that this server key never gets compromised, and their protocol loses all security in case of a breach. As such a passive leak of the server's stored verification data is one of the main threats in user authentication, our work aims to strengthen PBC to remain secure even when the server's key got compromised. We first show that the desired security against server compromise is impossible to achieve in the original framework. We then introduce a modified version of PBCs that circumvents our impossibility result and formally define a set of security properties, each being optimal for the respective corruption setting. Finally, we propose a surprisingly simple construction that provably achieves our stronger security guarantees, and is generically composed from basic building blocks.

1 Introduction

Password-based authentication is still the most common form of user authentication online. Their main benefit is convenience: users can access their accounts from any device based on human-memorizable information only. On the downside, passwords provide weak security guarantees. The biggest threats are server compromise, i.e., an attacker gaining access to the password data stored on the server side, and weak passwords that can be (online) guessed.

To provide better security for users, strong authentication solutions such as FIDO [19,24] see an increasing interest in the industry and among standardization communities. In these solutions, the user typically owns a cryptographically strong signing key, and authenticates by signing a challenge provided by the server who stores the corresponding public key. This solution eliminates both the risk of guessing attacks (the user now has a high-entropy key) and server compromise (the information on the server side is only the user's public key, i.e., not sensitive). However, this strong security comes for the price of reduced usability, as the user must securely manage cryptographic key material. This is particularly challenging when users want to access the key from many, and possibly low-security, devices. A common approach therefore is to rely on tamper-resistant hardware tokens, e.g. Yubikey [26], which is desirable from a security perspective, but clearly not ideal in terms of usability [20].

Password-Based Credentials. To combine the best of both worlds, Zhang et al. [27] recently proposed the concept of *password-based credentials* (PBC) that provide similarly strong security as the key-based solution, but without having to store sensitive key material on the user side. In the PBC-system, the user establishes a cryptographically strong access credential with the server upon registration. To avoid the need of secure hardware on the user side, the user does not store the sensitive credential directly, but only a password-protected version of it. When authenticating to the server, the user needs both the credential and her password. The twist of their solution is that this password-based credential is resistant to offline brute-force attacks against the password, and thus could even be synced via (untrusted) cloud providers or simply copied on many (low-security) devices. This offline-attack resistance is achieved by relying on a high-entropy key of the server for verifying whether an authentication token derived from the credential and password is correct. Thus, verifying whether a password guess was correct requires interaction with the server, which reduces the attack surface from offline to online attacks if an attacker knows the password-based credential. If the adversary does not possess the user's password-based credential, the security is essentially equivalent to strong authentication. Their security comes with one significant limitation though – it assumes the server never gets compromised.

Importance of Server Compromise. Server compromise is a major threat to password-based authentication, and refers to an attack where the adversary gains access to the authentication information maintained by the server, such as password hashes. The server itself is considered to be honest, but an attacker can now recover the users' access details to either gain access to a user's account at the compromised server or, if the same password is re-used across multiple services, even impersonate the user on different sites. Even major companies such as Yahoo [25], PayPal [11], Linkedin [1], Blizzard [21] or LastPass [23] have suffered from such attacks, resulting in millions of password hashes or password-protected files being compromised.

Thus, considering the threat of server compromise and building solutions that maintain security in such scenarios is crucial for end-user authentication. Surprisingly, despite having server compromise as a core motivation for their work, Zhang et al. [27] do not include server compromise attacks in their model. In fact, their PBC protocol loses all security if the server's data gets compromised, as the attacker can then impersonate any user who has registered with the server. This even holds regardless of the user's chosen password, as it does not require any additional offline attack to recover the password.

1.1 Our Contributions

We address the problem of password-based credentials that remain secure in the presence of server compromise. We show that the desired security is impossible to achieve in the framework proposed by Zhang, Wang and Yang [27] (henceforth called the ZWY framework). We adapt this framework to circumvent the impossibility result and propose a generic protocol that provably satisfies our stronger notion – and is even simpler than the one by Zhang et al.

Following the work by Zhang et al. [27], we formalize PBC as a password-based token scheme, i.e., the actual authentication protocol is abstracted away. On a high-level, the user registers with the server, obtaining a credential that is protected under her password. After registration, the user can generate a token by "signing" her username and message (which typically will be a fresh nonce in the actual authentication protocol) using the credential and password as a secret key input. The server verifies that token using it's *secret* verification key.

We extend and strengthen the ZWY security framework to capture the following high-level security guarantees:

Strong Unforgeability: An attacker without knowledge of the user's credential should not be able to forge an authentication token – thus essentially guaranteeing the same level as classic key-based strong authentication. This property must also hold when the adversary knows the user's password, and when the server is compromised, i.e., even if the adversary knows the server's verification key.

Online Unforgeability: When the adversary knows the user's credential (but not the server's verification key), tokens remain unforgeable as long as the adversary has not guessed the correct password. The strength of this property is that the adversary must not be able to offline attack the password but run an online attack against the honest server. Requiring participation of the server for each password guess, enables the server to notice suspicious access patterns and impose throttling on the affected account.

Offline Unforgeability: If both the user's credential and the server's key are compromised, the attacker can unavoidably test passwords in an offline way. However, we require the attacker to perform such an offline attack on *each* password. This adds a last layer of security for users with strong passwords.

The ZWY framework captures a security definition for a combined version of online unforgeability and a weaker form of strong unforgeability where the server could not be compromised. Their work did not cover or achieve offline unforgeability.

Impossibility of Security Against Server-Compromise in Single-Key Setting. In the ZWY framework [27], the server only has a single verification key for *all* users. The high-level idea of their concrete construction is as follows: the server has a global MAC key, and the credential is essentially a server's (algebraic) MAC on the username which the user encrypts under her password. The core idea of authentication is decrypting the credential with the password, recovering the MAC and sending it back to server (and bound to the message). Without knowing the server's high-entropy key, one cannot verify if the decrypted value is indeed a correct MAC, ensuring the desired online unforgeability. It is easy to see that this construction is not secure if the server's key got compromised, as the adversary can simply create MACs for all users he wants to impersonate.

In fact, we show that this is not merely a weakness of their scheme but inherent in the overall *single*-key setting. That is, we show that strong unforgeability and offline unforgeability are impossible to achieve when the server owns a single verification key for all users.

Framework for Multi-Key Password-Based Credentials. As two of the three desired security properties are impossible to achieve in the single-key ZWY framework [27], we propose a new variant – *Multi-key Password-based Credentials* (mkPBC) – where the server maintains an individual verification key for every user. Moving to a setting where the server maintains individual verification information for each user requires an additional property also concerned with server compromise, yet not captured by any of the three properties listed above:

Pw-Hiding: The server's verification key for a user should not leak any information about the user's password.

The reason this property is not covered by the unforgeability notions discussed above is that learning the password in the mkPBC scheme does not allow the server to impersonate the user (this still requires the user credential). However, as users tend to reuse their passwords across different sites, we want the password to remain fully hidden in case the server gets compromised.

We formally define all four properties through game-based security definitions, capturing the optimal security guarantees for a mkPBC scheme.

Simple Construction From Standard Building Blocks. Finally, we present a surprisingly simple generic mkPBC scheme ($\mathsf{PBC}_{\mathsf{StE}}$) constructed from standard building blocks – a pseudorandom function, public-key encryption and signature scheme. The challenge is in formally proving that it achieves all our security notions. To do so, we require the signature scheme to satisfy two properties in addition to unforgeability – *complete robustness* and *randomness injectivity*. Both are natural properties, and we show that they are achieved by standard signature schemes, such as Schnorr and DSA.

Interestingly, our construction does not only provide stronger security than the original scheme, but is also much simpler and generic: Whereas Zhang et al. [27] gave a concrete discrete-logarithm based construction that required the q-SDH and q-DDHI assumptions, our $\mathsf{PBC}_{\mathsf{StE}}$ only requires basic building blocks, and thus can be easily implemented using standard cryptographic libraries. The generic approach also allows to obtain a quantum-safe variant of our scheme if the generic building blocks are instantiated with PQC-variants.

2 Single-Key Password-Based Credentials

This section presents the idea and security of the ZWY framework by Zhang et al. [27], to which we refer to as *single-key* password-based credentials (skPBC). We show that no skPBC can achieve security in the presence of server compromise, which we consider a crucial goal and which motivates our switch to *multi-key* PBCs in the following section.

We start by presenting the definition of single-key PBCs before we present the impossibility result. We adopted the ZWY framework to our notation for consistency with our main result. For completeness, we summarize our editorial changes to the ZWY syntax and security definitions in Appendix A.2 and why they do not change the technical aspects of [27].

2.1 The ZWY Syntax

A (single-key) password-based credential system consists of two phases – a *registration phase* and a *signing phase* – and involves two parties: a server $\mathcal{S}$ and a user $\mathcal{U}$ who wishes to authenticate to the server. In the registration phase, the user registers herself at the server with a username uid and password pw. The server issues her a credential ask (= authenticated secret key) using a long term key ssk and stores her username in his database.

While the overall goal is to use PBC for user authentication, where $\mathcal{U}$ and $\mathcal{S}$ engage in a challenge-response protocol, this is abstracted away in PBCs by modeling a special type of authentication token τ. This token is created by the user for a (challenge) message m and username uid, using the user's credential ask and password pw. Verification is a secret-key operation and allows the server with key ssk to verify whether the message m was indeed signed by user uid. To provide a more formal definition, the single-key password based credential system can be stated as:

Definition 1 (Single-key Password-based Credential). *A single-key password-based credential scheme* $\mathsf{skPBC} = (\mathsf{KGen}, \langle\mathsf{RegU}, \mathsf{RegS}\rangle, \mathsf{Sign}, \mathsf{Vf})$ *with message space* $\mathcal{M}$*, and* $\mathcal{D}_{\mathsf{uid}}$*,* $\mathcal{D}_{\mathsf{pw}}$ *denoting the spaces of usernames and passwords respectively, is defined as follows:*

$\mathsf{KGen}(1^\lambda) \to (ssk, spk)$*: Outputs a server key pair* (ssk, spk)*.*

$\langle\mathsf{RegU}(spk, uid, pw), \mathsf{RegS}(ssk, uid)\rangle \to (ask; -)$*: An interactive registration protocol.* RegU *is the user's algorithm and receives as inputs the server public key* spk*, a username* $uid \in \mathcal{D}_{\mathsf{uid}}$ *and password* $pw \in \mathcal{D}_{\mathsf{pw}}$*.* RegS *is the algorithm of the server and receives as input its secret key* ssk *and user id* uid*. After successful registration, the user outputs a credential* ask *(the server does not have any output).*

$\mathsf{Sign}(uid, ask, pw, m) \to \tau$*: Generates an authentication token* τ *on message* $m \in \mathcal{M}$ *and username* uid *using* ask *and* pw*.*

$\mathsf{Vf}(ssk, uid, m, \tau) \to 0/1$*: Outputs 1 if authentication token* τ *is valid on* uid *and* m *under* ssk*, and 0 otherwise.*

We require all honestly generated authentication tokens using the correct combination of ask and pw to pass validation under ssk. More formally, the correctness of single-key PBC is defined as follows:

Definition 2 (Correctness (skPBC)). *A single-key PBC scheme is correct, if for all* $(ssk, spk) \leftarrow \mathsf{KGen}(1^\lambda)$*, for all* $(uid, pw) \in \mathcal{D}_{\mathsf{uid}} \times \mathcal{D}_{\mathsf{pw}}$*, for all* $m \in \mathcal{M}$ *it holds that* $\mathsf{Vf}(ssk, uid, m, \mathsf{Sign}(uid, ask, pw, m)) = 1$ *where* $(ask; -) \leftarrow \langle \mathsf{RegU}(spk, uid, pw), \mathsf{RegS}(ssk, uid)\rangle$.

2.2 Security Model of ZWY [27]

Zhang et al. [27] proposed the security definition *Existential Unforgeability under Chosen Message and Chosen Verification Queries Attack* (EUF-CMVA). This definition comes with two independent winning conditions and guarantees, (1) classic unforgeability if the adversary only knows the user's password but none of the keys (neither of server nor user) and (2) online unforgeability if the user's key got compromised.

Thus, this can be seen as a combined version of the strong and online unforgeability we described in the introduction, with one significant limitation though: the ZWY model does not allow for server compromise in the strong unforgeability game, thus we refer to their version as *weak unforgeability.* In fact, we show that strong unforgeability is impossible in their setting.

Furthermore, their work does not capture offline unforgeability, again due to the absence of server compromise, and we show that this is also impossible in their setting. Note that the pw-hiding property is not needed in skPBC, as the server does not maintain user-specific state which could depend on the password.

For consistency and ease of presentation, we split the EUF-CMVA game along the two independent winning conditions which correspond to weak and online unforgeability. In the following we only focus on the weak unforgeability and the impossibility of strong unforgeability. In Appendix A.1 we state the online unforgeability of the ZWY model which will be used in our comparison of the single-key and multi-key frameworks.

Weak Unforgeability. Weak unforgeability guarantees that the adversary cannot forge a valid authentication token for a user if he does not know the user's credential ask. This provides standard security for users whose credential have not been compromised.

This property is modelled as a game played between a challenger and an adversary. The adversary chooses the usernames of all users. The challenger registers them with randomly chosen passwords with the honest server. The adversary is given the passwords of all users and can then ask arbitrary honest users to sign messages of his choice (via $\mathcal{O}_{\mathsf{Sign}}$) and ask the server to verify tokens of his choice (via $\mathcal{O}_{\mathsf{Vf}}$, recall that this is necessary as verify is a secret-key operation). He can also corrupt users via the $\mathcal{O}_{\mathsf{RevCred}}$ oracle, which returns the credential ask_i of a user i of his choice. The adversary wins if he can forge an authentication token on a fresh message for a user whose credential he has

Oracle $\mathcal{O}_{\mathsf{Vf}}(i, m, \tau)$	**Oracle** $\mathcal{O}_{\mathsf{Sign}}(i, m)$	**Oracle** $\mathcal{O}_{\mathsf{RevCred}}(i)$
Return $\mathsf{Vf}(ssk, uid_i, m, \tau)$	$\tau \leftarrow \mathsf{Sign}(uid_i, ask_i, pw_i, m)$ Add (i, m) to Q, **Return** τ	Add i to $\mathsf{RevCred}$ **Return** ask_i

Experiment $\mathsf{Exp}^{x\mathsf{UNF}}_{\mathcal{A},\mathsf{skPBC}}(\lambda)$ for $x \in \{\mathsf{weak}, \mathsf{strong}\}$

$(ssk, spk) \leftarrow \mathsf{KGen}(1^\lambda)$, $\mathsf{RevCred} \leftarrow \emptyset$, $Q \leftarrow \emptyset$
$(uid_1, \ldots, uid_n, st) \leftarrow \mathcal{A}(spk)$, for $i = 1, \ldots n : pw_i \xleftarrow{r} \mathcal{D}_{\mathsf{pw}}$
$(ask_i; -) \leftarrow \langle \mathsf{RegU}(spk, uid_i, pw_i), \mathsf{RegS}(ssk, uid_i) \rangle$
$(uid_j, m^*, \tau^*) \leftarrow \mathcal{A}^{\mathcal{O}_{\mathsf{RevCred}}, \mathcal{O}_{\mathsf{Sign}}, \mathcal{O}_{\mathsf{Vf}}}(st, pw_1, \ldots, pw_n, ssk)$
Return 1 if $\mathsf{Vf}(ssk, uid_j, m^*, \tau^*) = 1 \wedge j \in [n] \wedge j \notin \mathsf{RevCred} \wedge (j, m^*) \notin Q$

Fig. 1: Weak and Strong (in blue) Unforgeability for skPBC. The oracles use the values (i, uid_i, ask_i, pw_i) established during registration.

not obtained. The security experiment $\mathsf{Exp}^{\mathsf{weakUNF}}_{\mathcal{A},\mathsf{skPBC}}(\lambda)$ is given in Fig. 1 and the security definition is as follows:

Definition 3 (skPBC Weak/Strong Unforgeability). *A skPBC scheme is x-unforgeable, for $x \in \{weakly, strongly\}$, if for all PPT adversaries $\mathcal{A}$, it holds that* $\Pr[\mathsf{Exp}^{x\mathsf{UNF}}_{\mathcal{A},\mathsf{skPBC}}(\lambda) = 1] \leq \mathsf{negl}(\lambda)$.

2.3 Impossibility of Strong (and Offline) Unforgeability

Lifting the security definition from weak to strong unforgeability is straightforward: to model server compromise, we give the adversary access to the server's secret state – here ssk – after registering honest users. We require the same unforgeability for users whose individual credentials he never learned (see Fig. 1).

However, we now show that achieving this notion is impossible in the single-key setting. The idea of the attack is simple: once the adversary has learned the server's secret key, he re-runs the registration of an arbitrary honest user with a password of his choice to obtain a valid user credential and creates tokens in her name. More precisely, the following adversary $\mathcal{A}$ wins the strong unforgeability game for skPBC with probability 1:

$\underline{\mathcal{A}(spk)}$
Pick $uid_1 \xleftarrow{r} \mathcal{D}_{\mathsf{uid}}$, send (uid_1, st) to the challenger, receive (st, pw_1, ssk);
Pick $pw' \xleftarrow{r} \mathcal{D}_{\mathsf{pw}}$ and run $(ask'; -) \leftarrow \langle \mathsf{RegU}(spk, uid_1, pw'), \mathsf{RegS}(ssk, uid_1) \rangle$
Choose $m^* \xleftarrow{r} \mathcal{M}$; compute $\tau^* \leftarrow \mathsf{Sign}(uid_1, ask', pw', m^*)$ and **output** (uid_1, m^*, τ^*)

Success Analysis of $\mathcal{A}$: By the correctness definition it holds that $\mathsf{Vf}(ssk, uid_1, m^*, \mathsf{Sign}(uid_1, ask', pw', m^*)) = 1$ since ask' is obtained by running the registration protocol with (uid_1, pw') and the correct issuer secret key ssk. The adversary did neither query $\mathcal{O}_{\mathsf{Sign}}(1, m^*)$ nor $\mathcal{O}_{\mathsf{RevCred}}(1)$, and thus wins the security experiment $\mathsf{Exp}^{\mathsf{strongUNF}}_{\mathcal{A},\mathsf{skPBC}}(\lambda)$ with probability 1.

The attack exploits the fact that a single key ssk is used to both register users and verify their tokens, and never gets updated when a user registers. Hence, the authentication cannot depend on any user-provided input, but solely on the server key (and the secrecy thereof). This attack also extends to the context of offline unforgeability since an adversary who knows ssk can forge authentication tokens for any user without offline dictionary attacks.

Another Weakness: No Registration Oracle. We note that the ZWY security model [27] has another weakness: it does not allow corrupt users to register, which allows to prove entirely insecure schemes secure (e.g. the server sends his secret key ssk to the user during registration). We stress that this is primarily an oversight in the security model, and can be easily fixed by granting the adversary such registration access. We do not see any issue in the concrete skPBC scheme proposed in [27] and conjecture that it can be proven secure in this adjusted security model.

3 Multi-key Password-Based Credentials

Motivated by the impossibility of strong unforgeability in the single-key setting, we now introduce our concept of multi-key password-based credentials. The crucial difference is that the server no longer has a single secret key to issue user credentials and verify their tokens. Instead, he generates a user-specific verification key for each registered user and uses that user-specific key when verifying a user's token. We modify the original PBC syntax to the multi-key setting and then formalize the desired security properties.

Syntax. While the overall idea and concept remain the same in the multi-key setting, we change how the server stores user-specific verification information. We do not assume that the server has a single key pair (ssk, spk). Instead, in the registration phase, the server will output a user-specific verification key avk which allows him to verify the user's authentication token.

Definition 4 (Multi-key Password-based Credential). *A multi-key PBC scheme* $\mathsf{mkPBC} = (\mathsf{Setup}, \langle \mathsf{RegU}, \mathsf{RegS} \rangle, \mathsf{Sign}, \mathsf{Vf})$ *with message space* $\mathcal{M}$*, username space* $\mathcal{D}_{\mathsf{uid}}$ *and password space* $\mathcal{D}_{\mathsf{pw}}$ *is defined as follows.*

$\mathsf{Setup}(1^\lambda) \rightarrow pp$*: Outputs public parameters* pp*. We assume all algorithms get the public parameters* pp *as implicit input.*

$\langle \mathsf{RegU}(uid, pw), \mathsf{RegS}(uid) \rangle \rightarrow (ask; avk)$*: An interactive protocol between* $\mathcal{U}$ *with* $(uid, pw) \in \mathcal{D}_{\mathsf{uid}} \times \mathcal{D}_{\mathsf{pw}}$ *and* $\mathcal{S}$*. After successful registration, the user outputs a credential* ask*, and the server outputs a user-specific verification key* avk*.*

$\mathsf{Sign}(uid, ask, pw, m) \rightarrow \tau$*: Generates an authentication token* τ *on message* $m \in \mathcal{M}$ *and username* uid*, using* ask *and* pw*.*

$\mathsf{Vf}(uid, avk, m, \tau) \rightarrow 0/1$*: Outputs 1 if authentication token* τ *is valid on* uid *and* m *under* avk *and 0 otherwise.*

We require all honestly generated authentication tokens using the correct combination of ask and pw to pass validation under the corresponding avk.

Definition 5 (Correctness of mkPBC). *A* mkPBC *scheme is correct, if for all* $pp \leftarrow \mathsf{Setup}(1^\lambda)$*,* $(uid, pw) \in \mathcal{D}_{\mathsf{uid}} \times \mathcal{D}_{\mathsf{pw}}$*,* $m \in \mathcal{M}$ *it holds that:* $\mathsf{Vf}(uid, avk, m, \mathsf{Sign}(uid, ask, pw, m)) = 1$ *where* $(ask; avk) \leftarrow \langle \mathsf{RegU}(uid, pw), \mathsf{RegS}(uid) \rangle$*.*

3.1 Security Model

We now provide a formal model for the following security properties motivated in Section 1.1 and partially inspired by the ZWY model [27]. The detailed security experiments are given in Fig. 2 and explained below.

- **Strong Unforgeability:** An adversary who does not know a user's ask cannot forge an authentication token for that user, even when he knows the user's password pw and the server's verification key avk.
- **Online Unforgeability:** An adversary who knows ask but not pw or avk cannot forge an authentication token more efficiently than through online guessing attacks, interacting with the server who has avk.
- **Offline Unforgeability:** If the adversary knows both ask and avk of a user, he has to conduct a brute-force offline dictionary attack on the password pw in order to forge an authentication token.
- **Pw-Hiding:** The avk does not leak any information about the underlying pw.

Optimal Security. We stress that all security guarantees are optimal for the respective corruption setting, i.e., achieve the strongest level of full/online/offline attack-resistance for each combination of corrupted keys and passwords. When defining these properties through formal security models, it is important to give the adversary therein as much "access" to honest parties as possible. In fact, this was not properly captured in the ZWY model: therein corrupt users where not allowed to register with an honest server, which allows entirely insecure schemes to be proven secure. Interestingly, our choice of letting the server maintain *independent* key material for all users, simplifies the modelling significantly: since the server in mkPBC does not have any long-term secret key used during registration or verification, the adversary can internally simulate the registration of any corrupt user (expressed through any combination of uid and pw) that he wants. Thus, for our security model, it suffices to consider only a single honest target user and let the adversary (internally) handle all other (corrupt) users in the system.

Strong Unforgeability. Without knowing the user's credential ask, we want the strongest security in the sense that an adversary can forge tokens in the name of an honest user uid with *negligible* probability only. This is modelled by letting the challenger run the registration for an honest user uid with password pw, obtaining ask and avk. It then hands avk, pw to the adversary, and grants $\mathcal{A}$ access to a sign oracle $\mathcal{O}_{\mathsf{Sign}}$, which returns tokens created with ask (and pw) for messages m_i of his choice. The adversary wins if he can produce a valid token τ^* for a fresh message m^* that verifies for the honest users uid and avk.

$\mathsf{Exp}^{\mathsf{xUNF}}_{\mathcal{A},\mathsf{mkPBC}}(\lambda)$ for $\mathsf{x} \in \{\mathsf{strong}, \mathsf{online}, \mathsf{offline}\}$

$pp \leftarrow \mathsf{Setup}(1^\lambda),\ \ Q \leftarrow \emptyset$

$(uid, st) \leftarrow \mathcal{A}(pp),\ \ pw \xleftarrow{r} \mathcal{D}_{\mathsf{pw}}$

$(ask; avk) \leftarrow \langle \mathsf{RegU}(uid, pw), \mathsf{RegS}(uid) \rangle$

$(m^*, \tau^*) \leftarrow \mathcal{A}^{\mathcal{O}}(\mathsf{keys}, st)$

Return 1 if $\mathsf{Vf}(uid, avk, m^*, \tau^*) = 1 \land m^* \notin Q$

Oracles init. with (uid, ask, avk, pw)

$\mathcal{O}_{\mathsf{Sign}}(m_i)$: $\tau_i \leftarrow \mathsf{Sign}(uid, ask, pw, m_i)$

$Q \leftarrow Q \cup m_i$, **Return** τ_i

$\mathcal{O}_{\mathsf{Vf}}(m_i, \tau_i)$: **Return** $\mathsf{Vf}(uid, avk, m_i, \tau_i)$

$\mathcal{O}_{\mathsf{TestPW}}(pw_i)$: **Return** $pw = pw_i$

With keys and $\mathcal{O}$ defined as:

x	keys	Oracles $\mathcal{O}$
strong	avk, pw	$\mathcal{O}_{\mathsf{Sign}}$
online	ask	$\mathcal{O}_{\mathsf{Sign}}, \mathcal{O}_{\mathsf{Vf}}$
offline	ask, avk	$\mathcal{O}_{\mathsf{Sign}}, \mathcal{O}_{\mathsf{TestPW}}$

$\mathsf{Exp}^{\mathsf{PW-Hiding}}_{\mathcal{A},\mathsf{mkPBC}}(\lambda)$

$(uid, pw_0, pw_1, st) \leftarrow \mathcal{A}(pp)$; $b \xleftarrow{r} \{0, 1\}$

$(ask; avk) \leftarrow \langle \mathsf{RegU}(uid, pw_b), \mathsf{RegS}(uid) \rangle$

$b' \leftarrow \mathcal{A}^{\mathcal{O}_{\mathsf{Sign}}}(avk, st)$ with $\mathcal{O}_{\mathsf{Sign}}$ using $pw := pw_b$

Return 1 if $b = b'$

Fig. 2: Security experiments and oracles for mkPBC. The overall goal of the adversary in our three unforgeability games is the same, and is shown in the combined xUNF experiment, where only the set of revealed keys and oracles differ depending on x.

Definition 6 (Strong Unforgeability). *A* mkPBC *scheme is strongly unforgeable, if for all PPT adversaries* $\mathcal{A}$*:* $\Pr[\mathsf{Exp}^{\mathsf{strongUNF}}_{\mathcal{A},\mathsf{mkPBC}}(\lambda) = 1] \leq \mathsf{negl}(\lambda)$.

Online Unforgeability. If the adversary knows the user's high-entropy credential ask it is impossible to achieve strong unforgeability anymore. As soon as $\mathcal{A}$ has correctly guessed the user's password, there is no security. The best we can hope for is security against online attacks, relying on the server's user-specific verification key avk as a second defense, i.e. the honest server's participation must be required to verify each of $\mathcal{A}$'s password guesses.

In the security game, this is modelled by giving $\mathcal{A}$ the credential ask of the honestly registered user uid, but neither avk nor pw. Consequently, we grant $\mathcal{A}$ access to avk through a verify oracle $\mathcal{O}_{\mathsf{Vf}}$ that allows the adversary to verify message-token pairs (m_i, σ_i) of his choice under the server's avk. Given that the adversary knows ask, he can use $\mathcal{O}_{\mathsf{Vf}}$ as a password test oracle, submitting tokens generated for the correct ask and different password guesses pw'.

It might look surprising that we grant $\mathcal{A}$ access to a sign oracle too – as he does know ask here – but this oracle is necessary since he does not know the corresponding pw and must be able to observe valid tokens by the honest user.

The adversary's goal is still to forge an authentication token for the honest user. The security definition needs to be weakened to online attacks though, and states that the adversary cannot win the experiment significantly better than through testing $q_{\mathsf{Vf}} + 1$ of the $|\mathcal{D}_{\mathsf{pw}}|$ passwords where q_{Vf} is the number of queries made to the $\mathcal{O}_{\mathsf{Vf}}$ oracle. The additional constant 1 is added because the forgery which $\mathcal{A}$ outputs in the end, can itself be seen as a password guess.

Definition 7 (Online Unforgeability). *A* mkPBC *scheme is online unforgeable, if for all PPT adversaries $\mathcal{A}$ it holds that* $\Pr[\mathsf{Exp}^{\mathsf{onlineUNF}}_{\mathcal{A},\mathsf{mkPBC}}(\lambda) = 1] \leq \frac{q_{\mathsf{Vf}}+1}{|\mathcal{D}_{\mathsf{pw}}|} + \mathsf{negl}(\lambda)$, *where* q_{Vf} *is the number of queries to the* $\mathcal{O}_{\mathsf{Vf}}$ *oracle.*

Offline Unforgeability. If both keys, ask and avk, related to an honest user are compromised, the unforgeability solely relies on the strength of the user password pw. The best we can hope for in this setting are offline attacks: the adversary can test passwords by signing a message using the corrupted ask and password guess pw' and verify the resulting token using the key avk. As soon as the adversary has correctly guessed pw, there is no secret left, and he can create tokens for arbitrary messages. Offline attacks are unavoidable in this case, but we also want them to be the best possible attack. This means that choosing a strong password adds an additional (albeit weak) layer of security for the user.

To quantify the offline amount of work the adversary has to perform, we took inspiration from security models of other password-based protocols [8,9,10] and introduce an oracle $\mathcal{O}_{\mathsf{TestPW}}$ which takes the adversaries password guess pw' and returns 1 if $pw = pw'$ and 0 else. The adversary's goal stays the same – forging an authentication token for the honest user – which he must not be able to do significantly better than through testing q_f of the $|\mathcal{D}_{\mathsf{pw}}|$ passwords where q_f is the number of queries made to the $\mathcal{O}_{\mathsf{TestPW}}$ oracle.

Note that proving a concrete scheme to satisfy this property inherently requires some idealized assumption such as the random oracle, which needs to get invoked on the user's password – otherwise we could simply not count the offline password guesses.

Definition 8 (Offline Unforgeability). *A* mkPBC *scheme is offline unforgeable, if for all PPT adversaries $\mathcal{A}$ it holds that* $\Pr[\mathsf{Exp}^{\mathsf{offlineUNF}}_{\mathcal{A},\mathsf{mkPBC}}(\lambda) = 1] \leq \frac{q_f}{|\mathcal{D}_{\mathsf{pw}}|} + \mathsf{negl}(\lambda)$, *where* q_f *is the number of queries to the oracle* $\mathcal{O}_{\mathsf{TestPW}}$.

PW-Hiding. This property guarantees that a malicious server learns nothing about the user's password, or rather that a user-specific key avk – despite being derived from a user password pw – does not leak any information about pw.

To model this property, we follow the classic indistinguishability approach. The adversary chooses two passwords pw_0 and pw_1 for a user uid. The challenger randomly chooses a bit b and runs the registration protocol for user uid and pw_b, yielding ask and avk. It hands avk to the adversary, whose goal is to output the correct bit b better than through guessing. To model any possible leakage through other parts of the PBC system, we also grant the adversary access to an $\mathcal{O}_{\mathsf{Sign}}$ oracle which is keyed with ask and pw_b.

Definition 9 (Pw-Hiding). *A* mkPBC *scheme is pw-hiding, if for all PPT adversaries $\mathcal{A}$ it holds that* $\Pr[\mathsf{Exp}^{\mathsf{PW-Hiding}}_{\mathcal{A},\mathsf{mkPBC}}(\lambda) = 1] \leq 1/2 + \mathsf{negl}(\lambda)$.

3.2 Comparison of mkPBC and skPBC

An imminent question that arises is how multi-key PBCs and single-key PBCs are related. Given the straightforwardness of our construction for multi-key PBC in Section 4.3 compared to the ZWY version, it is natural to question if the multi-key setting somewhat compromises the overall security guarantees. We show that the opposite is true by showcasing how a secure multi-key PBC can be converted into a secure single-key PBC scheme. Our transformation additionally requires the use of symmetric authenticated encryption (AE), and thus should be viewed as a relativized comparison.

The high-level idea of the transformation is as follows: In order to transform the mkPBC to have only one key, the server outsources storage of the user-specific verification keys avk to the users. In the transformation, the server in the skPBC scheme has a single long-term key ssk which is the secret key k_{AE} of an AE scheme. In the registration phase, the server and user run the mkPBC registration, but instead of letting the server store the obtained avk it returns its encryption $c \leftarrow \mathsf{AE.Enc}(k_{\mathsf{AE}}, (uid, avk))$ to the user. During authentication, the user passes c back to the server by appending it to the authentication token τ which is computed via the mkPBC process. The server can decrypt c to obtain the verification key avk and verify the user's token. For the security of the scheme, it is crucial that the user does not learn avk from c otherwise she could run offline attacks. Furthermore, it is important that users cannot pass the valid ciphertext of a different verification key avk' to the server as this would allow forgeries. Both, confidentiality and integrity, is achieved by using a secure *authenticated encryption* scheme.

In Appendix B, we formally describe this transformation and show that it yields an online and weakly unforgeable skPBC, if the multi-key PBC is online and strongly unforgeable and the AE is a secure authenticated encryption scheme. This result is not only of theoretical interest since in some applications it might be more suitable to use a single-key solution. One reason could be that securely storing a single key might be easier than securing potentially many keys for all users. In this case, our transformation using the $\mathsf{PBC}_{\mathsf{StE}}$ presented in Section 4.3 allows to construct a skPBC built on standard building blocks.

4 Our Instantiation: Sign-Then-Encrypt Based Scheme

In this section, we describe $\mathsf{PBC}_{\mathsf{StE}}$ which securely realizes all security guarantees described in Section 3. Our scheme is conceptually entirely different from the one proposed by Zhang et al. [27], which essentially relied on a DL-based algebraic MAC. Our scheme is generic and soley relies on basic building blocks: a signature scheme, encryption scheme and a pseudorandom function. In order to prove security, we need two less common properties from the signature scheme in addition to unforgeability: *complete robustness* and *randomness injectivity.* We stress that both are natural assumptions and argue that they are satisfied by standard signatures schemes such as Schnorr, DSA and BLS. We start by defining the main building blocks and their required security properties before describing our provably secure construction.

4.1 Building Blocks

We now introduce the building blocks needed for our construction, focusing on the lesser known properties that we will require from the signature scheme.

Notation. Since our construction depends on a signature scheme with deterministic key generation algorithm using explicit randomness, we write "$y := A(x; r)$" to highlight that the output y is derived deterministically by algorithm A on input x with randomness r. Conversely, when we write "$y \leftarrow A(x)$", the output y may be derived either deterministically or probabilistically by algorithm A from input x. We utilize "$s \xleftarrow{r} S$" to denote the uniformly random sampling of a value s from the set S.

Pseudorandom Function. We require a secure PRF $F : \{0,1\}^\lambda \times \mathcal{X} \rightarrow \mathcal{Y}$. In some of our security experiments, the adversary will be in possession of the PRF key, and we still want unpredictability of outputs – we then resort to assuming F to be a random oracle for the combined input domain of $\{0,1\}^\lambda \times \mathcal{X}$.

Public-key Encryption. A public-key encryption (PKE) scheme $\Pi_{Enc} := (\mathsf{KGen}_E, \mathsf{Enc}, \mathsf{Dec})$ consisting of key generation $(pk_{\mathsf{Enc}}, sk_{\mathsf{Enc}}) \leftarrow \mathsf{KGen}_E(1^\lambda)$, an encryption $c \leftarrow \mathsf{Enc}(pk_{\mathsf{Enc}}, m)$ and decryption algorithm $m \leftarrow \mathsf{Dec}(sk_{\mathsf{Enc}}, c)$. We require Π_{Enc} to be indistinguishable against chosen-ciphertext attacks (IND-CCA).

Signature Scheme. A signature scheme $\Pi_{Sign} := (\mathsf{Setup}_S, \mathsf{KGen}_S, \mathsf{Sign}_S, \mathsf{Vf}_S)$ with setup $pp \leftarrow \mathsf{Setup}(1^\lambda)$, key generation $(pk_{\mathsf{Sig}}, sk_{\mathsf{Sig}}) := \mathsf{KGen}(pp; r)$ for randomness r, sign algorithm $\sigma \leftarrow \mathsf{Sign}_S(sk_{\mathsf{Sig}}, m)$, and verify algorithm $b \leftarrow \mathsf{Vf}_S(pk_{\mathsf{Sig}}, m, \sigma)$. Note that we make the randomness used in key generation explicit and assume KGen to be a deterministic function when given randomness $r \in \mathcal{R}_\lambda$ as input. $\mathcal{R}_\lambda$ is part of the public parameters pp and denotes the randomness space. We require the scheme to be existentially unforgeable under chosen-message attacks (EUF-CMA): It must be infeasible for an adversary given pk_{Sig} from $(sk_{\mathsf{Sig}}, pk_{\mathsf{Sig}}) := \mathsf{KGen}(pp; r)$ for random $r \xleftarrow{r} \mathcal{R}_\lambda$ and access to a sign oracle to produce a valid signature on a fresh message. Our construction requires two additional properties: *complete robustness* and *randomness injectivity.*

Complete Robustness. Géraud and Naccache [13] formalized the notion of complete robustness which requires that it should be hard for an adversary to find a message-signature-pair which verifies under two different public keys.

Definition 10 (Complete Robustness). *A signature scheme* $\Pi_{Sign} := (\mathsf{Setup}, \mathsf{KGen}, \mathsf{Sign}, \mathsf{Vf})$ *achieves* complete robustness (CROB) *or is* CROB-secure *if for* $pp \leftarrow \mathsf{Setup}(1^\lambda)$ *it holds that for every PPT* $\mathcal{A}$*, the probability* $\Pr[(pk, pk', m, \sigma) \leftarrow \mathcal{A}(pp) : pk \neq pk' \wedge \mathsf{Vf}(pk, m, \sigma) = \mathsf{Vf}(pk', m, \sigma) = 1]$ *is negligible in* λ*.*

Randomness Injectivity. The second property we need is randomness injectivity which requires that the KGen algorithm is injective on the randomness space. We call a signature scheme randomness injective if it is hard for an adversary to find two distinct values $r, r' \in \mathcal{R}$, which, when given to KGen, map to the same sk or pk. This also implies that for every public key there exists only one secret key. More formally, randomness injectivity can be defined as follows:

Definition 11 (Randomness Injectivity). *A signature scheme Π_{Sign} := $(\mathsf{Setup}, \mathsf{KGen}, \mathsf{Sign}, \mathsf{Vf})$ is* randomness injective *if for $pp \leftarrow \mathsf{Setup}(1^\lambda)$ with $\mathcal{R}_\lambda \in pp$, it holds that for every PPT $\mathcal{A}$, the following probability is negligible in λ:*

$$\Pr[(r, r') \leftarrow \mathcal{A}(pp) : r, r' \in \mathcal{R}_\lambda \wedge r \neq r' \wedge (sk = sk' \vee pk = pk') \\ \textit{for } (pk, sk) := \mathsf{KGen}(pp; r), (pk', sk') := \mathsf{KGen}(pp; r')]$$

4.2 Our $\mathsf{PBC}_{\mathsf{StE}}$ Protocol

The idea of our protocol – referred to as $\mathsf{PBC}_{\mathsf{StE}}$ – is surprisingly simple and turns classic signature-based authentication into a secure mkPBC. In the following, we describe the intuition and give the full description in Fig. 3.

Upon registration, the user generates a signature key pair $(pk_{\mathsf{Sig}}, sk_{\mathsf{Sig}})$ and sends the public key pk_{Sig} to the server. Such a key pair enables strong authentication through signing (uid, m), but all security will be lost when an attacker gets access to the user's signing key. We therefore do not store (or even generate) the key normally, but derive it *deterministically* as $(pk_{\mathsf{Sig}}, sk_{\mathsf{Sig}}) := \mathsf{KGen}_S(pp; F(k, pw))$ from a PRF key k and the user's password pw. The user now only stores the PRF key k and re-derives the signature key pair when she wants to generate an authentication token.

This solution already satisfies strong and offline unforgeability as well as password hiding. The challenge is to also guarantee online unforgeability, i.e., ensuring that the knowledge of the user's key and an authentication token does not allow to brute-force the password. So far, this isn't achieved as an attacker who knows k and a valid signature σ can mount an offline password test by computing possible key-pairs $(pk'_{\mathsf{Sig}}, sk'_{\mathsf{Sig}})$ from password guesses pw' until he has found the correct pw' under which σ verifies.

Preventing Offline Attacks. We prevent this offline attack by hiding the actual signature σ in the token. Therefore, we let the user encrypt σ under an encryption public-key pk_{Enc} to which the server knows the corresponding secret key sk_{Enc}. More precisely, $(pk_{\mathsf{Enc}}, sk_{\mathsf{Enc}})$ is a key pair that the user normally generated upon registration, where she keeps pk_{Enc} as part of her credential, i.e., $ask = (k, pk_{\mathsf{Enc}})$ and sends the secret decryption key to the server, i.e., $avk = (sk_{\mathsf{Enc}}, pk_{\mathsf{Sig}})$. Now, only the server knowing sk_{Enc} can recover the signature from the authentication token and verify its validity. Thus, this encryption finally turns the verification into a secret-key operation, which is essential for the desired online unforgeability. We stress that this additional and explicit encryption layer is essential for our security and cannot be achieved from assuming secure channels between the user and server: honest user's can be subject to phishing attacks, and accidentally send authentication tokens to a malicious server.

The Challenge of Proving Online Unforgeability. While the additional encryption immediately removes the obvious offline attack, proving that this is sufficient to achieve online unforgeability is not straightforward.

The challenge is that the adversary knows the PRF key k and can offline attack the password and thereby recover the secret signing key $(pk_{\mathsf{Sig}}, sk_{\mathsf{Sig}}) := \mathsf{KGen}_S(pp; F(k, pw))$. Once he knows the correct secret key there is no security left. And indeed, we cannot rely on any unforgeability guarantees of the signature for this proof. The reason why our scheme is still secure stems from the fact that the adversary does not know which key is the *correct* one: he does not know pk_{Sig} (this is part of the server's secret key) nor any signature value (they are encrypted under the server's key). The only way for $\mathcal{A}$ to learn whether a recovered key is correct, is to compute a signature and send it for validation to the server. The crucial part in our proof is to show that every interaction with the honest server for such a verification is bound to a single password guess only, ensuring the desired online unforgeability.

To illustrate how the signature scheme could allow multiple password tests in one interaction, consider a signature σ on m which verifies under two different public keys pk_1 and pk_2 constructed from passwords pw_1 and pw_2. If the adversary sends (m, σ) to the server and learns that the signature is not valid, he concludes that the server's public key is neither pk_1 nor pk_2 and has ruled out the two passwords pw_1 and pw_2 with one interaction. Hence, we require that every signature verifies under at most one public key which is achieved through complete robustness. Another way how the signature scheme could allow multiple password tests is if the public key pk_1 can be constructed from multiple passwords pw_1 and pw_2. Therefore, we require that every password maps to a unique secret key and unique public key. This is achieved if F is injective, and if the signature scheme has randomness injectivity.

4.3 Security Analysis

In this section, we provide the main security theorems for our $\mathsf{PBC}_{\mathsf{StE}}$ scheme and sketch their proofs. The detailed proofs are given in Appendix D.

Theorem 1. *If F is a secure PRF and Π_{Sign} is an EUF-CMA secure signature scheme, then $\mathsf{PBC}_{\mathsf{StE}}$ is strongly unforgeable.*

Proof (Sketch). In the strong unforgeability game, the adversary knows the verification key $avk = (pk_{\mathsf{Sig}}, sk_{\mathsf{Enc}})$ and password pw of a user uid, but not the user credential $ask = (k, pk_{\mathsf{Enc}})$. He does have access to a sign oracle $\mathcal{O}_{\mathsf{Sign}}$ that creates tokens for ask, and $\mathcal{A}$ wins if he can create an authentication token τ^* which verifies under avk on a fresh message m^*. In this proof, we can ignore the encryption, as the adversary knows sk_{Enc}, i.e., for all tokens returned by $\mathcal{O}_{\mathsf{Sign}}$, he can recover the contained signature derived from $(pk_{\mathsf{Sig}}, sk_{\mathsf{Sig}}) := \mathsf{KGen}_S(pp; F(k, pw))$. Thus, the task of the adversary boils down to forging a standard signature under the unknown sk_{Sig}. This is infeasible if the signature scheme is unforgeable (EUF-CMA) under the assumption that the PRF-derived secret key is indistinguishable from a randomly chosen one. The latter follows from the pseudorandomness of F which concludes our proof.

Let $\Pi_{Sign} := (\mathsf{Setup}_S, \mathsf{KGen}_S, \mathsf{Sign}_S, \mathsf{Vf}_S)$ be a signature scheme with explicit randomness for randomness space $\mathcal{R}$, $F : \{0,1\}^\lambda \times \mathcal{D}_{\mathsf{pw}} \to \mathcal{R}$ be a pseudorandom function, and $\Pi_{Enc} := (\mathsf{KGen}_E, \mathsf{Enc}, \mathsf{Dec})$ be a public-key encryption scheme.

$\underline{\mathsf{Setup}(1^\lambda)}$: Output $pp \leftarrow \mathsf{Setup}_S(1^\lambda)$.

$\underline{\mathsf{RegU}(uid, pw) \rightleftharpoons \mathsf{RegS}(uid)}$:
User $\mathcal{U}$: $k \xleftarrow{r} \{0,1\}^\lambda$, $(pk_{\mathsf{Sig}}, sk_{\mathsf{Sig}}) := \mathsf{KGen}_S(pp; F(k, pw))$, $(pk_{\mathsf{Enc}}, sk_{\mathsf{Enc}}) \leftarrow \mathsf{KGen}_E(1^\lambda)$. She sends $(uid, avk := (pk_{\mathsf{Sig}}, sk_{\mathsf{Enc}}))$ to the server and outputs $ask := (k, pk_{\mathsf{Enc}})$.

Server $\mathcal{S}$: upon receiving (uid, avk) outputs avk (we assume that (uid, avk) are stored together on the application level).

$\underline{\mathsf{Sign}(uid, ask, pw, m)}$:
Parse $ask := (k, pk_{\mathsf{Enc}})$. Compute $(pk_{\mathsf{Sig}}, sk_{\mathsf{Sig}}) := \mathsf{KGen}_S(pp; F(k, pw))$, $\sigma \leftarrow \mathsf{Sign}_S(sk_{\mathsf{Sig}}, (uid, m))$, $\tau \leftarrow \mathsf{Enc}(pk_{\mathsf{Enc}}, \sigma)$ and output τ.

$\underline{\mathsf{Vf}(uid, avk, m, \tau)}$:
Parse $avk := (pk_{\mathsf{Sig}}, sk_{\mathsf{Enc}})$. Compute $\sigma \leftarrow \mathsf{Dec}(sk_{\mathsf{Enc}}, \tau)$ and output $\mathsf{Vf}_S(pk_{\mathsf{Sig}}, (uid, m), \sigma)$

Fig. 3: Our $\mathsf{PBC}_{\mathsf{StE}}$ scheme.

Theorem 2. *If F is a random oracle, Π_{Sign} is completely robust and randomness injective, and Π_{Enc} is CCA-secure, then $\mathsf{PBC}_{\mathsf{StE}}$ is online unforgeable.*

Proof (Sketch). Here, the adversary knows the high-entropy credential $ask = (k, pk_{\mathsf{Enc}})$ of a user uid, but neither her password pw nor the corresponding verification key $avk = (pk_{\mathsf{Sig}}, sk_{\mathsf{Enc}})$. Both are accessible through the $\mathcal{O}_{\mathsf{Sign}}$ and $\mathcal{O}_{\mathsf{Vf}}$ oracle though. We must show that if $\mathcal{A}$ outputs a valid token τ^* for a fresh message m^* for uid, he must have conducted a successful online-attack on the password. In this proof, we first show that, due to the CCA-security of encryption, the $\mathcal{O}_{\mathsf{Sign}}$ oracle does not give the adversary any information about the underlying signature. Then, we argue that the adversary knows the PRF key k and may offline guess passwords to create possible key pairs $(pk'_{\mathsf{Sig}}, sk'_{\mathsf{Sig}}) := \mathsf{KGen}_S(pp; F(k, pw'))$ and forge signatures under sk'_{Sig}. However, in order to create a valid authentication token, he needs to use the *correct* secret key sk_{Sig}. As the correct pk_{Sig} is part of the secret avk and $\mathcal{A}$ never sees any signature σ_i, his only chance of learning which key is correct, is by using the verify oracle $\mathcal{O}_{\mathsf{Vf}}$. Complete robustness of the signature ensures that every interaction with $\mathcal{O}_{\mathsf{Vf}}$ only leaks whether $pk_{\mathsf{Sig}} = pk'_{\mathsf{Sig}}$, allowing a single public-key guess per query. The injectivity of F and the randomness injectivity of Π_{Sign} ensure that this pk'_{Sig} maps to a single password guess, thus the adversary can only guess one password per interaction with $\mathcal{O}_{\mathsf{Vf}}$. This concludes our proof.

Theorem 3. *If F is a random oracle and if Π_{Sign} is EUF-CMA secure and randomness injective, then $\mathsf{PBC}_{\mathsf{StE}}$ is offline unforgeable.*

Proof (Sketch). In the offline unforgeability game, the adversary now knows all keys, i.e,. $ask = (k, pk_{\mathsf{Enc}})$ and $avk = (pk_{\mathsf{Sig}}, sk_{\mathsf{Enc}})$ of a user uid. The only secret left is her password pw, and we must show that forging a fresh token m^*, τ^* for uid requires to (at least) offline-attack the password. Note that the adversary is given k here, but not the actual signature key $(pk_{\mathsf{Sig}}, sk_{\mathsf{Sig}}) := \mathsf{KGen}_S(pp; F(k, pw))$, which still depends on the password. Thus the task of $\mathcal{A}$ again boils down to forging a valid signature under pk_{Sig}. He could either aim at forging the signature directly, i.e., without trying to recover the secret key, or brute-force the password to compute sk_{Sig}, as then creating a signature is trivial. The former is infeasible if the signature is unforgeable, and the latter is bounded by the number of password guesses if the signature is randomness injective (RI) and F a random oracle. RI guarantees that there is only one value $r = F(k, pw)$ such that $(pk_{\mathsf{Sig}}, sk_{\mathsf{Sig}}) = \mathsf{KGen}_S(pp; r)$, i.e., there is only a single password that leads to the correct key. Since the password pw was chosen uniformly at random from $\mathcal{D}_{\mathsf{pw}}$, the adversary needs to query the random oracle F for each password guess, and after q_F queries his success probability is bounded by $q_F/|\mathcal{D}_{\mathsf{pw}}|$.

Theorem 4. *If F is a secure PRF, then $\mathsf{PBC}_{\mathsf{StE}}$ is pw-hiding.*

Proof (Sketch). Recall that in the pw-hiding game the adversary receives a verification key $avk = (pk_{\mathsf{Sig}}, sk_{\mathsf{Enc}})$ that is either derived for pw_0 or pw_1, and his task is to determine the underlying password. In our scheme, the only password-dependent information is $(pk_{\mathsf{Sig}}, sk_{\mathsf{Sig}}) := \mathsf{KGen}_S(pp; F(k, pw_b))$. The adversary knows pk_{Sig}, but not the PRF key k, and has access to the key through the sign oracle for $ask = (k, pk_{\mathsf{Enc}})$. As k is chosen at random from $\{0,1\}^\lambda$, it immediately follows from the PRF property that the adversary cannot distinguish whether pk_{Sig} was created from $r = F(k, pw_b)$ or r chosen at random from $\mathcal{R}_\lambda$. Since in the latter case, the avk is independent of the password, the pw-hiding property follows. Note that we do not require any property from the signature scheme here, as the pw-hiding concerns confidentiality of the password instead of unforgeability as the other three properties.

Concrete Instantiation of Building Blocks. The requirements for both the PRF and PKE are standard, so we only focus on how randomness injectivity and complete robustness can be achieved by a signature scheme. For the signature scheme, we show that the DL-based standard signature schemes DSA [17], Schnorr [22] and BLS [6] all achieve both properties (apart from being (EUF-CMA) unforgeable).

Theorem 5. *The DSA, Schnorr and BLS signature scheme all achieve randomness injectivity information-theoretically. DSA and Schnorr are CROB-secure assuming a collision-resistant hash function, and BLS is information-theoretically CROB-secure.*

Proof. The proof of CROB-security is in Appendix C. For the randomness injectivity, observe that DL-based signature schemes where it holds that $pk = g^{sk}$ for $sk \xleftarrow{r} \mathbb{Z}_q$ achieve randomness injectivity by setting $\mathcal{R}_\lambda = \mathbb{Z}_q$ and $(g^r, r) := \mathsf{KGen}(pp; r)$.

Notable signature schemes which are not completely robust include RSA, GHR and Rabin signatures (see [18]). Nevertheless, Géraud and Naccache [13] show a generic method to transform any signature scheme into a completely robust scheme by appending a hash of the public key to the signature. This transformation also preserves the unforgeability property of the scheme.

5 Related Work

While our work builds upon the novel PBC work by Zhang et al. [27], we also put it in a bigger context of password-based authentication schemes.

Works without Online Unforgeability. Isler and Küpcü [15] give an overview of existing schemes where a user authenticates with the combination of a password and a password-based credential. As in PBCs, the password-based credential is not directly the user's secret key but only a password-protected version of it. They analyzed several existing works [2,5,16,14,4] and argued that most are not resistant against server-compromise and proposed a new scheme. The main drawback of all schemes (except DE-PAKE [16], discussed below) is that they do not achieve the same strong online unforgeability as [27] and our work. Roughly, when the password-based credential got compromised, their model only guarantees security when the adversary never sees any authentication token from the honest user, thus excluding phishing attacks from their model. Our work provides online unforgeability without assuming full secrecy of tokens.

DE-PAKE. Device Enhanced PAKE by Jarecki et al. [16] is a variant of password-authenticated key exchange where a user and a server derive a shared key based on the user's knowledge of a strong key (stored on an auxiliary device) and a password. Jarecki et al. show a generic solution which uses the Ford-Kaliski method [12] to strengthen her password into a strong key using a PRF and uses this strong key in an asymmetric PAKE protocol to derive a shared key with the server. Our work uses the same PRF-based method to strengthen a password into a key. Similarly to the work of Jarecki et al., we aim to achieve optimal protection against online and offline attacks, albeit in the context of pure user authentication instead of key exchange. Our $\mathsf{PBC}_{\mathsf{StE}}$ scheme uses simpler building blocks than the solution presented in [16]. As our scheme allows for non-interactive generation of challenge messages (e.g., by hashing the user id with a current timestamp), we can even achieve the optimal solution of user authentication with one message.

6 Conclusion

We revisited the existing framework of password-based credentials from Zhang et al. [27] and found that an important security property was missing – the resistance to server compromise. We showed that achieving this level of security is impossible in their single-key framework. While the attack is simple, it is of practical relevance considering that data breaches happen frequently. This motivated us to propose a new framework called multi-key password-based credentials which remain secure in the presence of server compromise. We established formal definitions for the optimal security levels and proposed a solution that utilizes generic building blocks and satisfies all desirable security properties.

References

1. 2012 Linkedin Breach had 117 Million Emails and Passwords Stolen, Not 6.5M (May 2016), https://www.trendmicro.com/vinfo/us/security/news/cyber-attacks/2012-linkedin-breach-117-million-emails-and-passwords-stolen-not-6-5m
2. Acar, T., Belenkiy, M., Küpçü, A.: Single Password Authentication. Computer Networks **57**, 2597–2614 (2013)
3. Bader, C., Jager, T., Li, Y., Schäge, S.: On the Impossibility of Tight Cryptographic Reductions. In: EUROCRYPT '16
4. Belenkiy, M., Acar, T., Jerez Morales, H.N., Küpcü, A.: Securing passwords against dictionary attacks, US Patent 9015489B2 (Oct 2011)
5. Bicakci, K., Atalay, N.B., Yuceel, M., Van Oorschot, P.C.: Exploration and Field Study of a Password Manager Using Icon-Based Passwords. In: FC '11
6. Boneh, D., Lynn, B., Shacham, H.: Short Signatures from the Weil Pairing. In: ASIACRYPT '01
7. Boneh, D., Shoup, V.: A graduate course in applied cryptography. Draft 0.6 (2023)
8. Boyen, X.: Hidden Credential Retrieval from a Reusable Password. In: ASIACCS '09
9. Camenisch, J., Lehmann, A., Neven, G.: Optimal Distributed Password Verification. In: CCS '15
10. Das, P., Hesse, J., Lehmann, A.: DPaSE: Distributed Password-Authenticated Symmetric Encryption. ASIACCS '22
11. Dobran, B.: 1.6 million PayPal customer details stolen in Major Data Breach (Jan 2022), https://phoenixnap.com/blog/paypal-customer-details-stolen
12. Ford, W., Kaliski, B.S.: Server-assisted Generation of a Strong Secret from a Password. In: WET ICE '00
13. Géraud, R., Naccache, D., Roşie, R.: Robust Encryption, Extended. CT-RSA '19
14. İşler, D., Küpçü, A.: Distributed single password protocol framework. Cryptology ePrint Archive, Report 2018/976 (2018), https://eprint.iacr.org/2018/976
15. İşler, D., Küpçü, A.: Threshold single password authentication. Cryptology ePrint Archive, Report 2018/977 (2018), https://eprint.iacr.org/2018/977
16. Jarecki, S., Krawczyk, H., Shirvanian, M., Saxena, N.: Device-Enhanced Password Protocols with Optimal Online-Offline Protection. In: ASIACCS '16
17. Kerry, C.F., Gallagher, P.D.: Digital Signature Standard (DSS). FIPS PUB pp. 186–4 (2013)
18. Koblitz, N., Menezes, A.: Another look at security definitions. Cryptology ePrint Archive, Report 2011/343 (2011), https://eprint.iacr.org/2011/343
19. Lindemann, R., Tiffany, E.: FIDO UAF Protocol Specification (2017)
20. Reynolds, J., Smith, T., Reese, K., Dickinson, L., Ruoti, S., Seamons, K.: A Tale of Two Studies: The Best and Worst of YubiKey Usability. In: S&P '18
21. Roman, J., Ross, R.: Blizzard Entertainment Reports Breach (Aug 2012), https://www.databreachtoday.asia/blizzard-entertainment-reports-breach-a-5034
22. Schnorr, C.P.: Efficient Identification and Signatures for Smart Cards. CRYPTO '90
23. Toubba, K.: Notice of Recent Security Incident (Dec 2022), https://blog.lastpass.com/2022/12/notice-of-recent-security-incident/
24. W3C Web Authentication Working Group: Web authentication: An API for accessing public key credentials Level 2 (Apr 2021), https://www.w3.org/TR/webauthn/

25. Williams, M.: Inside the Russian hack of Yahoo: How they did it (Oct 2017), https://www.csoonline.com/article/3180762/inside-the-russian-hack-of-yahoo-how-they-did-it.html
26. Yubico: .Net Yubikey SDK: User's Manual, https://docs.yubico.com/yesdk/users-manual/intro.html
27. Zhang, Z., Wang, Y., Yang, K.: Strong Authentication without Temper-Resistant Hardware and Application to Federated Identities. In: NDSS '20

A The ZWY Framework

In this section, we formalize the online unforgeability property presented by Zhang et al. [27]. In order to improve the clarity and the consistency with our framework of mkPBC, we made some minor changes to the syntax and security definitions of [27]. We explain the changes and why this does not affect the technical result.

A.1 Online Unforgeability of skPBC

The online unforgeability experiment of a single-key PBC is defined similarly to the weak unforgeability, with the roles of the credential ask and password pw reversed. That is, after the challenger has registered the users chosen by the adversary, the adversary receives the credentials ask_i's of all users but not the passwords pw_i. He can access user passwords through a reveal password oracle $\mathcal{O}_{\mathsf{RevPW}}$, though. Additionally, he has access to the same verify oracle $\mathcal{O}_{\mathsf{Vf}}$ and the same signing oracle $\mathcal{O}_{\mathsf{Sign}}$ as in the weak unforgeability game. The goal of the adversary is to forge an authentication token on a fresh message for a user whose password he has not revealed through a query to $\mathcal{O}_{\mathsf{RevPW}}$. The adversary can test passwords with the $\mathcal{O}_{\mathsf{Vf}}$ oracle, and thus, the security definition is weakened to online attacks. The experiment is given in Fig. 4, and the security definition is as follows:

Oracle $\mathcal{O}_{\mathsf{Vf}}(i,m,\tau)$	**Oracle** $\mathcal{O}_{\mathsf{Sign}}(i,m)$	**Oracle** $\mathcal{O}_{\mathsf{RevPW}}(i)$
Return $\mathsf{Vf}(ssk, uid_i, m, \tau)$	$\tau \leftarrow \mathsf{Sign}(uid_i, ask_i, pw_i, m)$	Add i to RevPW
	Add (i,m) to Q, **Return** τ	**Return** pw_i

Experiment $\mathsf{Exp}^{\mathsf{onlineUNF}}_{\mathcal{A},\mathsf{skPBC}}(\lambda)$

$(ssk, spk) \leftarrow \mathsf{KGen}(1^\lambda)$, $\mathsf{RevCred} \leftarrow \emptyset$, $Q \leftarrow \emptyset$

$(uid_1, \ldots, uid_n, st) \leftarrow \mathcal{A}(spk)$, for $i = 1, \ldots n : pw_i \xleftarrow{r} \mathcal{D}_{\mathsf{pw}}$

$(ask_i; -) \leftarrow \langle \mathsf{RegU}(spk, uid_i, pw_i), \mathsf{RegS}(ssk, uid_i) \rangle$

$(uid_j, m^*, \tau^*) \leftarrow \mathcal{A}^{\mathcal{O}_{\mathsf{RevPW}}, \mathcal{O}_{\mathsf{Sign}}, \mathcal{O}_{\mathsf{Vf}}}(st, ask_1, \ldots, ask_n)$

Return 1 if $\mathsf{Vf}(ssk, uid_j, m^*, \tau^*) = 1 \wedge j \in [n] \wedge j \notin \mathsf{RevPW} \wedge (j, m^*) \notin Q$

Fig. 4: Online Unforgeability for skPBC. The oracles use the values (i, uid_i, ask_i, pw_i) established during registration.

Definition 12 (Online Unforgeability). *A* skPBC *scheme is online unforgeable, if for all PPT adversaries* $\mathcal{A}$ *it holds that* $\Pr[\mathsf{Exp}^{\mathsf{onlineUNF}}_{\mathcal{A},\mathsf{skPBC}}(\lambda) = 1] \leq \frac{q_{\mathsf{Vf}}+1}{|\mathcal{D}_{\mathsf{pw}}|} + \mathsf{negl}(\lambda)$*, where* q_{Vf} *is the number of queries to the* $\mathcal{O}_{\mathsf{Vf}}$ *oracle.*

A.2 Changes to the ZWY Framework

Changes to the Syntax. We made the following minor changes to the syntax of ZWY [27]: (1) We do not explicitly describe the behaviour of the registration protocol if a party aborts. (2) We do not enforce the registration protocol to keep a registry *Reg* with *uid*'s but assume this happens on the application level.

Changes to the Security Experiments. The ZWY framework models password compromise through an oracle which reveals honest users' passwords. Since in the weak and strong unforgeability definition, the win condition of the adversary is independent of his knowledge of *pw*, we did not model this oracle but instead hand the adversary all user passwords directly.

Furthermore, the ZWY framework considers the forgery of a user who has not registered with the server a valid attack, while we removed this condition from the security experiment. We argue that this type of forgery is not a concern as it will be caught on the application level. This change was made to focus on attacks that are relevant to the security of the system.

B Comparison of mkPBC and skPBC

In this section, we detail how to obtain a secure skPBC from a secure mkPBC utilizing authenticated encryption. We first recall the definitions of ciphertext integrity and CCA security for a symmetric encryption scheme. We require both of these properties in our transformation and both of these properties are fulfilled by a symmetric authenticated encryption scheme [7, §9.2.3]. Then we formally describe the construction sketched in Section 3.2 and prove that it achieves weak and online unforgeability.

B.1 Ciphertext Integrity and CCA Security

Ciphertext integrity (CI) ensures that an attacker who can query an encryption oracle on chosen messages is unable to create a new ciphertext that decrypts properly.

Definition 13 (Ciphertext Integrity). *Let* $\Pi = (\mathsf{KGen}, \mathsf{Enc}, \mathsf{Dec})$ *be a symmetric encryption scheme. Define the following experiment* $\mathsf{Exp}^{\mathsf{CI}}_{\mathcal{A},\Pi}(\lambda)$ *between a challenger and an adversary* $\mathcal{A}$*:*

1. *The challenger generates* $k \leftarrow \mathsf{KGen}(1^\lambda)$ *and sends* 1^λ *to* $\mathcal{A}$*.*
2. $\mathcal{A}$ *has access to an encryption oracle: For* $i = 1, 2, ...$*, he can send messages* m_i *to the challenger, who returns* $c_i \leftarrow \mathsf{Enc}(k, m_i)$*.*

3. $\mathcal{A}$ *outputs a candidate ciphertext* c^*.
4. *If* $\mathsf{Dec}(k, c^*) \neq \perp$ *and* $c^* \notin \{c_1, c_2, \dots\}$, *the output of the experiment is 1.*

We say that Π *has ciphertext integrity if for all PPT adversaries* $\mathcal{A}$ *it holds that* $\Pr[\mathsf{Exp}^{\mathsf{CI}}_{\mathcal{A},\Pi}(\lambda) = 1] \leq \mathsf{negl}(\lambda)$.

Another security notion that we require is indistinguishability against chosen ciphertext attacks (IND-CCA). We use the Left-or-Right (LoR) style definition of CCA security which can be found in [7, §9.2.2] and which we recall here:

Definition 14 (IND-CCA Security). *Let* $\Pi = (\mathsf{KGen}, \mathsf{Enc}, \mathsf{Dec})$ *be a symmetric encryption scheme. Define the following experiment* $\mathsf{Exp}^{\mathsf{CCA}}_{\mathcal{A},\Pi}(\lambda)$ *between a challenger and an adversary* $\mathcal{A}$:

1. *The challenger generates* $k \leftarrow \mathsf{KGen}(1^\lambda)$, *chooses* $b \xleftarrow{r} \{0, 1\}$ *and sends* 1^λ *to* $\mathcal{A}$.
2. $\mathcal{A}$ *has access to an LoR-encryption oracle: For* $i = 1, 2, \dots$, *he can send messages* (m_{i0}, m_{i1}) *to the challenger, who returns* $c_i \leftarrow \mathsf{Enc}(k, m_{ib})$.
3. $\mathcal{A}$ *has access to a decryption oracle: For* $j = 1, 2, \dots$, *he can send ciphertexts* $\hat{c}_j$ *to the challenger, who returns* $m_j \leftarrow \mathsf{Dec}(k, \hat{c}_j)$.
4. $\mathcal{A}$ *outputs a bit* b'.
5. *If* $b = b'$ *and* $\forall j = 1, 2, \dots$ *it holds that* $\hat{c}_j \notin \{c_1, c_2, \dots\}$, *the output of the experiment is 1.*

We say that Π *is indistinguishable against chosen ciphertext attacks or (IND-) CCA secure if for all PPT adversaries* $\mathcal{A}$ *it holds that* $\Pr[\mathsf{Exp}^{\mathsf{CCA}}_{\mathcal{A},\Pi}(\lambda) = 1] \leq 1/2 + \mathsf{negl}(\lambda)$.

B.2 Transforming a Secure mkPBC into a Secure mkPBC

We are now able to formally define the transformation sketched in Section 3.2 and prove its security.

Definition 15 (skPBC from mkPBC and AE). *Given a multi-key PBC scheme* $\mathsf{mkPBC} = (\mathsf{mk.Setup}, \langle \mathsf{mk.RegU}, \mathsf{mk.RegS} \rangle, \mathsf{mk.Sign}, \mathsf{mk.Vf})$ *and symmetric authenticated encryption scheme* $\mathsf{AE} = (\mathsf{AE.KGen}, \mathsf{AE.Enc}, \mathsf{AE.Dec})$, *we define the single-key PBC scheme* $\mathsf{skPBC} = (\mathsf{sk.KGen}, \langle \mathsf{sk.RegU}, \mathsf{sk.RegS} \rangle, \mathsf{sk.Sign}, \mathsf{sk.Vf})$ *as follows:*

$\mathsf{sk.KGen}(1^\lambda)$: *Run* $spk \leftarrow \mathsf{mk.Setup}(1^\lambda)$, $ssk \leftarrow \mathsf{AE.KGen}(1^\lambda)$. *Output* (ssk, spk).

$\langle \mathsf{sk.RegU}(spk, uid, pw), \mathsf{sk.RegS}(ssk, uid) \rangle$: *Run* $(ask_{mk}, avk) \leftarrow \langle \mathsf{mk.RegU}(uid, pw), \mathsf{mk.RegS}(uid) \rangle$. *Compute* $c \leftarrow \mathsf{AE.Enc}(ssk, (uid, avk))$. *Output* $ask_{sk} := (ask_{mk}, c)$.

$\mathsf{sk.Sign}(uid, ask_{sk}, pw, m)$: *Parse* $ask_{sk} := (ask_{mk}, c)$, *compute* $\tau_{mk} \leftarrow \mathsf{mk.Sign}(uid, ask_{mk}, pw, m)$. *Output* $\tau_{sk} := (\tau_{mk}, c)$

$\mathsf{sk.Vf}(ssk, uid, m, \tau_{sk})$: *Parse* $\tau_{sk} := (\tau_{mk}, c)$. *Compute* $(uid', avk) \leftarrow \mathsf{AE.Dec}(ssk, c)$. *If* $uid' = uid$, *then output* $\mathsf{mk.Vf}(uid, avk, m, \tau_{mk})$, *else output 0.*

Theorem 6. *If* mkPBC *is a strongly unforgeable multi-key password based credential scheme, and if* AE *is a secure authenticated encryption scheme, then the* skPBC *scheme given in Definition 15 is a weakly unforgeable single-key password based credential scheme.*

Proof. The idea of the proof is to simulate the ciphertexts used in the skPBC weak unforgeability experiment with ciphertexts independent of the underlying plaintext. In order to simulate them perfectly, each user is assigned a random ciphertext c_i. That is, each ask_i contains a ciphertext c_i of a random message, and the oracles only verify the consistency of c_i. More precisely, the signing oracle appends c_i and the verify oracle checks whether a submitted token contains the same c_i. Then, the adversary can only win if he has forged an authentication token of the underlying mkPBC scheme.

We prove this by a hybrid argument. Let $\mathcal{A}$ be an efficient attacker in the weak unforgeability experiment for skPBC, and let $\mathcal{B}$ be an attacker in the strong unforgeability experiment for mkPBC who uses $\mathcal{A}$ as a subroutine.

Game G_0**:** This is the real experiment, where $\mathcal{B}$ plays the role of the challenger. Let Adv_i be the probability that $\mathcal{A}$ outputs a valid forgery of a fresh message in Game G_i.

$$\mathsf{Adv}_0 = \Pr[\mathsf{Exp}^{\mathsf{strongUNF}}_{\mathcal{A},\mathsf{skPBC}}(\lambda) = 1]$$

Game G_1**:** In this game, $\mathcal{B}$ changes the registration process of the experiment. For every user uid_i, instead of setting $c_i \leftarrow \mathsf{AE.Enc}(ssk, (uid_i, avk_i))$, c_i is chosen as $c_i \leftarrow \mathsf{AE.Enc}(ssk, m)$ for a randomly chosen message m from the message space of AE. $\mathcal{B}$ keeps a list L consisting of (c_i, uid_i, avk_i) to answer verification queries consistently. Whenever there is a query (i, m, τ) to $\mathcal{O}_{\mathsf{Vf}}$, $\mathcal{B}$ first parses $\tau := (\tau', c')$. If there is an entry (c', uid', avk') in the list L, then $\mathcal{B}$ returns $uid' = uid_i \wedge \mathsf{mk.Vf}(uid', avk', m, \tau')$. Otherwise, the simulation of $\mathcal{O}_{\mathsf{Vf}}$ is done as before.

Indistinguishability Argument: This change is indistinguishable, if the authenticated encryption scheme is CCA-secure, as the only change is in the creation of the c_i's while the rest of the game behaves identical. If an adversary was able to distinguish the two games, we could construct an efficient adversary $\mathcal{A}'$ breaking CCA-security of AE. To see this, note that ciphertexts in game G_0 are encryptions of (uid_i, avk_i) while in game G_1 they are encryptions of random messages m under the same ssk.

$$|\mathsf{Adv}_1 - \mathsf{Adv}_0| \leq \mathbf{Adv}^{\mathrm{CCA}}_{\mathcal{A}',\mathsf{AE}}(\lambda)$$

Game G_2**:** In G_2, for each query to $\mathcal{O}_{\mathsf{Vf}}$ with (i, m, τ), $\mathcal{B}$ parses $\tau := (\tau', c)$, and if $c \neq c_i$, he returns 0. Similarly, when $\mathcal{A}$ outputs a forgery (uid_j, m, τ), $\mathcal{B}$ parses $\tau := (\tau', c)$, and if $c \neq c_j$, $\mathcal{B}$ aborts.

Indistinguishability Argument: This game change is only distinguishable if the adversary $\mathcal{A}$ is able to create a valid and fresh ciphertext of (uid_i, avk_i) for some i. Thus, in order to distinguish, the adversary who might know $c_1, \ldots, c_n$ has to create a ciphertext c that decrypts properly, i.e. a ciphertext such that $\mathsf{Dec}(ssk, c) \neq \bot$. Thus, the games are only distinguishable with negligible probability if AE has ciphertext integrity.

$$|\mathsf{Adv}_2 - \mathsf{Adv}_1| \leq \mathbf{Adv}^{\mathrm{CI}}_{\mathcal{A}'',\mathsf{AE}}(\lambda)$$

Game G_3: In G_3, $\mathcal{B}$ guesses in advance the *uid* for which $\mathcal{A}$ will try to output a valid forgery. $\mathcal{B}$ guesses an index $j^* \in [n]$, and aborts if the adversary outputs a forgery for a user uid_j with $j \neq j^*$. This will occur with probability $(n-1)/n$. Note that this game hop is a transition based on a large failure event, however it is unavoidable when transitioning from a definition with multiple users to a single-user setting [3].

$$\mathsf{Adv}_2 = n \cdot \mathsf{Adv}_3$$

We now construct a reduction from the adversary $\mathcal{A}$ breaking the weak-unforgeability experiment in G_3 to an adversary $\mathcal{B}$ breaking the strong unforgeability experiment of mkPBC. The mkPBC adversary $\mathcal{B}$ receives public parameteres pp from the strong unforgeability game and initiates $\mathcal{A}(pp)$. After $\mathcal{A}$ has chosen n users, $\mathcal{B}$ sends uid_{j^*} to the challenger in the mkPBC experiment. He receives avk, pw and access to a sign oracle $\mathcal{O}_{\mathsf{Sign}}$. He then registers all users except for user j^* normally in the skPBC game. For user j^*, he does not run the registration protocol. Instead, he chooses $c_{j^*} \leftarrow \mathsf{AE.Enc}(ssk, m)$ for a random message m and sets her password pw_{j^*} to be pw. $\mathcal{B}$ simulates the $\mathcal{O}_{\mathsf{Sign}}$ oracle in the skPBC experiment for user j^* as follows: On input (j^*, m), $\mathcal{B}$ sends m to the $\mathcal{O}_{\mathsf{Sign}}$ oracle of mkPBC and receives token τ'. He then outputs $\tau := (\tau', c_{j^*})$. To simulate the $\mathcal{O}_{\mathsf{Vf}}$ oracle, on input (j^*, m, τ), $\mathcal{B}$ parses $\tau := (\tau', c)$. If $c = c_{j^*}$, he returns $\mathsf{mk.Vf}(uid_{j^*}, avk, m, \tau')$, otherwise he returns 0. Now, $\mathcal{B}$ simulates the experiment of G_3 indistinguishably to the adversary $\mathcal{A}$.
If $\mathcal{A}$ outputs (uid_{j^*}, m^*, τ^*), then $\mathcal{B}$ parses $\tau^* := (\tau', c)$ and outputs (m^*, τ') as forgery for the mkPBC game. If τ^* is a valid authentication token in G_3, then it must hold that $c = c_{j^*}$ (otherwise $\mathcal{B}$ would have aborted in G_2), and that $\mathsf{mk.Vf}(uid_{j^*}, avk, m, \tau') = 1$, thus it is also a valid authentication token in the mkPBC game. If m^* was never queried to $\mathcal{O}_{\mathsf{Sign}}$ by $\mathcal{A}$ in the skPBC experiment, then it was also never queried to $\mathcal{O}_{\mathsf{Sign}}$ by $\mathcal{B}$ in the mkPBC experiment. Thus, we have constructed an efficient adversary against the strong unforgeability experiment of mkPBC and it holds that

$$\mathsf{Adv}_3 \leq \Pr[\mathsf{Exp}^{\mathsf{strongUNF}}_{\mathcal{A},\mathsf{mkPBC}}(\lambda) = 1]$$

This concludes the proof.

Theorem 7. *If* mkPBC *is an online unforgeable multi-key password based credential scheme, and if* AE *is a secure, strongly unforgeable authenticated encryption scheme, then the* skPBC *scheme given in Def. 15 is a online unforgeable single-key password based credential scheme.*

Security Property	Leaked Values			Assumptions		
	User		Server			
	$ask = (k, pk_{\mathsf{Enc}})$	pw	$avk = (pk_{\mathsf{Sig}}, sk_{\mathsf{Enc}})$	F	Signature	Encryption
Strong Unforgeability	$\times$	✓	✓	Secure PRF	Unf	$\times$
Online Unforgeability	✓	$\times$	$\times$	RO	CROB & RI	CCA
Offline Unforgeability	✓	$\times$	✓	RO	Unf & RI	$\times$
Pw-Hiding	$\times$	$\times$	✓	Secure PRF	$\times$	$\times$

Table 1: Overview of the different security properties adapted to the PBC scheme and with the security assumptions needed for the properties for $\mathsf{PBC_{StE}}$. CROB stands for complete robustness and RI is randomness injectivity.

Proof. We prove this by a hybrid argument, similar to Theorem 6. G_0 is the real online unforgeability experiment with adversary $\mathcal{A}$ where the adversary $\mathcal{B}$ of the mkPBC online unforgeability experiment plays the role of the challenger. The first three game hops are the same as in the proof of Theorem 6, which makes the ciphertexts independent of the plaintext (uid, avk) and lets $\mathcal{B}$ choose the target user at the beginning of the experiment.

The efficient adversary $\mathcal{A}$ in G_3 can then be reduced to an efficient adversary $\mathcal{B}$ in the online unforgability experiment of mkPBC. $\mathcal{B}$ receives public parameters from the online unforgeability experiment, and starts $(uid_1, \ldots, uid_n, st) \leftarrow \mathcal{A}(pp)$. He registers (uid_{j^*}, st) with the mkPBC challenger and receives (ask_{mk,j^*}, st') and access to the oracles $\mathcal{O}_{\mathsf{Sign,mk}}$ and $\mathcal{O}_{\mathsf{Vf,mk}}$. He registers all users except for uid_{j^*} in the skPBC experiment as before, and for user j^*, he sets $ask_{sk,j^*} := (ask_{mk,j^*}, c_{j^*})$ where $c_{j^*} \leftarrow \mathsf{AE.Enc}(ssk, m)$ for a random message m. $\mathcal{B}$ simulates the $\mathcal{O}_{\mathsf{Sign,sk}}$ oracle for user j^* as follows: On input (j^*, m), he sends m to $\mathcal{O}_{\mathsf{Sign,mk}}$ and receives token τ'. The output of $\mathcal{O}_{\mathsf{Sign,sk}}$ is then $\tau := (\tau', c_{j^*})$. To simulate the $\mathcal{O}_{\mathsf{Vf,sk}}$ oracle for user j^*, on input (j^*, m, τ), $\mathcal{B}$ parses $\tau := (\tau', c)$. If $c = c_{j^*}$, he queries (m, τ') to the $\mathcal{O}_{\mathsf{Vf,mk}}$ oracle and returns the result of this query as output of the $\mathcal{O}_{\mathsf{Vf,sk}}$ oracle. With these modifications, $\mathcal{B}$ simulates the online unforgeability experiment for $\mathcal{A}$ perfectly. If $\mathcal{A}$ outputs (uid_{j^*}, m^*, τ^*), then $\mathcal{B}$ parses $\tau^* := (\tau', c)$ and outputs (m^*, τ') as valid forgery for the mkPBC experiment. This forgery is valid by the same argument as in Theorem 6, m^* was never queried to $\mathcal{O}_{\mathsf{Sign,sk}}$ and thus neither to $\mathcal{O}_{\mathsf{Sign,mk}}$. Each query to $\mathcal{O}_{\mathsf{Vf,sk}}$ on $(j^*, \cdot, \cdot)$ corresponds to one query to $\mathcal{O}_{\mathsf{Vf,mk}}$ from $\mathcal{B}$. Therefore it holds that $\mathsf{Adv}_3 \leq \Pr[\mathsf{Exp}^{\mathsf{onlineUNF}}_{\mathcal{B},\mathsf{mkPBC}}(\lambda) = 1]$, concluding the proof.

C Signatures with Complete Robustness

In this section, we give formal definitions of the Schnorr, DSA and BLS signature schemes and show that they all achieve complete robustness.

Definition 16 (Schnorr Signature Scheme). *The Schnorr [22] signature scheme* $\Pi_{Schnorr} = (\mathsf{Setup}, \mathsf{KGen}, \mathsf{Sign}, \mathsf{Vf})$ *is defined as follows:*

$\mathsf{Setup}(1^\lambda)$: *Output a tuple* $(\mathbb{G}, q, g)$ *consisting of a group* $\mathbb{G}$ *of prime order* q *and generator* g*, and the description of a hash function* $H : \{0,1\}^* \to \mathbb{Z}_q$*,* R_λ *is set to* $\mathbb{Z}_q$*.*

$\mathsf{KGen}(pp; r)$: *Compute* $pk \leftarrow g^r$*, set* $sk \leftarrow r$ *and output* (pk, sk)*.*

$\mathsf{Sign}(sk, m)$: *Choose* $k \xleftarrow{r} \mathbb{Z}_q$*, set* $I \leftarrow g^k$*, compute* $r \leftarrow H(I, m)$ *and* $s \leftarrow r \cdot sk + k \mod q$*. Output* $\sigma := (r, s)$*.*

$\mathsf{Vf}(pk, m, \sigma)$: *Parse* $\sigma := (r, s)$*, compute* $I \leftarrow g^s \cdot pk^{-r}$ *and output 1 if* $H(I, m) = r$*.*

Definition 17 (DSA Signature Scheme). *The DSA [17] signature scheme* $\Pi_{DSA} = (\mathsf{Setup}, \mathsf{KGen}, \mathsf{Sign}, \mathsf{Vf})$ *is defined as follows:*

$\mathsf{Setup}(1^\lambda)$: *Output a tuple* $(\mathbb{G}, q, g)$ *consisting of a group* $\mathbb{G}$ *of prime order* q *and generator* g*, and the description of a hash function* $H : \{0,1\}^* \to \mathbb{Z}_q$ *and a hash function* $F : \mathbb{G} \to \mathbb{Z}_q$*.* R_λ *is set to* $\mathbb{Z}_q$*.*

$\mathsf{KGen}(pp; r)$: *Compute* $pk \leftarrow g^r$*, set* $sk \leftarrow r$ *and output* (pk, sk)*.*

$\mathsf{Sign}(sk, m)$: *Choose* $k \xleftarrow{r} \mathbb{Z}_q^*$*, set* $r \leftarrow F(g^k)$*, compute* $s \leftarrow k^{-1} \cdot H(m) + xr \mod q$*. If* $r = 0$ *or* $s = 0$ *start again with a fresh choice of* k*. Output* $\sigma := (r, s)$*.*

$\mathsf{Vf}(pk, m, \sigma)$: *Parse* $\sigma := (r, s)$*. If* $r = 0$ *or* $s = 0$*, output 0. Output 1 if* $F(g^{H(m)\cdot s^{-1}} \cdot pk^{r \cdot s^{-1}}) = r$*.*

Definition 18 (BLS Signature Scheme). *The BLS [6] signature scheme* $\Pi_{BLS} = (\mathsf{Setup}, \mathsf{KGen}, \mathsf{Sign}, \mathsf{Vf})$ *is defined as follows:*

$\mathsf{Setup}(1^\lambda)$: *Output a tuple* $(\mathbb{G}, \mathbb{G}_T, q, g)$ *consisting of two groups* $\mathbb{G}, \mathbb{G}_T$ *of prime order* q *and generator* g*, and the description of a bilinear pairing* $e : \mathbb{G} \times \mathbb{G} \to \mathbb{G}_T$ *and a hash function* $H : \{0,1\}^* \to \mathbb{G}$*.* R_λ *is set to* $\mathbb{Z}_q$*.*

$\mathsf{KGen}(pp; r)$: *Compute* $pk \leftarrow g^r$*, set* $sk \leftarrow r$ *and output* (pk, sk)*.*

$\mathsf{Sign}(sk, m)$: *Output* $\sigma \leftarrow H(m)^{sk}$*.*

$\mathsf{Vf}(pk, m, \sigma)$: *Output 1 if* $e(\sigma, g) = e(H(m), pk)$*.*

Proof of Complete Robustness Since the complete robustness only considers the verification algorithm we can ignore the key generation and signing algorithms.

DSA: In DSA, a signature $\sigma := (r, s)$ verifies for m under pk if $F(g^{H(m)\cdot s^{-1}} \cdot pk^{r \cdot s^{-1}}) = r$ for two hash functions F and H. Thus, σ verifies under a second public key pk' only if $F(g^{H(m)\cdot s^{-1}} \cdot pk^{r\cdot s^{-1}}) = F(g^{H(m)\cdot s^{-1}} \cdot (pk')^{r \cdot s^{-1}})$ which happens only with negligible probability if F is collision resistant.

Schnorr: In Schnorr signatures, a signature $\sigma = (r, s)$ verifies under pk for message m if $H(g^s \cdot pk^{-r}, m) = r$ for a hash function H. Thus, σ verifies under a second public key pk' only if $H(g^s \cdot pk^{-r}, m) = H(g^s \cdot (pk')^{-r}, m)$ which happens only with negligible probability if the hash function H is collision resistant.

BLS: In BLS signatures, a signature σ verifies under pk for message m if $e(\sigma, g) = e(H(m), pk)$. Thus, it verifies under a second public key pk' only if $e(H(m), pk) = e(H(m), pk')$. But this means that $pk = pk'$ and the signature only verifies under a single public key $pk = pk'$.

D Proofs

In this section, we provide the full proof of security for our multi-key password based credential scheme $\mathsf{PBC}_{\mathsf{StE}}$.

D.1 Proof of Strong Unforgeability

We start with the proof of strong unforgeability. Recall that the adversary in the strong unforgeability experiment knows the verification key *avk* and *pw* of a user *uid* and has access to a sign oracle but does not know the user's credential *ask*. The adversary's goal is to create a valid authentication token on a fresh message for *uid*.

Proof of Theorem 1

Proof. Let $\mathcal{A}$ be an efficient adversary in the strong unforgeability experiment. We construct an adversary $\mathcal{B}$ for the unforgeability of Π_{Sign}, i.e. $\mathcal{B}$ is an adversary in the EUF-CMA experiment $\mathsf{Exp}^{\mathsf{EUF-CMA}}_{\mathcal{B},\Pi_{Sign}}(\lambda)$ that uses $\mathcal{A}$ as a subroutine. $\mathcal{B}$ receives a public key pk, has access to a signing oracle $\mathcal{O}_{\mathsf{Sign_S}}(m_i)$ which on input m_i returns $\mathsf{Sign}(sk, m_i)$ for an unknown secret key sk and is tasked to output a forgery (m^*, σ^*) such that $\mathsf{Vf}_S(pk, m^*, \sigma^*) = 1$, and m^* is fresh, i.e. was never queried to $\mathcal{O}_{\mathsf{Sign_S}}$.

We proceed with a hybrid argument. Let Adv_i be the probability that $\mathcal{A}$ outputs a valid forgery (for the mkPBC scheme) of a fresh message in Game G_i.

Game G_0: This is the real strong unforgeability experiment, where $\mathcal{B}$ plays the role of the challenger. If the adversary $\mathcal{A}$ outputs a valid forgery of a fresh message, then he wins the strong unforgeability game. Therefore

$$\mathsf{Adv}_0 = \Pr[\mathsf{Exp}^{\mathsf{strongUNF}}_{\mathcal{A},\mathsf{PBC}_{\mathsf{StE}}}(\lambda) = 1]$$

Game G_1: In G_1, the PRF output is replaced with a random value. During registration, $\mathcal{B}$ does not compute $r \leftarrow F(k, pw)$ but instead samples $r \xleftarrow{r} \mathcal{R}$ to get $(pk_{\mathsf{Sig}}, sk_{\mathsf{Sig}}) := \mathsf{KGen}_S(pp; r)$. $\mathcal{B}$ stores the value of sk_{Sig} and uses it to simulate the $\mathcal{O}_{\mathsf{Sign}}$ oracle, which returns $\tau \leftarrow \mathsf{Enc}(pk_{\mathsf{Enc}}, \mathsf{Sign}_S(sk_{\mathsf{Sig}}, (uid, m_i)))$ on message m_i (instead of recomputing $r \leftarrow F(k, pw)$ to sign the message). **Indistinguishability Argument:** This game is indistinguishable from G_0 if F is a secure PRF. If the games were distinguishable, we could create an efficient PRF distinguisher $\mathcal{C}$ who can distinguish the output of the PRF $F(k, pw)$ from the output of a truly random function $f(pw)$. Formally, the PRF distinguisher $\mathcal{C}$ has access to an oracle $\mathcal{O}(\cdot)$ and has to decide whether $\mathcal{O}(\cdot)$ returns $F(k, \cdot)$ or $f(\cdot)$ for a random function $f : \mathcal{D}_{\mathsf{pw}} \rightarrow \mathcal{R}$. $\mathcal{C}$ starts the strong unforgeability experiment with adversary $\mathcal{A}$ and after receiving (uid, st), picks $pw \xleftarrow{r} \mathcal{D}_{\mathsf{pw}}$, sets $r \leftarrow \mathcal{O}(pw)$ and computes $(pk_{\mathsf{Sig}}, sk_{\mathsf{Sig}}) := \mathsf{KGen}_S(pp; r)$. To simulate the $\mathcal{O}_{\mathsf{Sign}}$ oracle, the distinguisher $\mathcal{C}$ on input m_i returns $\sigma \leftarrow \mathsf{Enc}(pk_{\mathsf{Enc}}, \mathsf{Sign}(sk_{\mathsf{Sig}}, (uid, m_i)))$. $\mathcal{C}$ hands the adversary values

(avk, pw, st). If $\mathcal{C}$'s oracle returns the output of the PRF $F(k, pw)$, then from $\mathcal{A}$'s point of view this experiment is identical to G_0, and if the oracle returns $f(pw)$ for a random function f, the experiment is identical to G_1. Therefore, it holds that

$$|\mathsf{Adv}_1 - \mathsf{Adv}_0| \leq \mathbf{Adv}^{\mathrm{PRF}}_{\mathcal{C},F}(1^\lambda)$$

Game G_2: In G_2, $\mathcal{B}$ embeds the values of the EUF-CMA experiment with Π_{Sign}: He sets $pk_{\mathsf{Sig}} \leftarrow pk$ during registration, and simulates the sign oracle $\mathcal{O}_{\mathsf{Sign}}$ with the sign oracle $\mathcal{O}_{\mathsf{Sign}_\mathsf{S}}$ by returning $\tau \leftarrow \mathsf{Enc}(pk_{\mathsf{Enc}}, \mathcal{O}_{\mathsf{Sign}_\mathsf{S}}(uid, m_i))$ on message m_i. When $\mathcal{A}$ outputs (m^*, τ^*), then $\mathcal{B}$ computes $\sigma^* \leftarrow \mathsf{Dec}(sk_{\mathsf{Enc}}, \tau^*)$ and outputs $((uid, m^*), \sigma^*)$ as forgery in the unforgeability experiment with Π_{Sign}.

Indistinguishability Argument: In Game 2, pk is taken from the EUF-CMA experiment where it is constructed from $(pk, sk) := \mathsf{KGen}_S(1^\lambda; r)$ for a random $r \xleftarrow{r} \mathcal{R}_\lambda$. The distribution of pk is therefore equivalent in G_1 and G_2. Furthermore, the oracle $\mathcal{O}_{\mathsf{Sign}}$ in G_2 returns tokens that are constructed exactly as in G_1. Therefore, it holds that

$$\mathsf{Adv}_2 = \mathsf{Adv}_1$$

Success Probability of $\mathcal{B}$: If $\mathcal{A}$ outputs a valid forgery τ^* on a fresh message m^*, i.e. if $\mathcal{A}$ wins the strong unforgeability experiment, then by the construction of $\mathsf{PBC}_{\mathsf{StE}}$ it holds that $\mathsf{Vf}_S(pk_{\mathsf{Sig}}, (uid, m^*), \mathsf{Dec}(sk_{\mathsf{Enc}}, \tau^*)) = 1$, and m^* was never queried to $\mathcal{O}_{\mathsf{Sign}}$. But this means that $\mathcal{B}$ also never queried (uid, m^*) to $\mathcal{O}_{\mathsf{Sign}_\mathsf{S}}$, and $\mathcal{B}$ wins the EUF-CMA experiment by outputting $((uid, m^*), \sigma^*)$. Thus, we can construct a successful attacker $\mathcal{B}$ against Π_{Sign} from a successful attacker $\mathcal{A}$ in G_2 and it holds that

$$\mathsf{Adv}_2 \leq \mathbf{Adv}^{\text{EUF-CMA}}_{\mathcal{B},\Pi_{Sign}}(\lambda)$$

Combining the game hops, we obtain

$$\Pr[\mathsf{Exp}^{\mathsf{strongUNF}}_{\mathcal{A},\mathsf{PBC}_{\mathsf{StE}}}(\lambda) = 1] \leq \mathbf{Adv}^{\mathrm{PRF}}_{\mathcal{C},F}(1^\lambda) + \mathbf{Adv}^{\text{EUF-CMA}}_{\mathcal{B},\Pi_{Sign}}(\lambda)$$

This concludes the proof.

D.2 Proof Of Online Unforgeability

In the online unforgeability experiment, the adversary now obtains the credential ask of the user but not her verification key avk or password pw. The adversary has access to both a sign and verifiy oracle and his goal is again to create a valid authentication token on a fresh message.

Since the adversary's information gain through the verify oracle is an important aspect of the proof, we define an experiment to capture this information gain. We require that for a signature scheme, interacting with a verify oracle instantiated with an unknown public key pk should only leak limited information about pk. This tailored experiment will be helpful to prove the online unforgeability of Π_{Sign}, and we show that every completely robust signature scheme fulfills this property which we formally define as pk-guessing security.

Therefore, we define the following PK-Guessing game for a signature scheme $\Pi := (\mathsf{Setup}, \mathsf{KGen}, \mathsf{Sign}, \mathsf{Vf})$: The adversary provides a list of public keys $\bar{PK} := (pk_1, \ldots, pk_n)$. The challenger chooses one of the public keys pk_i at random and gives the adversary access to a verify oracle with pk_i. The adversary's goal is to determine the correct index i of the public key chosen by the challenger.

The adversary has full control over the public keys he provides to the challenger, so we can assume that he might know the corresponding secret key. Then he can create a signature for each public key pk' in his list and test whether this signature verifies under the verify oracle, thus checking whether $pk' = pk_i$. We require that the signature scheme does not allow for any attack venue better than that, i.e. no adversary should do better than the one testing signatures for each public key from $\bar{PK}$ against the verify oracle. The experiment is defined in Fig. 5.

Experiment $\mathsf{Exp}_{\mathcal{A},\Pi}^{\mathsf{PK-Guessing}}(n)$
$pp \leftarrow \mathsf{Setup}(1^\lambda)$
$(pk_1, \ldots, pk_n, st) \leftarrow \mathcal{A}(pp, n)$
$i \xleftarrow{r} \mathbb{Z}_n$
$i^* \leftarrow \mathcal{A}^{\mathsf{Vf}(pk_i, \cdot, \cdot)}(st)$
Return 1 if $i^* = i \wedge (\forall i, j \in \mathbb{Z}_n, i \neq j : pk_i \neq pk_j)$.

Fig. 5: PK-Guessing Game.

We say that a signature scheme is pk-guessing secure, if for all PPT adversaries $\mathcal{A}$ it holds that:

$$\Pr[\mathsf{Exp}_{\mathcal{A},\Pi}^{\mathsf{PK-Guessing}}(n) = 1] \leq \frac{q_{\mathsf{Vf}} + 1}{n}$$

This property is a construction tailored to our scenario, however we can show that this security property is implied by complete robustness.

Theorem 8. *Every signature scheme with complete robustness has pk-guessing security.*

Proof. Since every signature only verifies under one public key, and the public key is chosen uniformly at random from all n public keys provided by the adversary, the best strategy for the adversary is to pick message-signature-pairs that verify under one of the public keys and submit it to the verify oracle. If this check returns 1 for one of them, the adversary can output the message signature pair and win. This check returns 1 with probability $1/n$ since pk_i is chosen uniformly at random from the set of all n public keys. In case the Vf oracle returns 0, then this leaks no information to the adversary other than $pk_i \neq pk'$ if the submitted message-signature-pair verifies under (the unique) public key pk'. Therefore the probability of the adversary winning the PK-Guessing game is bounded by $(q_{\mathsf{Vf}} + 1)/n$ since each of the q_{Vf} queries to Vf has probability at most $1/n$ at succeeding and the output message-signature-pair also has probability of $1/n$ of verifying under pk_i.

Proof Of Theorem 2 Again, we proceed with a series of game hops. Let $\mathcal{A}$ be an efficient adversary in the online unforgeability experiment, and let $\mathcal{B}$ be an adversary in the PK-Guessing experiment who uses $\mathcal{A}$ as a subroutine.

Game G_0: This is the original online unforgeability experiment, where $\mathcal{B}$ plays the role of the challenger. $\mathcal{B}$ simulates the random oracle F by lazy sampling. That is, $\mathcal{B}$ maintains a list L of entries (k, pw, r). For each query (k', pw') to F, $\mathcal{B}$ checks whether there is a entry (k', pw', r') in L. If this is the case, $\mathcal{B}$ returns r', otherwise he returns a random $r' \xleftarrow{r} \mathcal{R}_\lambda$ and adds (k', pw', r') to L. We have

$$\mathsf{Adv}_0 = \Pr[\mathsf{Exp}^{\mathsf{onlineUNF}}_{\mathcal{A},\mathsf{PBC}_{\mathsf{StE}}}(\lambda) = 1]$$

Game G_1: In this game, $\mathcal{B}$ makes answers of the $\mathcal{O}_{\mathsf{Sign}}$ oracle independent of the input message m_i. That is, for every query m_i to $\mathcal{O}_{\mathsf{Sign}}$, $\mathcal{B}$ chooses a random σ_i from the signature space of Π_{Sign}, computes $\tau_i \leftarrow \mathsf{Enc}(pk_{\mathsf{Enc}}, \sigma_i)$, stores (m_i, τ_i) in a list Q_{Sign} and returns τ_i. For every query (m_i, τ_i) to $\mathcal{O}_{\mathsf{Vf}}$, if $(m_i, \tau_i) \in Q_{Sign}$, the challenger returns 1, otherwise the challenger proceeds exactly as in G_0, i.e. checks validity of the signature under avk.

Indistinguishability Argument: This change is indistinguishable if Π_{Enc} is CCA-secure. We show this by a reduction. If these games were distinguishable, we could construct an adversary $\mathcal{C}^{\mathsf{Dec}(sk,\cdot),\mathsf{Enc}_{LoR_b}(pk,\cdot,\cdot)}(\lambda)$ breaking the CCA-security of Π_{Enc}. The adversary $\mathcal{C}$ proceeds as follows: For every query m_i from the adversary $\mathcal{A}$ to the $\mathcal{O}_{\mathsf{Sign}}$ oracle, $\mathcal{B}$ sets $\sigma_0 \leftarrow \mathsf{Sign}_S(sk_{\mathsf{Sig}}, (uid, m_i))$, $\sigma_1 \xleftarrow{r} \mathcal{S}$ (signature space $\mathcal{S}$) and submits (σ_0, σ_1) to the LoR-encryption oracle. He then outputs the answer c he receives from the oracle to answer the $\mathcal{O}_{\mathsf{Sign}}$ query.

For a $\mathcal{O}_{\mathsf{Vf}}$ query (m_i, τ_i), if $(m_i, \tau_i) \in Q_{Sign}$, then $\mathcal{C}$ returns 1, otherwise he sends τ_i to his $\mathsf{Dec}(sk, \cdot)$ oracle to receive σ_i and outputs $\mathsf{Vf}_S(pk_{\mathsf{Sig}}, (uid, m_i), \sigma_i)$. At the end, if the output of $\mathcal{A}$ is (m^*, τ^*) with $m^* \notin Q$, $\mathcal{B}$ sends τ^* to his Dec oracle to receive σ^* and returns 1 if $\mathsf{Vf}_S(pk_{\mathsf{Sig}}, (uid, m^*), \sigma^*) = 1$. If the LoR-oracle returns the encryption of the left message, the game is equivalent to G_0 and if the oracle returns the encryption of the right message, the game is equivalent to G_1. Thus,

$$|\mathsf{Adv}_1 - \mathsf{Adv}_0| \leq \mathbf{Adv}^{\mathrm{CCA}}_{\mathcal{C},\Pi_{Enc}}(\lambda)$$

Game G_2: In order to exclude two for one password tests, $\mathcal{B}$ aborts on a collision of the random oracle.

Indistinguishability argument: The games are indistinguishable unless an abort happens. By the birthday bound, this probability is given by

$$|\mathsf{Adv}_2 - \mathsf{Adv}_1| \leq \frac{q_F^2}{|\mathcal{R}|}$$

Game G_3**:** In this game, $\mathcal{B}$ uses $\mathcal{A}$ to win the PK-Guessing game. First, $\mathcal{B}$ computes for all passwords $pw_1, pw_s, \ldots, pw_{|\mathcal{D}_{\mathsf{pw}}|}$ the values

$$\begin{gathered}(sk_1, pk_1) := \mathsf{KGen}_S(pp; F(k^*, pw_1)),\\ \vdots \\ (sk_{|\mathcal{D}_{\mathsf{pw}}|}, pk_{|\mathcal{D}_{\mathsf{pw}}|}) := \mathsf{KGen}_S(pp; F(k^*, pw_{|\mathcal{D}_{\mathsf{pw}}|}))\end{gathered}$$

and submits $(pk_1, \ldots, pk_{|\mathcal{D}_{\mathsf{pw}}|})$ to the PK-Guessing challenger. Note that each of the submitted pk_i is unique due to the randomness injectivity property. To simulate the $\mathcal{O}_{\mathsf{Vf}}$ oracle, on input (m_j, τ_j) to $\mathcal{O}_{\mathsf{Vf}}$, $\mathcal{B}$ computes $\sigma_j \leftarrow \mathsf{Dec}(sk_{\mathsf{Enc}}, \tau_j)$, submits $((uid, m_j), \sigma_j)$ to his own $\mathsf{Vf}(pk_i, \cdot, \cdot)$ oracle and returns the result. When $\mathcal{A}$ outputs a forgery (m^*, τ^*), $\mathcal{B}$ checks for all $j \in \{1, \ldots, |\mathcal{D}_{\mathsf{pw}}|\}$ if $\mathsf{Vf}(pk_j, (uid, m^*), \tau^*) = 1$. If there is such a j, he outputs $i^* := j$ as his guess to the PK-Guessing game, otherwise he outputs $i^* := 1$. $\mathcal{B}$ simulates the experiment exactly as in G_2, therefore

$$\mathsf{Adv}_3 = \mathsf{Adv}_2$$

If $\mathcal{A}$ wins in G_3, then he has output a forgery (m^*, τ^*) such that $\mathsf{Vf}(pk_i, (uid, m^*), \mathsf{Dec}(sk_{\mathsf{Enc}}, \tau^*)) = 1$ and $\mathcal{B}$ wins the PK-Guessing game. Thus,

$$\mathsf{Adv}_3 \leq \mathbf{Adv}^{\mathsf{PK-Guessing}}_{\mathcal{B}, \Pi_{Sign}}(|\mathcal{D}_{\mathsf{pw}}|) \leq \frac{q_{\mathsf{Vf}} + 1}{|\mathcal{D}_{\mathsf{pw}}|}$$

Combining the game hops, we obtain

$$\Pr[\mathsf{Exp}^{\mathsf{onlineUNF}}_{\mathcal{A}, \mathsf{PBC}_{\mathsf{StE}}}(\lambda) = 1] \leq \mathbf{Adv}^{\mathrm{CCA}}_{\mathcal{C}, \Pi_{Enc}}(\lambda) + \frac{q_F^2}{|\mathcal{R}|} + \frac{q_{\mathsf{Vf}} + 1}{|\mathcal{D}_{\mathsf{pw}}|}$$

This concludes the proof.

D.3 Proof of Offline Unforgeability

In the offline unforgeability experiment, the adversary knows both the credential ask and verification key avk of the user, but not her password pw, and he still has access to a sign oracle. The adversary's goal stays the same – creating a valid authentication token on a fresh message – and we require that the adversary cannot do better than offline-attacking the password.

Proof of Theorem 3 Let $\mathcal{A}$ be an efficient adversary in the offline unforgeability experiment. As in the proof of Theorem 1, we construct an adversary $\mathcal{B}$ for the EUF-CMA experiment of Π_{Sign} using the adversary $\mathcal{A}$ as a subroutine. The game hops are as follows:

Game G_0**:** This is the original offline unforgeability experiment, where $\mathcal{B}$ plays the role of the challenger. Recall that in this game, $\mathcal{A}$ has the user's credential $ask = (k, pk_{\mathsf{Enc}})$ and the server's verification key $avk = (pk_{\mathsf{Sig}}, sk_{\mathsf{Enc}})$, so he can mount offline attacks on the user's password pw. Since F is modeled as a random oracle, an offline test on password pw' by the adversary has to involve a query (k, pw') to the random oracle F. Therefore we assume that each query to F on (k', pw') with $k = k'$ is counted as a query pw' to $\mathcal{O}_{\mathsf{TestPW}}$, and thus it holds that $q_F \leq q_f$.
$\mathcal{B}$ simulates the random oracle F by lazy sampling as in G_0 of the proof of Theorem 2. We have

$$\mathsf{Adv}_0 = \Pr[\mathsf{Exp}^{\mathsf{offlineUNF}}_{\mathcal{A},\mathsf{PBC_{StE}}}(\lambda) = 1]$$

Game G_1**:** In this game, $\mathcal{B}$ aborts on a collision of the random oracle. This is the first step to exclude two for one password tests.
Indistinguishability Argument: G_0 and G_1 are identical unless an abort happens. This probability is bounded by the birthday bound, and therefore

$$|\mathsf{Adv}_1 - \mathsf{Adv}_0| \leq \frac{q_F^2}{|\mathcal{R}_\lambda|}$$

Game G_2**:** In this game, $\mathcal{B}$ registers the user uid with a random $r \xleftarrow{r} \mathcal{R}_\lambda$ instead of using $r = F(k, pw)$.
Indistinguishability argument: Due to the randomness injectivity, there is only one $r = F(k, pw)$ such that $(pk_{\mathsf{Sig}}, sk_{\mathsf{Sig}}) = \mathsf{KGen}_S(pp; r)$. Under the random oracle assumption, this change is therefore only noticeable if the adversary queries F on exactly the input (k, pw). The adversary knows k, and the probability of the adversary correctly guessing the pw chosen uniformly at random by $\mathcal{B}$ is $1/|\mathcal{D}_{\mathsf{pw}}|$. Hence, this game hop is distinguishable with probability at most $q_F/|\mathcal{D}_{\mathsf{pw}}|$ where q_F is the number of queries to the random oracle F. Note that this bound is not negligible in λ. Thus,

$$|\mathsf{Adv}_2 - \mathsf{Adv}_1| \leq \frac{q_F}{|\mathcal{D}_{\mathsf{pw}}|} \leq \frac{q_f}{|\mathcal{D}_{\mathsf{pw}}|}$$

Game G_3**:** In G_3, we make a reduction to the unforgeability of Π_{Sign}. The reduction is identical to the reduction in G_2 of the proof of Theorem 1, thus we omit it here. The reduction shows that every successful adversary $\mathcal{A}$ in G_3 can be transformed into an adversary $\mathcal{B}$ forging a signature for Π_{Sign}. Thus,

$$\mathsf{Adv}_3 = \mathsf{Adv}_2$$
$$\mathsf{Adv}_3 \leq \mathbf{Adv}^{\text{EUF-CMA}}_{\mathcal{B},\Pi_{Sign}}(\lambda)$$

Combining all game hops, we obtain

$$\Pr[\mathsf{Exp}^{\mathsf{offlineUNF}}_{\mathcal{A},\mathsf{PBC}_{\mathsf{StE}}}(\lambda) = 1] \leq \frac{q_F^2}{|\mathcal{R}_\lambda|} + \frac{q_f}{|\mathcal{D}_{\mathsf{pw}}|} + \mathbf{Adv}^{\text{EUF-CMA}}_{\mathcal{B},\Pi_{Sign}}(\lambda)$$

This concludes the proof.

D.4 Proof of Password Hiding

In order to show that $\mathsf{PBC}_{\mathsf{StE}}$ is password hiding, we need to show that the adversary who receives the verification key avk of a user and has access to a sign oracle cannot distinguish which of his chosen passwords pw_0 or pw_1 was used to instantiate the user.

Proof of Theorem 4 We show that the verification key does not leak information about the underlying password if the PRF F is secure. Let $\mathcal{A}$ be an efficient adversary in the pw-hiding experiment, and let $\mathcal{B}$ be an adversary who plays the role of the challenger in the experiment.

Game G_0**:** This is the real pw-hiding experiment, where $\mathcal{B}$ plays the role of the challenger. That is, $\mathcal{B}$ receives (uid, pw_0, pw_1, st), picks $b \xleftarrow{r} \{0,1\}$ and creates $avk = (pk_{\mathsf{Sig}}, sk_{\mathsf{Enc}})$ using $(pk_{\mathsf{Sig}}, sk_{\mathsf{Sig}}) := \mathsf{KGen}_S(pp; F(k, pw_b))$. He passes avk to $\mathcal{A}$ who has to guess the correct bit b while having access to a sign oracle $\mathcal{O}_{\mathsf{Sign}}$. Let Adv_i be the probability that $\mathcal{A}$ outputs the correct bit b chosen by $\mathcal{B}$ in game G_i.

$$\mathsf{Adv}_0 = \Pr[\mathsf{Exp}^{\mathsf{PW-Hiding}}_{\mathcal{A},\mathsf{PBC}_{\mathsf{StE}}}(\lambda) = 1]$$

Game G_1**:** In this game, $\mathcal{B}$ creates $(pk_{\mathsf{Sig}}, sk_{\mathsf{Sig}}) := \mathsf{KGen}_S(pp; r)$ using a random $r \xleftarrow{r} \mathcal{R}_\lambda$ instead of $r = F(k, pw_b)$. $\mathcal{B}$ remembers sk_{Sig} and simulates the sign oracle $\mathcal{O}_{\mathsf{Sign}}$ by returning $\mathsf{Enc}(pk_{\mathsf{Enc}}, \mathsf{Sign}(sk_{\mathsf{Sig}}, (uid, m_i)))$ on input m_i.
Indistinguishability Argument: This change is identical to the change from G_0 to G_1 in the Proof of Theorem 1. Therefore, it holds that

$$|\mathsf{Adv}_1 - \mathsf{Adv}_0| \leq \mathbf{Adv}^{\text{PRF}}_{\mathcal{C},F}(1^\lambda)$$

Now, all values in G_1 are independent of the password pw_b and the adversary can only guess the bit b. Therefore, it holds that

$$\mathsf{Adv}_1 = \frac{1}{2}$$

Combining the game hops, we receive

$$\Pr[\mathsf{Exp}^{\mathsf{PW-Hiding}}_{\mathcal{A},\mathsf{PBC}_{\mathsf{StE}}}(\lambda) = 1] \leq \frac{1}{2} + \mathbf{Adv}^{\text{PRF}}_{\mathcal{C},F}(1^\lambda)$$

This concludes the proof.