Publ. Math. Debrecen
Supplementum **100** (2022), 533–565
DOI: 10.5486/PMD.2022.Suppl.1

Provably secure identity-based remote password registration

By Csanád Bertók, Andrea Huszti, Szabolcs Kovács and Norbert Oláh

Abstract. One of the most significant challenges is the secure user authentication. If it becomes breached, confidentiality and integrity of the data or services may be compromised. The most widespread solution for entity authentication is the password-based scheme. It is easy to use and deploy. During password registration typically users create or activate their account along with their password through their verification email, and service providers are authenticated based on their Secure Sockets Layer / Transport Layer Security (SSL/TLS) certificate. We propose a certificate-less secure blind registration protocol (CLS-BPR) which is a password registration scheme based on identity-based cryptography, i.e., both the user and the service provider are authenticated by their short-lived identity-based secret key. For secure storage a bilinear map with a salt is applied, therefore in case of an offline attack the adversary is forced to calculate a computationally expensive bilinear map for each password candidate and salt that slows down the attack. New adversarial model with new secure password registration scheme are introduced. We show that the proposed protocol is based on the assumptions that solving the Bilinear Diffie–Hellman problem is computationally infeasible, the bilinear map is a one-way function, Mac is existentially unforgeable under an adaptive chosen-message attack, where the bilinear map is considered in the generic bilinear group model and the hash functions are supposed as random oracles.

Mathematics Subject Classification: 94A60, 94A62.
Key words and phrases: identity-based cryptograpy, elliptic curves, pairing, password registration.
The presented research has been partially supported by the SETIT Project (no. 2018-1.2.1-NKP-2018-00004) and TKP Project (No. TKP2020-NKA-04). SETIT Project has been implemented with the support provided from the National Research, Development and Innovation Fund of Hungary, financed under the 2018-1.2.1-NKP funding scheme. Project no. TKP2020-NKA-04 has been implemented with the support provided from the National Research, Development and Innovation Fund of Hungary, financed under the 2020-4.1.1-TKP2020 funding scheme. Research was supported by Eötvös Loránd Research Network.

1. Introduction

Information systems are only as secure as their weakest point. Entity authentication is based on possession of secret information, known or verified only by the entities participating in the process. Many authentication systems depend on a password, which is a string of characters used to verify the identity of a user. Since it is easy to use and deploy, password usage is a widespread form of user authentication. Passwords are applied in many cryptographic schemes and systems, e.g., password authentication schemes or Password-Based Key Derivation Function (PBKDF) [35], [44], [10], [27].

Furthermore, in certain cases passwords serve as authentication data for key exchange protocols. The basic setting takes two parties into account that share the same password with the goal of establishing shared master or session keys. Such model, known as Password-Authenticated Key Exchange (PAKE), was first studied by Bellovin and Merritt [4] and later formalized by Bellare et al. [3] in the game-based indistinguishability approach but several PAKE protocols were also recommended [9], [11], [30], [21].

In practice, passwords are often used for Single Sign-On (SSO) [31], [18] or the Kerberos authentication protocol [20]. These solutions are usually centralized and their main advantage is that users only need to authenticate themselves once. However, there are some disadvantages, including the Single Point of Failure, the Multi-User Computer Risks or Potential Data Leaks, which can pose serious threats if an attacker successfully compromises a user or service provider and it can break the security of multiple accounts [33], [41].

Even though the remote registration of passwords is probably one of the most important aspects of security and the initial step of any remote password-based protocol, it receives insufficient attention. In cryptographic password-based protocols, password registration is often skipped assuming that the passwords are set securely and known to the parties before the protocol is executed and implemented. During the implementation of registration, the chosen passwords are transmitted to the server through a secure channel (e.g., TLS channel) and users create or activate their account with their password through the verification email. Nevertheless, the TLS implementations are rather complex with the use of certifications as the users need to manage and update them. Registrations may be incomplete, and one of the shortcomings is when the TLS channel is not used, it may lead to a breach and leakage of confidential data (which is in conflict with the General Data Protection Regulation (GDPR)).

Moreover, the vulnerabilities of password schemes are well-known. Users may choose "weak" passwords or not change the default passwords, which are easy for attackers to guess. Another criticism of passwords is that if any site where a certain password is reused becomes compromised and the system administrators do not follow the best industry practices, the participants' other accounts may also become compromised by the attacker who can guess the password with an offline attack. Several attacks aim to figure out passwords, applying dictionary, rainbow tables or brute-force attack. An attacker conducting an offline attack hashes each password guess. While many common hashing algorithms were designed to make execution quicker, certain hashing algorithms were deliberately designed to be slow in order to hobble attackers conducting an offline attack. For example the bcrypt hashing scheme [37] can be configured with a cost factor that exponentially increases its execution time by requiring a sequential series of computations. Besides the password, usually a salt value is also used. Salt is a short (12 - 48 bits) random piece of data that is concatenated with the password before hashing. It is then stored with the hash of password information. An attacker who succeeds in stealing the password file or database is forced to run an exhaustive, computationally expensive offline attack to find the users's passwords from the salted hashes. Another way to make cracking of hashed passwords more complicated is to iterate the hash calculation [23]. Even with a very strong hardware an attacker would be able to crack only a small number of passwords in a given time frame. However, password cracking methods and the algorithm speed improved. The "password-cracking tools" can be very efficient (such as Hashcat [19] and John the Ripper [34]) and it can be assumed that the hash values of the leaked passwords are not safer than plain passwords when compromised by an attacker [13], [29], [8].

Additional solutions could also be applied for improving password security (and increasing password entropy), such as password meters, pre-assigned strong passwords, and salts. An alternative login method could be used where smart cards perform the authentication, however, these methods include another set of flaws [17], [36], [40], [43]. The password policies serve to rule out potentially weak passwords and by this contribute to the protection of IT systems. When implementing a web-based password authentication mechanism, typically a password registration is applied with the password policy determined by the server. The corresponding compliance check of the password is performed either by the client or on the server side and depends on the available trust assumptions. Another drawback of registrations is that the client's password is revealed to the server, which means clients need to trust the server to process and store the received

password. However, server compromise attacks are often based on plain password databases where passwords are easily revealed, thus they must be protected (e.g., by password hashing). On the server side the hash of the password is stored for each user in a password file or database. During login, passwords usually appear in cleartext at the server, and security can be harmed if the TLS channel is established with a compromised server's public key (a major concern given today's common Public Key Infrastructure (PKI) failures). When PKI becomes compromised, the software does not verify certifications correctly and users accept invalid or suspicious certifications.

To improve the security of password registration, Kiefer and Manulis introduced a new family of protocols called Blind Password Registration (BPR) for Verifier-Based Password-Authenticated Key Exchange (VPAKE) [24] and two-server PAKE [26]. They proposed Zero-Knowledge Password Policy Checks (ZKPPC) which enables blind registration. Users register their chosen password with a server and prove that it suits the password policy without revealing any information about the password, which prevents password leakage from the server. BPR protocol can be executed over the TLS channel established between the client and the server. They define a security model for stand-alone blind password registration protocols which fulfil policy compliance and dictionary attack resistance requirements. OPAQUE [28] is an asymmetric PAKE protocol containing a password registration scheme as well. Password registration can be offline, PKI-based or out-of-band. Blindness and resistance against offline attacks are considered.

An alternative to certificate-based cryptography and TLS is the identity-based cryptography, which was first proposed by Shamir in 1984 [38]. The basic idea of identity-based cryptography involves an identity-based asymmetric key pair where an arbitrary string can be used as a user public key. For this, a trusted authority or Private Key Generator (PKG) is required to generate private keys from public keys and a master secret key. The PKG also publishes public information required for all encryption, decryption, signature, and verification algorithms in the system. This is referred to as *Identity-Based Cryptography* (IBC) and Boneh and Franklin [7] formalized the notion of Identity-Based Encryption (IBE), which uses bilinear pairings over elliptic curve groups. In the IBE setting, the public key of a user can be any arbitrary string which is typically an e-mail address. There is no need for Bob to go to the Certificate Authority to verify Alice's public key. In this way, an IBE can greatly simplify certificate management. However, Identity-Based Cryptography has a well-known disadvantage. If the PKG is corrupted and PKG's secret key is revealed, all messages and secret

keys are compromised. Our proposed scheme provides password secrecy even when PKG is attacked. A cross-platform identity-based system using Webassembly is suggested in [42]. They recommend an efficent library, called CryptID, which is an opensource IBC implementation for desktops, mobiles, and Internet of Things (IoT) platforms.

1.1. Our contribution. In this paper, an identity-based password registration scheme is proposed, which aims to register users to service providers. Identity-Based Cryptography is well applicable whenever unique identification data can be assigned to an entity in a controlled way. In an enterprise environment for example, after the employees provide their personal data they are delegated a unique enterprise email address. Another example can be when we buy a car, as the owner of the car is assigned a number-plate. The importance of registration is shown by the fact that in modern cars, manufacturers provide the possibility to connect vehicles to the owner through a user account. These accounts also require the user to register a traditional password.

In our case, Identity-Based Cryptography is only used for password registration, where the master secret key is changed daily. This way, the new entrant receives a short-lived secret key from the PKG server, thus eliminating vulnerability of the secret key in Identity-Based Cryptography. Corruption of the secret key does not result in the change of the public key, with new system parameters new secret keys are generated.

We assume that the server has an extra secret key besides the identity-based key pair. This secret key ensures secrecy and prevents the attacker from accessing the password information from the earlier registration even when the PKG becomes corrupt. In this system, PKG may be distributed, which can further increase security and applying the scheme can suit distributed systems.

In our scheme, a device (e.g., smart card) or an application is required, which generates the salt value. It also checks the password which is chosen by the user. We assume that the password policy, which is demanded, is applied on the client side (such as JavaScript API or other application). This device is used by the user only once during the process of registration. During transmission, mutual authentication of participants and confidentiality of the password and the salt are ensured by applying the temporary identity-based key pair. The server calculates the result of the bilinear map of the password and the salt, and stores the calculated value along with the salt.

Our registration scheme can be applied for standard user login applications, the server receives the password in the usual way, then performs the bilinear

mapping with the salt and compares it with the stored value. Our scheme also fits to those authentication processes, when the bilinear mapping is performed on the client side, and the value is sent to the server for verification. The proposed protocol is suitable for the registration phase of the two-factor or one-factor password-based authentication process, depending on whether the device is used for authentication or not.

Note that our proposed registration is also fully blind, as users' passwords and temporary PKG secret key are not known by the server during the registration and subsequent authentication.

In contrast to traditional registration solutions, our solution does not require a TLS channel and can also omit the associated certificate management, which can be efficiently implemented in a corporate or educational institution. According to our implementation, our protocol is more cost-effective than the above-mentioned TLS-based and the other blind solutions ([24], [26]). It is not necessary to manage certificates or execute costly cryptographic primitives (digital signature).

For password verification and storage, bilinear mapping is used, which meets the requirements of password storage (one-way function). In addition, the bilinear mapping applied for password storage is a "slow" function and it can be extended for multi-rounds.

The registration we recommend is flexible, which is optimal for SSO and Kerberos, but it is also suitable for systems where different passwords must be applied for each service. The bilinear map of the password and the salt can be used as a long-lived symmetric key and applied for entity authentication or session key generation.

Comparison	Certificates	Blind	Interactions	Online
BPR - two server	yes	yes	3	yes
BPR - VPAKE	yes	yes	3	yes
TLS-based	yes	no	4	yes
OPAQUE (offline)	no	yes	2	no
OPAQUE (PKI-based)	yes	yes	3	yes
Our proposition	no	yes	2	yes

We have also formalized the security analysis of the registration protocol. Unlike other schemes ([24], [26]) besides the password hashing scheme we also consider the interactions, when the password information is sent securely. Consequently, we prove that our solution is secure against online attacks as well. We introduce the definition for a secure password registration scheme, provide an adversarial model and show that our scheme is provably secure. Security of

the proposed registration protocol is based on the assumptions that solving the Bilinear Diffie–Hellman problem is computationally infeasible the bilinear map is a one-way function and Mac is existentially unforgeable under an adaptive chosen-message attack, where the bilinear map is considered in the generic bilinear group model and the hash functions are supposed as random oracles.

Comparing the offline part of our scheme to [24] and [26], our protocol is still resistant against offline attacks even when the server is corrupted and the client is weakly corrupted.

1.2. Outline of article. In Section 2, we describe the necessary preliminaries and demonstrate the steps of the protocol between a user and a server. In Section 3, we present a security analysis, which includes the security model with the security requirements. We also formalize the adversarial model, and security proofs. Section 4 contains implementation issues that is followed by our conclusion in Section 5.

2. Certificate-less secure blind registration protocol

In this section, we introduce a password registration protocol with password and salt confirmation, i.e., the client is able to confirm that the server knows the map of the correct password and the salt.

The protocol consists of a Setup and a Registration phase. During the Setup system parameters and keys are generated for the participants, during the Registration phase the client sends its password information to the server and confirms that the server has received the verification value. The protocol fulfils all the necessary requirements, such as password secrecy, mutual authentication and resistance against offline attacks.

2.1. Preliminaries. Let us review the definition of the admissible bilinear map [7].

Definition 1. Let $\mathbb{G}$ additive and $\mathbb{G}_{\mathbb{T}}$ multiplicative be two groups of order q for some prime q. A map $\hat{e} : \mathbb{G} \times \mathbb{G} \to \mathbb{G}_{\mathbb{T}}$ is an admissible bilinear map if satisfies the following properties:

(1) Bilinear: We say that a map $\hat{e} : \mathbb{G} \times \mathbb{G} \to \mathbb{G}_{\mathbb{T}}$ is bilinear if $\hat{e}(aP, bQ) = \hat{e}(P, Q)^{ab}$ for all $P, Q \in \mathbb{G}$ and all $a, b \in \mathbb{Z}$.

(2) Non-degenerate: The map does not send all pairs in $\mathbb{G} \times \mathbb{G}$ to the identity in $\mathbb{G}_{\mathbb{T}}$. Since $\mathbb{G}$, $\mathbb{G}_{\mathbb{T}}$ are groups of prime order, if P is a generator of $\mathbb{G}$ then $\hat{e}(P, P)$ is a generator of $\mathbb{G}_{\mathbb{T}}$.

(3) Computable: There is an efficient algorithm to compute $\hat{e}(P, Q)$ for any $P, Q \in \mathbb{G}$.

The Weil and Tate pairings prove the existence of such constructions. Typically, $\mathbb{G}$ is a group of points of an elliptic curve and $\mathbb{G}_{\mathbb{T}}$ is the multiplicative group of a finite field. The Bilinear Diffie–Hellman Problem is strongly related to the bilinear map.

Definition 2 (Bilinear Diffie–Hellman Problem). Let $\hat{e} : \mathbb{G} \times \mathbb{G} \to \mathbb{G}_{\mathbb{T}}$ be a bilinear map on $(\mathbb{G}, \mathbb{G}_{\mathbb{T}})$. Given (P, aP, bP, cP) for some $a, b, c \in \mathbb{Z}_q^*$, compute $\hat{e}(P, P)^{abc}$.

Definition 3 (One-way Pairing [16]). Let $\hat{e} : \mathbb{G} \times \mathbb{G} \to \mathbb{G}_{\mathbb{T}}$ be a bilinear map on $(\mathbb{G}, \mathbb{G}_{\mathbb{T}})$. We say that $\hat{e}$ is a one-way pairing if for any polynomial time (in a security parameter κ) algorithm $\mathcal{A}$ that takes as input $G \in \mathbb{G}$ and $g \in \mathbb{G}_{\mathbb{T}}$ and produces as output an element of $\mathbb{G}$ the probability $\Pr[\hat{e}(G, \mathcal{A}(G, g)) = g]$ is negligible. The probability is taken over the possible values of G and g.

Definition 4 (Computational Diffie–Hellman Problem). Let q be a prime $a, b \in \mathbb{Z}_q^*$ and $\mathbb{G}$ be a multiplicative group of order q. For a given (g, g^a, g^b) (with $g \in \mathbb{G}$) compute g^{ab}.

Lemma 1. *Let $\hat{e} : \mathbb{G} \times \mathbb{G} \to \mathbb{G}_T$ be a bilinear map defined as in Definition 1. Let $\langle G \rangle = \mathbb{G}$, where $\langle G \rangle$ denotes the subgroup generated by G and $\langle g \rangle = \mathbb{G}_T$ be any elements such that $\hat{e}(G, G) = g$. If the Computational Diffie–Hellman (CDH) Problem is infeasible for $g, g^a, g^b \in \mathbb{G}_T$ with any $a, b \in \mathbb{Z}_q^*$ then $\hat{e}$ is a one-way pairing.*

Proof. Let G be a generator of $\mathbb{G}$ and $g, g^a, g^b \in \mathbb{G}$ be given as was described in the lemma. Suppose now that $\hat{e}$ is not one-way, thus we can find (in polynomial time) $G_1, G_2 \in \mathbb{G}$ such that $\hat{e}(G, G_1) = g^a$ and $\hat{e}(G, G_2) = g^b$. Since G is a generator in $\mathbb{G}$, we can write $G_1 = xG$ and $G_2 = yG$ with some integers x, y. However since $\hat{e}$ is bilinear, $x = a$ and $y = b$ must be true. Thus $\hat{e}(G_1, G_2) = \hat{e}(aG, bG) = \hat{e}(G, G)^{ab} = g^{ab}$ which contradicts the infeasibility of the CDH problem. □

Since our password hashing scheme uses bilinear pairings on elliptic curves, we need an efficient way to map passwords first into $\mathbb{Z}_p$ (with p given in the Appendix), then these elements of $\mathbb{Z}_p$ into a point on the curve. Mapping passwords into $\mathbb{Z}_p$ can be done easily by concatenating the ASCII value of each character, then taking the result mod p. For mapping messages (passwords) from $\mathbb{Z}_p$

to an elliptic curve over $\mathbb{Z}_p$ it is desirable to choose an "almost always" injective encoding. A similarly good property would be that the mapping is efficiently computable and reversible, so we can easily obtain both the encoded message from an (almost) arbitrary element of $\mathbb{Z}_p$ and also our initial message (password) from (almost) any given curve point. Finally for cryptographic algorithms it is also desirable that the mapping is surjective, since otherwise our possible message (password) space is unnecessarily limited. Unfortunately general encodings which fulfills these requirements are very scarce, however in [15] the authors provide such a mapping.

Since the results in [15] are formulated in a more general form, stating them here is out of the scope of the present paper, thus we only provide their main result in a simplified form and for the exact technical details we refer to [15] Sections 1 and 2. From this point let q be an odd prime congruent to 3 modulo 4, g a positive integer and $E : y^2 = x^{2g+1} + a_1 x^{2g-1} + \cdots + a_g x$ an elliptic curve over $\mathbb{Z}_q$. Denote by $\left(\frac{\cdot}{q}\right)$ and $\sqrt{\cdot}$ the Legendre symbol and the square root over $\mathbb{Z}_q$. Finally let $\varepsilon(x) = \left(\frac{f(x)}{q}\right)$. The proposed encoding is

$$\begin{aligned} \mathrm{tr} : \mathbb{Z}_q &\longrightarrow E \\ x &\mapsto \left(\varepsilon(x) \cdot x,\, \varepsilon(x)\sqrt{\varepsilon(x) \cdot f(x)}\right) \end{aligned}$$

Since $\varepsilon(x) \in \{-1, 0, 1\}$ and for every $x \in \mathbb{Z}_q$ $f(-x) = -f(x)$ then it is clear that $(\varepsilon(x)\sqrt{\varepsilon(x) \cdot f(x)})^2 = f(\varepsilon(x) \cdot x)$ holds. Let T denote the set of roots in $\mathbb{Z}_q$ of $f(x)$ and W the set consisting of the points of E in the form of $(x, 0)$, and the point at infinity. In [15] the authors proved that the encoding tr induces a bijection $\mathbb{Z}_q \setminus T \longrightarrow E \setminus W$. In our case the elliptic curve is $E : y^2 = x^3 + x$ over $\mathbb{Z}_q$, where the prime q is given in the Appendix. It can be verified that both requirements stated in [15] are fulfilled thus the theorem holds for E. Thus based on the results stated in [15] tr is "almost always" a bijection.

It can also be seen that this mapping is efficiently reversible, because for any point $P = (x, y) \in E$ the numerical value of the original message (password) is either x or $-x$.

The only thing what remains is to prove that tr can be efficiently computed. In Section 3.2 of [15] the authors provide an algorithm (Algorithm 1. in the cited paper) to calculate tr without calculating the Legendre symbol and the square root by using only a single exponentiation and several multiplications. Thus it can be concluded that tr satisfies every requirement stated above.

2.2. Description of cryptographic primitives. Before we prove the security properties of the proposed protocol, the necessary security assumptions for the basic primitives are detailed.

Definition 5. A message authentication code (Mac) is a tuple of polynomial-time algorithms $(\text{Key}, \text{Mac}, \text{Ver})$ such that:

(1) The key-generation algorithm Key takes as input the security parameter 1^κ and outputs a key K with $|K| \geq \kappa$. Key is probabilistic.

(2) The tag-generation algorithm Mac takes as input a key K and a message $m \in \{0,1\}^*$, and outputs a tag t. We write this as $t := \text{Mac}_K(m)$. We assume that Mac is deterministic.

(3) The verification algorithm Ver takes as input a key K, a message m, and a tag t. It outputs a bit b, with $b = 1$ meaning valid and $b = 0$ meaning invalid. We assume without loss of generality that Ver is deterministic, and so write this as $b := \text{Ver}(m, t)$.

Consider the experiment for a message authentication code $(\text{Key}, \text{Mac}, \text{Ver})$, an adversary $\mathcal{A}$, and security parameter κ, as follows. The message authentication experiment $\text{Exp}^{eforge}_{\text{Mac}}(\mathcal{A})$:

(1) A random key K is generated by running $\text{Key}(1^\kappa)$.

(2) The adversary $\mathcal{A}$ is given input 1^κ and oracle access to $\text{Mac}_K(.)$. The adversary eventually outputs a pair (m, t).

(3) The output of the experiment is defined to be 1 if and only if $\text{Ver}_K(m, t) = 1$ and m was never asked from the oracle $\text{Mac}_K(.)$ before.

Definition 6. A message authentication code $(\text{Key}, \text{Mac}, \text{Ver})$ is *existentially unforgeable under an adaptive chosen-message attack*, if for all probabilistic polynomial-time adversaries $\mathcal{A}$, $\Pr[\text{Exp}^{eforge}_{\text{Mac}}(\mathcal{A}) = 1]$ is negligible. (Note that the attacker cannot choose Key.)

A function μ is *negligible* if for every positive polynomial $p(.)$ there exists an N such that for all integers $n > N$ it holds that $\mu(n) < \frac{1}{p(n)}$.

2.3. Setup. We differentiate two participants: A *client* (C) requesting registration and a *server* (S). During the setup, all system parameters and keys are generated for the identity-based environment. A Private Key Generator (PKG) generates the identity-based secret keys for the participants. We denote the set of all binary strings of finite length by $\{0,1\}^*$. A security parameter k and the descriptions of groups $\mathbb{G}, \mathbb{G}_\mathbb{T}$ of order q are given, where q is a large prime, and the bilinear map $\hat{e} : \mathbb{G} \times \mathbb{G} \to \mathbb{G}_\mathbb{T}$ and the function tr from Section 2.1 are made

publicly available. The descriptions include polynomial time (in k) algorithms to compute the group operations in $\mathbb{G}, \mathbb{G}_{\mathbb{T}}$ as well as $\hat{e}$.

We build up the identity-based environment as follows. Let P be a generator of $\mathbb{G}$. Choose a random $\alpha \in \mathbb{Z}_q^*$ and generate parameters $P, \alpha P$. The master secret key for the system is α and the system parameters par are given by par $= (\mathbb{G}, \mathbb{G}_{\mathbb{T}}, \hat{e}, \mathrm{tr}, P, \alpha P, H, \mathrm{Mac})$, where $H : \{0,1\}^* \to \{0,1\}^{\iota}$ is a cryptographic hash function, and ι is the size of the long-lived key being exchanged. Mac : $\{0,1\}^* \to \{0,1\}^{\nu}$ is a Message Authentication Code function, where ν, ι are not necessarily different. System parameters par are publicly known.

Identities (e.g., e-mail address) denoted by ID_C and ID_S are generated for the participants. Public keys are derived, i.e., $PK_C = Q_C = \mathrm{tr}(ID_C)$ and $PK_S = Q_S = \mathrm{tr}(ID_S)$. The PKG calculates the participants' secret keys $SK_C = \alpha Q_C$ and $SK_S = \alpha Q_S$. The server randomly generates $x \in \mathbb{Z}_q^*$, and sends $(Q_S, x\alpha P)$ to the PKG.

2.4. Registration phase. In the registration phase, a client (C) registers to the server (S) and sends the chosen salted password securely with the salt. An identity-based setting is applied, all the benefits of Identity-Based Cryptography are utilized, i.e., we leave the chain of trust (long certificate chains) and the Public Key Infrastructure. We take advantages of the characteristics of the bilinear map $\hat{e}$ including bilinearity and one-way function. In this phase, mutual authentication between the client and the server is processed. Moreover, at the end of this phase S stores the identity of C, the salt and the salted password securely received from C. A long-lived key K is also exchanged. Figure 1 shows the process of registration between the client and the server.

During the setup, system parameters including $P, \alpha P$, the public keys Q_C, Q_S and $(Q_S, x\alpha P)$ are made public.

- C chooses a random value $z \in \mathbb{Z}_q^*$ which serves as a salt and a password psw. C computes the encoding from Section 2.1 to get $R = \mathrm{tr}(psw)$. Subsequently, C creates a message $m = \hat{e}(Q_S, zx\alpha P + \alpha Q_C) \cdot \hat{e}(zP, R)$ and a verification value $V = H(\hat{e}(Q_S, zx\alpha P + \alpha Q_C)||K)$, where $||$ denotes the concatenation of the messages. $K = H(\hat{e}(zP, R))$ serves as a key that is transferred with the server. Values Q_C , m, V and zP are sent to server S. Value zP is the salt and stored in the server's password database. The salt is needed for S to verify the validity of $\hat{e}(zP, R)$. Authentications of the client and the message are based on the short-lived identity-based secret key αQ_C and the correctness of V,

- After receiving the registration request message (Q_C, m, V, zP) from C, S computes $\overline{K} = m \cdot \hat{e}(\alpha Q_S, xzP + Q_C)$, where αQ_S is S's short-lived secret key. Then S computes the value $V' = H(\hat{e}(\alpha Q_S, xzP + Q_C)||K')$, where $K' = H(\overline{K})$ and checks whether $V = V'$ holds. If they are equal, then S is sure of the authenticity of the client and the validity of the other values $\overline{K}$ and zP. S stores $Q_C, \hat{e}(zP, R)$ and zP in the database. Thereafter S generates a random value $r \in \mathbb{Z}_q^*$ and computes a Mac value $\mathrm{Mac}_{K'}(r)$. S sends a response $(Q_S, \mathrm{Mac}_{K'}(r), r)$,
- C receives the S message and calculates the Mac value applying $K = H(\hat{e}(zP, R))$. If $\mathrm{Mac}_K(r)$ is correct, then C is successfully authenticated by S, and C also confirms that server S knows $\hat{e}(zP, R)$.

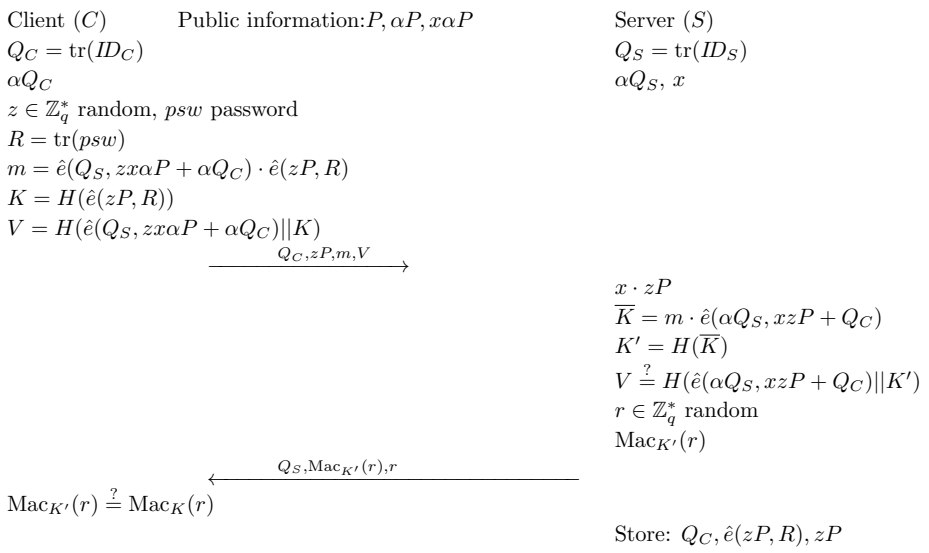

Figure 1. Password registration protocol

In the proposed password registration protocol, the client chooses a password (psw) and the salt (z) is generated. The bilinear map – a one-way map – of the password and the salt ($\hat{e}(zP, R)$) is securely sent and stored on server side. The authenticity of message $\hat{e}(zP, R)$ and $z\alpha P$ is verified by the server as follows. The identity of the sender is verified by calculating $\hat{e}(zP, R)$ from message m and xzP, applying secret server key αQ_S. Data integrity of the messages is verified by checking the correctness of V. Confidentiality of $\hat{e}(zP, R)$ is assured. Value $\hat{e}(Q_S, zx\alpha P + \alpha Q_C)$ randomized by x is multiplied by $\hat{e}(zP, R)$.

The client is able to confirm that the server has received and stored the correct password and salt information by checking the correctness of the $\mathrm{Mac}_{K'}(r)$ value. In order to prevent replay attacks, value r ensures that the Mac value is fresh for every registration. Considering the time complexity, this password registration is very efficient, since there is only one bilinear map calculation on user side and one bilinear mapping on server side besides the Mac, hash operations and point multiplication by a scalar.

3. Security analysis

In this section first we define the security model, then we present the definition for a secure password registration protocol. Finally, we state that our proposed scheme is a secure password registration protocol, which resists online (e.g., impersonation attack, password compromising) and offline attacks (e.g., dictionary attacks, rainbow tables).

We review these security requirements informally.

For a secure password registration protocol, basic requirements are mutual authentication of the participants, password secrecy during transmission and resistance against offline attacks. Secure mutual authentication of participants prevents adversaries to impersonate a legal user or server. During the password registration, the newly generated password is confidential, an adversary should not have any information about it. It is essential that password information should be stored on the server side in a way that it should be secure against offline attacks. By the end of the protocol, the client is able to verify that the server knows and stores the proper password information.

3.1. Security model. In 1993 M. Bellare and P. Rogaway proposed an indistinguishability-based model (see [1], [2]). The basic concept of this security model is applied for password registration protocols [26] and [25].

In [25] authors consider the dictionary attack resistance property, i.e., server learns nothing about the password in the verifier. Passive attacks are considered in which the adversary must not be able to retrieve the password from the password verifier faster than with a brute-force attack. In their security model three oracles are listed. The `Execute(C,S)` oracle models a passive attack that executes the protocol. It returns the protocol transcript and the state of the server instance S. Oracle `Send`(C, S, m) models an active attack that sends message m from client instance C, to server instance S. It returns the server's answer m if there exists any. Oracle `Finalise(C,S,psw)` takes a client, server pair (C, S)

and a password *psw* as input, and returns 1 iff there exists a server instance that accepted password verifier information for *psw*.

In [26] the adversary has access to oracles `Setup`, `Send`, `Execute` and `Corrupt` for interaction with the protocol participants. *Password blindness* is defined by a distinguishing experiment where the attacker, after interacting with the oracles, outputs a challenge comprising of two passwords, two clients, and a pair of servers. After a random assignment of passwords to the two clients, the adversary interacts with the oracles again and has to decide which client possesses which password.

We provide a security model that considers online attacks in addition to resistance against offline attacks. Our proposed model takes the whole registration process into account unlike [26] and [25]. We have regard to all communication messages between the client and the server as well. Hence mutual authentication of the participants and password secrecy are also studied during transmission. We define security goals for password registration protocols that consider the whole registration process assuming the minimum requirements.

We apply the generic bilinear group (GBG) model, which was introduced by D. Boneh and et. al. in [6]. In the GBG model, two random encodings Ω_0, Ω_1 of the additive group $\mathbb{Z}_q^+$ are considered and injective maps $\Omega_0, \Omega_1 : \mathbb{Z}_q^+ \to \Theta$ are employed, where Θ is a bitstring set and $|\Theta| = q$. We write that $\mathbb{G} = \{\Omega_0(x) | x \in \mathbb{Z}_q^+\}$ and $\mathbb{G}_\mathbb{T} = \{\Omega_1(x) | x \in \mathbb{Z}_q^+\}$. In the GBG model, an oracle executes the group operation and takes two encodings of group elements as input, then outputs an encoding of a third element. The group is allowed for a pairing operation, which is an additional oracle. We give oracles $\mathbb{Q}_\mathbb{G}$, $\mathbb{Q}_{\mathbb{G}_\mathbb{T}}$ that execute the group operation on $\mathbb{G}, \mathbb{G}_\mathbb{T}$ and an oracle $\mathbb{Q}_\mathbb{P}$ to calculate a bilinear map $\hat{e} : \mathbb{G} \times \mathbb{G} \to \mathbb{G}_\mathbb{T}$. For any operations on groups, the adversary must issue the associated group queries to the polynomial time adversary $\mathcal{F}$ to get the results.

For any $a, b \in \mathbb{Z}_q$, queries $\mathbb{Q}_\mathbb{G}$, $\mathbb{Q}_{\mathbb{G}_\mathbb{T}}$ and $\mathbb{Q}_\mathbb{P}$ possess the following properties:

- $\mathbb{Q}_\mathbb{G}(\Omega_0(a), \Omega_0(b)) \to \Omega_0(a + b \mod q)$,
- $\mathbb{Q}_{\mathbb{G}_\mathbb{T}}(\Omega_1(a), \Omega_1(b)) \to \Omega_1(a + b \mod q)$,
- $\mathbb{Q}_\mathbb{P}(\Omega_0(a), \Omega_0(b)) \to \Omega_1(ab \mod q)$.

We detail the security model. We will denote 1^κ the string consisting of κ consecutive 1 bits, where $\kappa \in \mathbb{N}$ is a security parameter. *ID* is the union of the finite, disjoint, nonempty sets Client $= \{1, 2, \ldots, \mathrm{T}_1(\kappa)\}$ and Server $= \{1, 2, \ldots, \mathrm{T}_2(\kappa)\}$, where $T_i(\kappa)$, $i \in 1, 2$ is a bound on the number of participants in κ for some polynomial function T_i. Each participant is modelled by an oracle $\prod_{I,J}^l$, which simulates a participant I executing a protocol session in the belief that it is communicating with another participant J for the lth time, where $l \in \{1, \ldots, T_3(\kappa)\}$

for some polynomial function T_3. Oracles keep transcripts, which contain all messages they have sent and received and the queries they have answered. Each participant holds an identity-based key pair generated by the *PKG oracle* during the setup. The final output of the registration is the password information denoted by psw_data_C that is necessary to verify the identity of C.

A registration protocol run is considered to be successful, if the participants confirm the password information. Participants' oracle instances are terminated when they finish a protocol run. An oracle can be in state accepted before it is terminated. The server is in state accepted, if it decides to store the password information psw_data_C. The client is in state accepted, if it confirms that server stores the correct psw_data_C, after receipt of properly formulated messages.

We give the definition of a registration protocol as follows. In general a protocol determines what step a participant instance should take as a response to the adversarial message.

Definition 7. A registration protocol is a pair $P = (\Pi, \Gamma)$ of probabilistic polynomial time (in the security parameter κ) computable functions, where Π specifies how (honest) players behave and Γ generates key pairs for the participants.

Π takes as input:

κ:	the security parameter;
I:	identity of the sender;
J:	identity of the intended recipient;
pk_I, sk_I:	identity based key pair of I;
pk_J:	identity based public key of J;
$tran$:	ordered set of messages transmitted and received by I in this run of the protocol;

$\Pi(\kappa, I, J, pk_I, sk_I, pk_J, tran)$ outputs a triple (m, δ, η), where:

$m \in \{0,1\}^* \cup \{*\}$:	the next message to be sent from I to J ($*$ indicates no message is sent);
$\delta \in \{Accept, Reject, *\}$:	$C's$ current decision ($*$ indicates no decision yet reached);
$\eta \in \{psw\_data_C, *\}$:	client's password information stored by the server, ($*$ indicates no password information is stored);

3.2. Adversarial model. We assume that the adversary is $\mathcal{A} \notin ID$, i.e., neither a user nor a server. $\mathcal{A}$ is a probabilistic polynomial time Turing Machine with an access to the participants' oracles, i.e., it has a query tape where oracle queries

and their answers are written. $\mathcal{A}$ is able to relay, modify, delay or delete messages. $\mathcal{A}$ is allowed to make the following queries that model adversarial attacks.

`Send`$(\prod^i_{I,J}, M)$**:** This oracle query models an active attack. $\mathcal{A}$ sends the message M to oracle $\prod^i_{I,J}$ that returns a message m, which is sent by the user instance as a response to M. Besides m oracle $\prod^i_{I,J}$ also provides information whether the oracle is in state (δ) **Accepted**, **Rejected** or $*$. The query enables $\mathcal{A}$ to initiate a protocol run between participants I and J by query `Send`$(\prod^i_{I,J}, \lambda)$.

`Corrupt`$(\prod^i_{I,J})$**:** This oracle query models the corruption of participant I. This oracle query models an adversary hacks into the machine. Replying to this oracle query a participant oracle $\prod^i_{I,J}$ provides information about I's asymmetric secret keys and state, i.e., all the values calculated and stored by participant I. If I is a server, then both its secret key and psw_data_C are returned. If I is a client, the password itself and the salt are given as well.

`Reveal`$(\prod^i_{I,J})$**:** This models an insecure usage of a password. If oracle $\prod^i_{I,J}$ is in state accepted, holding a password psw, then this query returns psw to $\mathcal{A}$. This query models the attacks, when the adversary persuades a participant to leak the password, e.g., via a social engineering attack.

`Test`$(\prod^i_{I,J})$**:** This oracle query models the semantic security of the password. It is allowed to be asked only once in a protocol run. If participant I is in state accepted, then a coin b is flipped. If $b = 1$, then psw is returned to the adversary, if $b = 0$, then a random value from the distribution of the password is returned.

We define $\mathcal{A}$'s advantage, the probability that $\mathcal{A}$ can distinguish the password held by the queried oracle from a random string, as follows:

$$Adv^{\mathcal{A}}(\kappa) = |\Pr[\text{guess correct}] - 1/2|.$$

`Execute`(C, S)**:** This oracle models a passive attack. It takes new unopened client and server instances and proceeds honest executions of the protocol. If there is a record for client C on server S then it aborts, otherwise it outputs the transcript of the protocol and S's states after the execution, i.e., all values including password verification information and the salt are stored.

`Finalise`(C, S, psw)**:** This oracle query models the verification of a password psw. Takes a client, server pair (C, S) and a password psw as input, and returns 1 iff there exists a server instance that is in state accepted and stores (C, psw_data_C), where psw_data_C is the verification data of the input psw. We assume that no `Send` was queried for (C, S). Otherwise, returns 0.

An oracle is **opened** or **corrupted**, if it has answered a query $\texttt{Reveal}(\prod_{I,J}^{i})$ or $\texttt{Corrupt}(\prod_{I,J}^{i})$, respectively. We differentiate strong and weak corruption models. In the case of the weak corruption model only the asymmetric keys are transferred, the adversary does not completely compromise the machine. Other values generated and stored during the protocol run are not revealed. A model is called strong corruption model if asymmetric keys and the state (including the password) are also revealed. This is the case when the state is revealed via a malware installed on the machine. Applying these secret values the adversary is able to calculate messages that might be sent to a participant oracle with query $\texttt{Send}(\prod_{I,J}^{i}, M)$.

The output of the oracle `Execute` with query `Finalise` makes it possible to model dictionary-like attacks. Oracle `Finalise` models the attack, when the adversary verifies a client's password stored on server side. We emphasize that `Send` is not queried, since a passive attack is formalized. This refers to the attack when an adversary has access to the transcripts of the protocol and the password database or files. Oracle `Execute` is queried to model the generation of transcript and password data. We consider honest executions, but participants might be corrupted. While modelling offline attacks we assume that the client is weakly corrupted. We note that the success of a password extraction via an online attack is measured by oracle `Test`.

There are concurrent and non-concurrent security models. The concurrent model assumes that several copies of the protocol can be processed concurrently, i.e., several instances of the same participant can be active simultaneously. For the non-concurrent model at most one participant instance can be active per participant.

During the attack an experiment of running a protocol $P = (\Pi, \Gamma)$ in the presence of an adversary $\mathcal{A}$ is examined. First Γ is run to generate keys and system parameters for all participants, then $\mathcal{A}$ initializes all participant oracles and asks polynomially number of oracle queries including $\texttt{Send}(\prod_{I,J}^{i}, M)$, $\texttt{Reveal}(\prod_{I,J}^{i})$, $\texttt{Corrupt}(\prod_{I,J}^{i})$ to the participant oracles and queries $\texttt{Execute}(C, S)$ and $\texttt{Finalise}\ (C, S, psw)$. Finally $\mathcal{A}$ asks a $\texttt{Test}(\prod_{I,J}^{i})$ query.

In order to give the definition of a secure registration protocol, we need to review the definition of conversation and matching conversation from [5]. They were also formalized in [1].

Matching conversation formalizes real-time communication between entities I and J, it is necessary to define authentication property of a protocol. We give the definition of the event $\texttt{No-Matching}^{\mathcal{A}}(\kappa)$ where definition is given in [5].

Definition 8. Consider an adversary $\mathcal{A}$ and a participant oracle $\prod_{I,J}^{s}$. We define the *conversation* $C_{I,J}^{s}$ of $\prod_{I,J}^{s}$ as a sequence of

$$C_{I,J}^{s} = (\tau_1, \alpha_1, \beta_1), (\tau_2, \alpha_2, \beta_2), \ldots, (\tau_m, \alpha_m, \beta_m),$$

where τ_i denotes the time when oracle query α_i and oracle reply β_i are given $(i = 1, \ldots, m)$.

Naturally $\tau_i > \tau_j$, iff $i > j$. $\mathcal{A}$ terminates after receiving the reply β_m, i.e., does not ask more oracle queries. During a conversation the initiator and responder oracles are differentiated. $\prod_{I,J}^{s}$ is an initiator oracle if $\alpha_1 = \kappa$, otherwise it is a responder. Consider the definition for matching conversation when the number of protocol flows is odd.

Definition 9. Running protocol P in the presence of $\mathcal{A}$, we assume that the number of flows is $2\rho - 1$, for a positive integer ρ, $\prod_{I,J}^{s}$ is an initiator and $\prod_{J,I}^{t}$ is a responder oracle that engage in conversations C and C', respectively.

C' is a *matching conversation* to C, if there exist $\tau_0 < \tau_1 < \cdots < \tau_{R-1}$ and $\alpha_1, \beta_1, \ldots, \beta_{\rho-1}, \alpha_\rho$ such that C is prefixed by:

$$(\tau_0, \lambda, \alpha_1), (\tau_2, \beta_1, \alpha_2), \ldots, (\tau_{2\rho-2}, \beta_{\rho-1}, \alpha_\rho),$$

and C' is prefixed by:

$$(\tau_1, \alpha_1, \beta_1), (\tau_3, \alpha_2, \beta_2), \ldots, (\tau_{2\rho-3}, \alpha_{\rho-1}, \beta_{\rho-1}).$$

C is a *matching conversation* to C', if there exist $\tau_0 < \tau_1 < \cdots < \tau_R$ and $\alpha_1, \beta_1, \ldots, \beta_{\rho-1}, \alpha_\rho$ such that C' is prefixed by:

$$(\tau_1, \alpha_1, \beta_1), (\tau_3, \alpha_2, \beta_2), \ldots, (\tau_{2\rho-3}, \alpha_{\rho-1}, \beta_{\rho-1}), (\tau_{2\rho-1}, \alpha_\rho, *),$$

and C is prefixed by:

$$(\tau_0, \lambda, \alpha_1), (\tau_2, \beta_1, \alpha_2), \ldots, (\tau_{2\rho-2}, \beta_{\rho-1}, \alpha_\rho).$$

If C is a matching conversation to C' and C' is a matching conversation to C, then $\prod_{I,J}^{s}$ and $\prod_{J,I}^{t}$ are said to have had *matching conversation*.

Matching conversation formalizes real-time communication between entities I and J, it is necessary to define authentication property of a protocol. We give the definition of the event $\texttt{No-Matching}^{\mathcal{A}}(\kappa)$ given in [5].

Definition 10. $\texttt{No-Matching}^{\mathcal{A}}(\kappa)$ denotes an event when in a protocol P in the presence of an adversary $\mathcal{A}$ assuming there exist a participant oracle $\prod_{I,J}^{s}$ which is accepted, but there is no other oracle $\prod_{J,I}^{t}$ having a matching conversation with $\prod_{I,J}^{s}$.

We review the definition of min-entropy for a dictionary from [25]. Passwords are considered as character strings, where the distribution of characters depends on the used character set Σ, character positions and the password string itself.

Definition 11. Let Σ denote a finite character set. Further let D_n be a subset of words of length n over Σ, which is called dictionary. Denote $\mathbb{D}_\Sigma$ the probability distribution of the elements of D_n. Then the min-entropy for D_n is defined as

$$\beta_{D_n} = -\max_{c_0 \ldots c_{n-1} \in D_n} \sum_{i=0}^{n-1} \mathbb{D}_\Sigma(c_i) \log_2 \mathbb{D}_\Sigma(c_i).$$

A password registration scheme is secure if the values stored on the server-side leak as little information as possible on the password, i.e., an attacker can not retrieve the password from the password verification value more efficiently than by performing a brute-force attack over the dictionary.

We give the definition of benign adversary.

Definition 12. An adversary is called *benign* if it is deterministic, and restricts its action to choosing a tuple of oracles containing one client and one server oracle, and then faithfully conveying each flow from one oracle to the other, with the client oracle beginning first.

Definition 13. A protocol is a *secure registration protocol* if

- Online resistance:
  - (1) In the presence of the *benign adversary* the client and the server oracle communicating with the client always accept. The server stores the password verification value confirmed by the client,

  and for every adversary $\mathcal{A}$
  - (2) If there is an uncorrupted client oracle having matching conversations with an uncorrupted server oracle then they always accept. The server stores the password verification value confirmed by the client,
  - (3) For uncorrupted server and client oracles the probability of $\texttt{No-Matching}^{\mathcal{A}}(\kappa)$ is negligible,
  - (4) For the tested oracle $\mathrm{Adv}^{\mathcal{A}}(\kappa)$ is negligible. If it is a client oracle, then it is unopened.

- Offline resistance:
  (5) If for all dictionaries D_n adversary $\mathcal{A}$ generates at most t tuples (C, S, psw), then

$$\Pr[\texttt{Finalise}(C, S, psw) = 1] \leq \frac{t}{2^{\beta_{D_n}} \cdot t_{pre}} + \mu(\kappa),$$

  where $\mu(n)$ is negligible and t_{pre} denotes the computational cost to calculate the input value of the one-way function from the password.

In the definition above only necessary security assumptions are given. According to (3), we assume that participants do not disclose their secret keys in order to assure mutual authentication of the participants. To provide semantic security of the password, it is assumed that the client does not reveal the password. In the case of offline attacks, we assume that besides having access to the transcripts and password information stored on server side, the adversary also gains the secret keys of the server and the client. We do not assume that the server and the client are uncorrupted. We only assume that the client is unopened, and does not reveal the password.

We define two security models. In the case of client-server protocols, clients usually are assumed to be malicious, i.e., they deviate form the steps of the protocol, they apply any type of strategy to attack. The servers providing some service are usually considered to be honest, meaning they do not launch any attack or honest-but-curious, i.e., they initiate only passive attacks, not leaving any trace of the attack. Depending on whether the server is honest or honest-but-curious, we differentiate *honest and honest-but-curious models.* In [26] and [25] honest models are used.

3.3. Security proof. Let's consider the hash functions in the random oracle model and the bilinear maps in the bilinear group model.

Theorem 2. *The proposed password registration protocol is resistant against online attacks in the honest-but-curious model, assuming Mac is* existentially unforgeable under an adaptive chosen-message attack, *solving the* Bilinear Diffie–Hellman problem *is computationally infeasible, moreover the bilinear map is considered in the generic bilinear group model and the hash functions are supposed as random oracles.*

Proof. Proving conditions (1) and (2) of the Definition 13 is trivial, since the steps of the protocol are followed. Let us consider condition (3), which holds if the assumption that the Mac is existentially unforgeable under an adaptive

chosen-message attack and the Bilinear Diffie–Hellman Problem holds. Moreover the hash functions are random oracles. Let us see it in details.

Consider an adversary $\mathcal{A}$ and suppose that $\Pr[\texttt{No-Matching}^{\mathcal{A}}(\kappa)]$ is non-negligible. There are two cases: either the server, or the client oracle is accepted.

Case 1. Let $\mathcal{A}$ `succeeds` denote the event that in $\mathcal{A}$'s experiment there is a server oracle $\prod_{S,C}$ that is *accepted*, but there is no client oracle $\prod_{C,S}$ having matching conversation to $\prod_{S,C}$.

We assume that

$$\Pr[\mathcal{A} \texttt{ succeeds}] = n_S(\kappa),$$

where $n_S(\kappa)$ is non-negligible. We construct a polynomial time adversary $\mathcal{F}$ that for given $a\mathbf{P}, b\mathbf{P}, c\mathbf{P}, \mathbf{P}$ calculates $BDH(a\mathbf{P}, b\mathbf{P}, c\mathbf{P}, \mathbf{P}) = \hat{e}(\mathbf{P}, \mathbf{P})^{abc}$. $\mathcal{F}$ randomly picks $C \in$ Client and $S \in$ Server. Let $\Delta = \{C, S\}$ denote the identities of protocol participants. $\prod_{S,C}$ denotes the server oracle that communicates to the client C. $\mathcal{F}$ also chooses randomly a particular session $t \in \{1, \ldots, T_3(\kappa)\}$. Given security parameter κ, adversary $\mathcal{F}$ sets $par = (\mathbb{G}, \mathbb{G}_{\mathbb{T}}, \mathbb{Q}_{\mathbb{P}}, \mathrm{tr}, \mathbf{P}, c\mathbf{P}, H, \mathrm{Mac})$, for calculating hash values, encodings and bilinear map $\mathcal{F}$ calls hash, tr and $\mathbb{Q}_{\mathbb{P}}$ oracles, respectively. To make the proof easier to follow let $a\mathbf{P}$ denote the value returned by oracle $\mathbb{Q}_{\mathbb{G}}$ as a result of applying group operation a times for any $a \in Z_q$ and the answer of oracle query $\mathbb{Q}_{\mathbb{G}_{\mathbb{T}}}$ for inputs c, d is denoted by $c \cdot d$. $\mathcal{F}$ also sets public keys $Q_S = a\mathbf{P}, Q_C = b\mathbf{P}$, whenever $\mathrm{tr}(ID_S)$ or $\mathrm{tr}(ID_C)$ are asked $\mathcal{F}$ answers $Q_S = a\mathbf{P}, Q_C = b\mathbf{P}$, respectively. $\mathcal{F}$ randomly chooses value $\bar{x} \in \mathbb{Z}_q^*$ and sets $\bar{x}c\mathbf{P}$ as a server public value. Value $\bar{x}c\mathbf{P}$ is sent to oracle PKG. $\mathcal{F}$ runs $\mathcal{A}$ and answers $\mathcal{A}$'s queries as follows.

(1) $\mathcal{F}$ answers H hash and tr encoding oracle queries at random (like a real random oracle would), except if ID_S or ID_C is asked.

(2) $\mathcal{F}$ answers `Corrupt` queries according to Π, reveals secret keys, internal states and secret values. Queries to the current server and the client oracles are refused.

(3) $\mathcal{F}$ answers `Reveal` queries as specified in Π. This query is refused if it is asked from $\prod_{S,C}$, since $\prod_{S,C}$ does not hold the password.

(4) If $\mathcal{A}$ does not involve $\prod_{C,S}$ as a client oracle which communicates to the server oracle $\prod_{S,C}$, then $\mathcal{F}$ gives up. If $\mathcal{A}$ does not invoke $\prod_{C,S}$ as an initiator oracle, then $\mathcal{F}$ gives up. Otherwise $\mathcal{A}$ generates a password psw and a random $\bar{t}$ value, and calculates $z\mathbf{P}$ and $\mathbf{R}$. Eventually $\mathcal{A}$ asks bilinear map oracle queries $\mathbb{Q}_{\mathbb{P}}(.)$ to get $\hat{e}(Q_S, z\bar{x}c\mathbf{P} + cQ_C)$ and $\hat{e}(z\mathbf{P}, \mathbf{R})$, $\mathcal{F}$ answers these queries. $\mathcal{A}$ calculates m, where $m = \hat{e}(Q_S, z\bar{x}c\mathbf{P} + cQ_C) \cdot \hat{e}(z\mathbf{P}, \mathbf{R})$. $\mathcal{A}$ asks

hash oracle query $H(.)$ of $\hat{e}(z\mathbf{P}, \mathbf{R})$ from $\mathcal{F}$ to get key K. If K is a previously used value, then $\mathcal{F}$ gives up. $\mathcal{A}$ asks hash oracle query $H(.)$ to get V, where $V = H(\hat{e}(Q_S, z\bar{x}c\mathbf{P} + cQ_C)||K)$. $\mathcal{F}$ answers the hash query and if V is a previously used value, then $\mathcal{F}$ gives up. $\mathcal{A}$ asks query $\texttt{Send}(\prod_{S,C}, M)$, where

$$M = Q_C||z\mathbf{P}||m||V.$$

Since $\mathcal{F}$ knows $\hat{e}(z\mathbf{P}, \mathbf{R})$, $\mathcal{F}$ calculates K and answers $Q_S||\operatorname{Mac}_K(r)||r)$, where r is random. If some later time $\mathcal{A}$ does not ask the $\texttt{Send}(\prod_{C,S}, Q_S||\operatorname{Mac}_K(r)||r)$, then $\mathcal{F}$ gives up, otherwise $\prod_{S,C}$ gets accepted. $\mathcal{F}$ calculates and outputs $\hat{e}(Q_S, z\bar{x}c\mathbf{P} + cQ_C) \cdot \hat{e}(Q_S, c\mathbf{P})^{-z\bar{x}}) = \hat{e}(\mathbf{P}, \mathbf{P})^{abc}$.

(5) $\mathcal{F}$ answers `Execute` and `Finalise` queries as specified in Π. Query `Finalise` is refused if `Send` was queried before.

Assume that $\mathcal{A}$ is successful, event $\mathcal{A}$ `succeeds` happens with $n_S(\kappa)$ non-negligible probability. We show that $\mathcal{F}$ wins its experiment with non-negligible probability. For the analysis the probability that $\mathcal{F}$ chooses the correct participants Δ, session t and succeeds is:

$$\xi_1(\kappa) = \frac{n_S(\kappa)}{T_1(\kappa)T_2(\kappa)T_3(\kappa)} - \lambda(\kappa),$$

where $\lambda(\kappa)$ denotes the probability that $\mathcal{F}$ previously calculated the flow. $\xi_1(\kappa)$ is the multiplication of probabilities which contains the $n_S(\kappa)$ probability of that $\mathcal{A}$ `succeeds`, $\frac{1}{T_1(\kappa)}, \frac{1}{T_2(\kappa)}, \frac{1}{T_3(\kappa)}$ denote the probability that the correct client, server participants and the appropriate session are chosen. The $\xi_1(\kappa)$ is non-negligible, if $n_S(\kappa)$ is non-negligible, $T_i(\kappa)$ (i=1,...,3) is polynomial in κ and $\lambda(\kappa)$ is negligible. That contradicts the security assumption of the BDH problem, hence $n_S(\kappa)$ must be negligible.

Case 2. Let $\mathcal{A}$ `succeeds` denote the event that in $\mathcal{A}$'s experiment there is a client oracle $\prod_{C,S}$ that is *accepted*, but there is no server oracle $\prod_{S,C}$ having matching conversation to $\prod_{C,S}$. There are two cases: either $\mathcal{F}$ is able to proceed an existential forgery against Mac under an adaptive chosen message attack, or we will show how to construct BDH problem solver $\mathcal{F}$ that uses an adversary $\mathcal{A}$.

- Case 2.1

  We assume that

$$\Pr[\mathcal{A} \texttt{ succeeds}] = n_{C_{2_1}}(\kappa),$$

where $n_{C_{2_1}}(\kappa)$ is non-negligible. We construct a polynomial time adversary $\mathcal{F}$ that is able to proceed an existential forgery against Mac under an adaptive

chosen message attack. $\mathcal{F}$'s task is to generate a valid (m, t) message-tag pair, where m was never asked from the oracle $\mathrm{Mac}_K(.)$. $\mathcal{F}$ picks the protocol participants and a session $t \in \{1, \ldots, T_3(\kappa)\}$, let $\Delta = \{C, S\}$ denote identities. Let $\prod_{C,S}$ denote the client and $\prod_{S,C}$ the server oracle. $\mathcal{F}$ sets $par = (\mathbb{G}, \mathbb{G}_{\mathbb{T}}, \mathbb{Q}_{\mathbb{P}}, \mathrm{tr}, P, \alpha P, H, \mathrm{Mac})$, where α chosen randomly. To make the proof easier to follow let $a\mathbf{P}$ denote the value returned by oracle $\mathbb{Q}_{\mathbb{G}}$ as a result of applying group operation a times for any $a \in Z_q$ and the answer of oracle query $\mathbb{Q}_{\mathbb{G}_{\mathbb{T}}}$ for inputs c, d is denoted by $c \cdot d$. The key generation Γ is simulated as follows. $\mathcal{F}$ sets public keys as $Q_S = \mathrm{tr}(ID_S), Q_C = \mathrm{tr}(ID_S)$, and $\alpha\,\mathrm{tr}(ID_S)$ or $\alpha\,\mathrm{tr}(ID_C)$, respectively. $\mathcal{F}$ answers $\mathcal{A}$'s oracle queries as follows.

(1) $\mathcal{F}$ answers queries to oracles $H(.), \mathrm{tr}(.), \mathbb{Q}_{\mathbb{P}}$, `Corrupt`, `Reveal` in the same way as in Case 1,

(2) $\mathcal{F}$ answers `Send` queries according to Π with the generated random values z, s. If $\mathcal{A}$ does not involve $\prod_{S,C}$ as a server oracle which communicates to the client oracle $\prod_{C,S}$, then $\mathcal{F}$ gives up. If $\mathcal{A}$ does not invoke $\prod_{C,S}$ as an initiator oracle, then $\mathcal{F}$ gives up, otherwise $\mathcal{A}$ asks oracle query $\mathtt{Send}(\prod_{C,S}, \lambda)$. $\mathcal{F}$ responses

$$M_1 = Q_C||zP||m||V$$

with $m = \hat{e}(Q_S, zx\alpha P + \alpha Q_C) \cdot \hat{e}(zP, R)$ and $V = H_2(\hat{e}(Q_S, zx\alpha P + \alpha Q_C)||K)$, where z is random and R is generated with the tr oracle. The αQ_C secret key is calculated by $\mathcal{F}$.
If some later time $\mathcal{A}$ does not ask the $\mathrm{Mac}_K(.)$ oracle queries, then $\mathcal{F}$ gives up. Otherwise $\mathcal{F}$ answers these queries using oracle $\mathrm{Mac}_K(.)$. Eventually $\mathcal{A}$ creates

$$M_2 = Q_S||\bar{t}||\overline{m}$$

and calls $\mathrm{Send}(\prod_{\mathrm{C,S}}, \mathrm{M}_2)$. If $\overline{m}$ was asked to oracle $\mathrm{Mac}_K(.)$ before, then $\mathcal{F}$ gives up. If $\bar{t} \neq \mathrm{Mac}_K(r)$, then $\mathcal{F}$ gives up, otherwise $\prod_{C,S}$ gets accepted. $\mathcal{F}$ responses $(\overline{m}, \bar{t})$ to the challenger. If $\mathcal{A}$ succeeds with non-negligible probability, then $\mathcal{F}$ outputs a valid forgery $(\overline{m}, \bar{t})$, where $\overline{m}$ was never asked to oracle $\mathrm{Mac}_K(.)$ before.
Assume that $\mathcal{A}$ is successful, event $\mathcal{A}$ `succeeds` happens with $n_{C_{2_1}}(\kappa)$ non-negligible probability. Hence following the algorithm above $\mathcal{F}$ calculates a valid $(\overline{m}, \bar{t})$ pair. We show that $\mathcal{F}$ wins its experiment with

non-negligible probability. The probability that $\mathcal{F}$ chooses correct participants Δ, session t and succeeds is

$$\xi_{2_1}(\kappa) = \frac{n_{C_{2_1}}(\kappa)}{T_1(\kappa)T_2(\kappa)T_3(\kappa)} - \lambda(\kappa),$$

where $\lambda(\kappa)$ denotes the probability that $\mathcal{F}$ previously calculated the flow. Since $n_{C_{2_1}}(\kappa)$ is non-negligible, $T_i(\kappa)$ (i=1,...,3) is polynomial in κ and $\lambda(\kappa)$ is negligible thus $\xi_{2_1}(\kappa)$ is non-negligible. That contradicts the security assumption of Mac, hence $n_{C_{2_1}}(\kappa)$ must be negligible.

- Case 2.2

Let $\mathcal{A}$ `succeeds` denote the event that in $\mathcal{A}$'s experiment there is a client oracle $\prod_{C,S}$ that is *accepted*, but there is no server oracle $\prod_{S,C}$ having matching conversation to $\prod_{C,S}$. We assume that

$$\Pr[\mathcal{A} \texttt{ succeeds}] = n_{C_{2_2}}(\kappa),$$

where $n_{C_{2_2}}(\kappa)$ is non-negligible. In this case we can construct a polynomial time adversary that for given $\mathbf{P}, a\mathbf{P}, b\mathbf{P}, c\mathbf{P}$, calculates $\hat{e}(\mathbf{P},\mathbf{P})^{abc}$.

$\mathcal{F}$ picks the protocol participants and a session $t \in \{1,\ldots,T_3(\kappa)\}$, let $\Delta = \{C, S\}$ denote identities. $\prod_{C,S}$ denotes the client and $\prod_{S,C}$ the server oracle. $\mathcal{F}$ sets par $= (\mathbb{G}, \mathbb{G}_\mathbb{T}, \mathbb{Q}_\mathbb{P}, \mathrm{tr}, \mathbf{P}, \mathrm{c}\mathbf{P}, \mathrm{H}, \mathrm{Mac})$. To make the proof easier to follow let $a\mathbf{P}$ denote the value returned by oracle $\mathbb{Q}_\mathbb{G}$ as a result of applying group operation a times for any $a \in Z_q$ and the answer of oracle query $\mathbb{Q}_{\mathbb{G}_\mathbb{T}}$ for inputs c, d is denoted by $c \cdot d$. The key generation Γ is simulated similarly to Case 1., hence $Q_S = a\mathbf{P}$, $Q_C = b\mathbf{P}$ and randomly chooses values $\bar{x}, \bar{z} \in \mathbb{Z}_q^*$ and sets $\bar{x}c\mathbf{P}$ as a server public value and computes $\bar{z}\mathbf{P}$. Value $\bar{x}c\mathbf{P}$ is sent to oracle PKG. $\mathcal{F}$ answers $\mathcal{A}$'s oracle queries as follows.

$\mathcal{F}$ answers queries to oracles $H(.), \mathrm{tr}(.), \mathbb{Q}_\mathbb{P}$, `Corrupt`, `Reveal` in the same way as in Case 1.

$\mathcal{F}$ answers `Send` queries as follows. If $\mathcal{A}$ does not involve $\prod_{S,C}$ as a server oracle which communicates to the client oracle $\prod_{C,S}$ or $\prod_{C,S}$ is not an initiator oracle, then $\mathcal{F}$ gives up. Otherwise $\mathcal{A}$ asks oracle query `Send`$(\prod_{C,S}, \lambda)$. $\mathcal{F}$ responses

$$M_1 = Q_C || \bar{z}\mathbf{P} || m_1 || V_1$$

with $m_1 \in \mathbb{G}_\mathbb{T}$ chosen randomly and V_1 is a fresh random value chosen by the random oracle as a hash value.

$\mathcal{A}$ eventually asks oracle $\mathbb{Q}_\mathbb{P}$ to calculate values $\hat{e}(a\mathbf{P}, \bar{z}\bar{x}c\mathbf{P} + cb\mathbf{P})$ and $\hat{e}(\bar{z}\mathbf{P}, R)$ for some random value R and the hash oracle for $\hat{e}(\bar{z}\mathbf{P}, R)$. If these

oracle queries were asked before, then $\mathcal{F}$ gives up, otherwise answers the queries. $\mathcal{F}$ multiplies $\hat{e}(a\mathbf{P}, \bar{z}\bar{x}c\mathbf{P} + cb\mathbf{P})$ and $\hat{e}(\bar{z}\mathbf{P}, R)$ and verifies whether the result is m_1. If the result is not m_1, then $\mathcal{F}$ gives up. Otherwise if some time later oracle $H(.)$ is asked for $\hat{e}(a\mathbf{P}, \bar{z}\bar{x}c\mathbf{P} + cb\mathbf{P})||H(\hat{e}(\bar{z}\mathbf{P}, R))$, then V_1 is answered. $\mathcal{A}$ generates a random value r_S and calculates $\mathrm{Mac}_K(r_S)$ and asks query $\texttt{Send}(\prod_{C,S}, M_2)$, where

$$M_2 = Q_S||\,\mathrm{Mac}_K(r_S)||r_S.$$

If M_2 is not valid or not asked, then $\mathcal{F}$ gives up, otherwise $\prod_{C,S}$ gets accepted.

Since $\mathcal{A}$ asked $\hat{e}(\bar{z}\mathbf{P}, R)$ from oracle $H(.)$, $\mathcal{F}$ is able to output $m_1 \cdot \hat{e}(\bar{z}\mathbf{P}, R)^{-1} \cdot \hat{e}(a\mathbf{P}, c\mathbf{P})^{-\bar{z}\bar{x}} = \hat{e}(\mathbf{P}, \mathbf{P})^{abc}$.

Assume that $\mathcal{A}$ is successful, event $\mathcal{A}$ `succeeds` happens with $n_{C_{2_2}}(\kappa)$ non-negligible probability. $\mathcal{F}$ outputs the solution of BDHP. We show that $\mathcal{F}$ wins its experiment with non-negligible probability. The probability that $\mathcal{F}$ chooses the correct participants Δ, session t and succeeds is:

$$\xi_3(\kappa) = \frac{n_{C_{2_2}}(\kappa)}{T_1(\kappa)T_2(\kappa)T_3(\kappa)} - \lambda(\kappa)$$

where $\lambda(\kappa)$ is the probability that the flow was already calculated before. Similarly to Case 2.1 $\xi_3(\kappa)$ is non-negligible, if $n_{C_{2_2}}(\kappa)$ is non-negligible, $T_i(\kappa)$ (i=1,...,3) is polynomial in κ. That contradicts the assumption of Bilinear Diffie–Hellman, hence $n_{C_{2_2}}(\kappa)$ must be negligible.

We turn to condition (4). Consider an adversary $\mathcal{A}$ and suppose that $Adv^{\mathcal{A}}(\kappa)$ is non-negligible.

Case 3. Let $\mathcal{A}$ `succeeds` against $\prod^s_{C,S}$ denote the event that $\mathcal{A}$ asks $\texttt{Test}(\prod^s_{C,S})$ query and outputs the correct bit. Hence

$$\Pr[\mathcal{A}\ \texttt{succeeds}] = \frac{1}{2} + n(\kappa),$$

where $n(\kappa)$ is non-negligible.

Let A_κ denote the event that $\mathcal{A}$ picks either a server or a client oracle $\prod^s_{C,S}$ and asks its `Test` query such that oracle $\prod^s_{C,S}$ has had a matching conversation to $\prod^t_{S,C}$.

$$\Pr[\mathcal{A}\ \texttt{succeeds}] = \Pr[\mathcal{A}\ \texttt{succeeds}|A_\kappa]\Pr[A_\kappa] + \Pr[\mathcal{A}\ \texttt{succeeds}|\overline{A}_\kappa]\Pr[\overline{A}_\kappa].$$

According to the previous section $\Pr[\overline{A}_\kappa] = \mu(\kappa)$, where $\mu(\kappa) \in \{n_{C_{2_1}}(\kappa), n_{C_{2_2}}(\kappa), n_S(\kappa)\}$ and $\Pr[A_\kappa] = 1 - \mu(\kappa)$, where $\mu(\kappa)$ is negligible, hence

$$\frac{1}{2} + n(\kappa) \leq \Pr[\mathcal{A} \texttt{ succeeds}|A_\kappa] \Pr[A_\kappa] + \mu(\kappa)$$

and we get

$$\frac{1}{2} + n_1(\kappa) = \Pr[\mathcal{A} \texttt{ succeeds}|A_\kappa],$$

for a non-negligible $n_1(\kappa)$. We have two cases.

Let B_κ denote the event that for given $Q_C, Q_S, x\alpha P, zP, \alpha P, P, m, V$ adversary $\mathcal{A}$ asks K to oracle $H(.)$, where $K = e(zP, R)$ with $R = \mathrm{tr}(psw)$ for password psw. This is the case of 2.2 ($n_{C_{2_2}}(\kappa)$), where we showed how to construct BDH problem solver $\mathcal{F}$ that uses an adversary $\mathcal{A}$. Moreover $\mathcal{F}$ also breaks the one-wayness of the bilinear map given in Definition 3 with $\mathcal{A}$, since R is asked from oracle $\mathbb{Q}_{\mathbb{P}}(.)$.

$$\begin{aligned}\Pr[\mathcal{A} \texttt{ succeeds}|A_\kappa] = {} & \Pr[\mathcal{A} \texttt{ succeeds}|A_\kappa \wedge B_\kappa] \Pr[B_\kappa|A_\kappa] \\ & + \Pr[\mathcal{A} \texttt{ succeeds}|A_\kappa \wedge \overline{B}_\kappa] Pr[\overline{B}_\kappa|A_\kappa].\end{aligned}$$

Since $\Pr[\mathcal{A} \texttt{ succeeds}|A_\kappa \wedge \overline{B}_\kappa] = \frac{1}{2}$,

$$\frac{1}{2} + n_1(\kappa) \leq \Pr[\mathcal{A} \texttt{ succeeds}|A_\kappa \wedge B_\kappa] \Pr[B_\kappa|A_\kappa] + \frac{1}{2},$$

hence $\Pr[B_\kappa|A_\kappa]$ is non-negligible. We construct a polynomial time adversary $\mathcal{F}$ that for given Q_C, Q_S, $x\alpha P$, zP, αP, P, m, V calculates psw.

$$\xi_4(\kappa) = \frac{n_1(\kappa)}{T_1(\kappa)T_2(\kappa)T_3(\kappa)}$$

that is non-negligible if $n_1(\kappa)$ is non-negligible, $T_i(\kappa)$ (i=1,...,3) is polynomial in κ and denotes the same as in Case 2.2. This contradicts to the BDHP assumption, hence $n_1(\kappa)$ and $Adv^{\mathcal{A}}(\kappa)$ must be negligible.

Let see the other case when C_κ denotes the event that $\mathcal{A}$ is able to recover K itself, and thus carries out Mac existential forgery. This is the case of 2.1. Moreover $\mathcal{A}$ also calculates psw having K and zP, i.e., breaks one-wayness of the bilinear map.

$$\begin{aligned}\Pr[\mathcal{A} \texttt{ succeeds}|A_\kappa] = {} & \Pr[\mathcal{A} \texttt{ succeeds}|A_\kappa \wedge C_\kappa] \Pr[C_\kappa|A_\kappa] \\ & + \Pr[\mathcal{A} \texttt{ succeeds}|A_\kappa \wedge \overline{C}_\kappa] \Pr[\overline{C}_\kappa|A_\kappa].\end{aligned}$$

Since $\Pr[\mathcal{A}\ \texttt{succeeds}|A_\kappa \wedge \overline{C}_\kappa] = \frac{1}{2}$,

$$\frac{1}{2} + n_1(\kappa) \leq \Pr[\mathcal{A}\ \texttt{succeeds}|A_\kappa \wedge C_\kappa] \Pr[C_\kappa|A_\kappa] + \frac{1}{2},$$

hence $\Pr[C_\kappa|A_\kappa]$ is non-negligible. $\mathcal{F}$ proceeds Mac existential forgery non-negligibly and also breaks $\mathcal{F}$ one-wayness of the bilinear map with non-negligible probability.

We construct a polynomial time adversary $\mathcal{F}$ that for given Q_C, Q_S, $x\alpha P$, $zP, \alpha P, P, m, V$ calculates K and psw.

$$\xi_5(\kappa) = \frac{n_1(\kappa)}{T_1(\kappa)T_2(\kappa)T_3(\kappa)}$$

that is non-negligible if $n_1(\kappa)$ is non-negligible, $T_i(\kappa)$ (i=1,...,3) is polynomial in κ and denotes the same as in Case 2.1. This contradicts to the Mac or the one-way pairing assumption, hence $n_1(\kappa)$ and $\mathrm{Adv}^{\mathcal{A}}(\kappa)$ must be negligible. □

Theorem 3. *The proposed password registration protocol is resistant against offline attacks in the random oracle model, if the bilinear map is a one-way pairing and the client is weakly corrupted.*

Proof. Let $\mathcal{A}$ `succeeds` against $\prod_{S,C}$ denote the event that $\prod_{S,C}$ is accepted and $\mathcal{A}$ is able to output a valid (S, C, psw) tuple. Hence

$$\Pr[\mathcal{A}\ \texttt{succeeds}] = n_d(\kappa),$$

where $n_d(\kappa)$ is non-negligible. We construct an efficient algorithm that breaks one-wayness of the bilinear map, for given $\mathbf{P}, z\mathbf{P}, \hat{e}(z\mathbf{P}, R)$ outputs R.

We construct a polynomial time adversary $\mathcal{F}$, which picks the protocol participants $\Delta = \{C, S\}$ and a session $s \in \{1, \ldots, T_3(\kappa)\}$. $\mathcal{F}$ sets par $= (\mathbb{G}, \mathbb{G}_{\mathbb{T}}, \mathbb{Q}_{\mathbb{P}}, \mathrm{tr}, \mathbf{P}, \alpha\mathbf{P}, H, \mathrm{Mac})$ and simulates the key generation Γ similarly to Case 1. of the proof of Theorem 2. $\mathcal{F}$ answers $\mathcal{A}$'s oracle queries as follows.

Adversary $\mathcal{F}$ answers $H(.)$ hash oracle query at random. For `Corrupt` query $\mathcal{F}$ answers secret keys of the participant oracles and the state of the server oracle. Adversary $\mathcal{F}$ refuses oracle queries `Reveal` and `Send`.

Adversary $\mathcal{F}$ answers to the `Execute` oracle the transcripts generated by honest executions of the protocol with the help of the secret keys and reveals password information and the salt values stored by the server oracle. After polynomial number of executions, either for the given transcript values $(\alpha Q_S, x, z\mathbf{P}, m = \hat{e}(Q_S, zx\alpha\mathbf{P} + \alpha Q_C) \cdot \hat{e}(z\mathbf{P}, R))$ or for the given password information stored by

the server $(\hat{e}(z\mathbf{P}, R), z\mathbf{P})$ adversary $\mathcal{A}$ eventually generates valid (C, S, psw). Adversary $\mathcal{F}$ outputs $R = \mathrm{tr}(psw)$. The following probability is calculated

$$\xi_6(\kappa) = \frac{n_d(\kappa)}{T_1(\kappa)T_2(\kappa)T_3(\kappa)} - \frac{t}{2^{\beta_{D_n}} \cdot t_{pre}},$$

where $\frac{t}{2^{\beta_{D_n}} \cdot t_{pre}}$ denotes the probability that $\mathcal{A}$ finds psw by trying t number of (C, S, psw) tuples, where β_{D_n} is the min-entropy for dictionary D_n and t_{pre} denotes the computational cost to calculate the input value of the bilinear map from the password. Since t is polynomially bounded in κ and $n_d(\kappa)$ is non-negligible $\xi_6(\kappa)$ is non-negligible, that contradicts to the bilinear pairing one-wayness assumption. □

4. Efficiency

In order to confirm the results obtained we implemented the protocol for performance evaluation. The implementation was created in Python, which version is 3.9., and performed on an average personal computer with an AMD Ryzen 5 2600 processor, which has 6 cores and 12 threads with a clock rate of 3.4 GHz to 3.9GHz, 16 GB of 3600MHz RAM, and an M.2 NVMe SSD with 3200 MB/s writing and 3500 MB/s reading speed. In the Appendix the elliptic curve and its parameters can be checked.

4.1. Computation cost. The following table summarizes the numbers of the main operations on both server and client side. We can realize that the most applied operation - the hash - is also the fastest operation in the registration. We note that our registration implementation is single threaded. The reason for not implementing a multithreaded version is that the bottleneck in the implementation of the underlying computation of Tate pairings and scalar multiplications, which we did not focus in our work. However, even so the runtime of our protocol is convincing and registering multiple users at the same time can be extremely fast.

Operation	**User**	**Server**
Hash	5	3
EC scalar mult.	3	2
Bilinear pairing	3	2

Table 1. Number of operations

Table 2 shows the average execution time of the protocol's main operations. The operations run 10000 times to make the run time more accurate. The bilinear pairing is the most expensive operation, but still its run time is under 0.01 seconds.

Operation	**Time**
HMac	0,0000011
EC scalar mult.	0,002880
Bilinear pairing	0,007043

Table 2. Execution time of protocol's operation

4.2. Comparison with other schemes. The performance evaluation is based on the running time of the protocol compared to two other available solutions, to the Blind Registration Protocol and to the TLS handshake. Table 3 shows the result of comparison. All of our tests are repeated 100 times to make sure to get a precise result. The performance tests of the BPR protocols were completed on a laptop with an Intel Core Duo P8600 at 2.40GHz. We provide the computational time for only the TLS protocol run, the registration process takes more time, since an e-mail-based verification is also needed. Our comparison shows that our registration protocol efficiency achieves a better solution.

Scheme	**Client**	**Server**	**Full**
BPR- two server [26]	1,4 s	0,68 s	2,76 s
BPR - VPAKE [25]	0,72 s	0,67 s	1,5 s
TLS			0,168 s
Our proposition	0,072 s	0,023s	0,095 s

Table 3. Execution time of protocols

5. Conclusion

We have designed a password registration protocol, that could be an ideal alternative for the traditional registration method (email or TLS/SSL connection). It is important to note that during the registration we use the bilinear mapping, Mac and hash function hence we achieve good results in computational time. We have introduced a new definition for secure password registration protocols that considers security requirements for the password transmission and storage, as

well. We give a detailed security analysis, and we prove that our proposed protocol is a secure based on the assumptions that solving the Bilinear Diffie–Hellman problem is computationally infeasible the bilinear map is a one-way function and Mac is existentially unforgeable under an adaptive chosen-message attack, where the bilinear map is considered in the generic bilinear group model and the hash functions are supposed as random oracles.

Appendix

- The used elliptic curve: $y^2 = x^3 + x$ which is a supersingular curve over $\mathbb{Z}_p$ with $p = 731329576256467855322039941411215536384068289627312830254310277821058411810144462486413246228592183502383911176278505421042514024101864935444574549103938 7$
- In $\mathbb{Z}_q^*$ and in $\mathbb{G}, \mathbb{G}_T$ we use $q = 730750818665451459101842416358141509827966402561$

References

[1] M. Bellare and P. Rogaway, Entity authentication and key distribution, In: Advances in Cryptology – Crypto 93 Proceedings, Lecture Notes in Computer Science, Vol. 773, *Springer–Verlag, Berlin–Heidelberg*, 1994, 232–249.

[2] M. Bellare and P. Rogaway, Provably secure session key distribution: the three party case, In: Proceedings of the Twenty-Seventh Annual ACM Symposium on Theory of Computing, 1995, 57–66.

[3] M. Bellare, D. Pointcheval and P. Rogaway, Authenticated key exchange secure against dictionary attacks, In: Advances in Cryptology – EUROCRYPT2000, Lecture Notes in Computer Science, Vol. 1807, *Springer*, 2000, 139–155.

[4] S. M. Bellovin and M. Merritt, Encrypted key exchange: password-based protocols secure against dictionary attacks, In: IEEE Computer Society Symposium on Research in Security and Privacy–S&P, 1992, 72–84.

[5] S. Blake-Wilson, D. Johnson and A. Menezes, Key agreement protocols and their security analysis, In: Proceedings of the Sixth IMA International Conference on Cryptography and Coding, Lecture Notes in Computer Science, Vol. 1355, 30–45, 1997.

[6] D. Boneh, X. Boyen, and E. J. Goh, Hierarchical identity-based encryption with constant size ciphertext, In: Proceedings of EUROCRYPT'05, Lecture Notes in Computer Science, Vol. 3494, *Springer*, 2005, 440–456.

[7] D. Boneh and M. Franklin, Identity based encryption from the Weil pairing, In: Advances in Cryptography – Proceedings of Crypto 2001, Vol. 2139, 2001, 213–229.

[8] J. Bonneau, The science of guessing: analyzing an anonymized corpus of 70 million passwords, *IEEE S&P. IEEE Computer Society* (2012), 538–552.

[9] V. Boyko, P. MacKenzie and S. Patel, Provably secure password-authenticated key exchange using Diffie–Hellman, In: Proceedings of EUROCRYPT'00, Lecture Notes in Computer Science, Vol. 1807, *Springer*, 2000, 156–171.

[10] M. Brož and V. Matyáš, Selecting a new key derivation function for disk encryption, In: International Workshop on Security and Trust Management, Lecture Notes in Computer Science, Vol. 9331, *Springer*, 2015, 185–199.

[11] R. Canetti, S. Halevi, J. Katz, Y. Lindell, and P. MacKenzie, Universally composable password-based key exchange, In: Advances in Cryptology – Proceedings of EUROCRYPT'05, *Springer*, 2005, 404–421.

[12] L. Chen and C. Kudla, Identity based authenticated key agreement protocols from pairings, In: Proceedings of 16th IEEE Computer Security Foundations Workshop, 2003, 219–233.

[13] M. Dürmuth and T. Kranz, On password guessing with GPUs and FPGAs, In: Technology and Practice of Passwords – Proceedings of PASSWORDS'14, 2014, 19–38.

[14] W. Ford and B. S. Kaliski, Server-Assisted Generation of a Strong Secret from a Password, In: Proceedings of 9th IEEE International Workshops on Enabling Technologies: Infrastructure for Collaborative Enterprises, 2000, 176–180.

[15] P.-A. Fouque and M. Tibouchi, Deterministic encoding and hashing to odd hyperelliptic curves, In: Proceedings of International Conference on Pairing-Based Cryptography, Pairing-Based Cryptography – Pairing, 2010, 265–277.

[16] D. Freeman, Pairing-based identification schemes, *Hewlett-Packard Laboratories, Report No. HPL-2005-154* (2005).

[17] D. C. Feldmeier, and P. R. Karn, Unix password security-ten years later, In: Proceedings of Conference on the Theory and Application of Cryptology – CRYPTO 1989, Lecture Notes in Computer Science, Vol. 435, *Springer*, 1989.

[18] M. F. Grubb and R. Carte, Single Sign-On and the system administrator, In: Proceedings of the 12th Systems Administration Conference – LISA'98, 1998.

[19] Hashcat, hashcat - advanced password recovery, http://hashcat.net/.

[20] Internet Engineering Task Force, RFC 1510: The Kerberos Network Authentication Service (V5), 1993.

[21] S. Jarecki, M. Jubur, H. Krawczyk, N. Saxena and M. Shirvanian, Two-factor password-authenticated key exchange with end-to-end security, *ACM Transactions on Privacy and Security (TOPS)* **24.3** (2021), 1–37.

[22] S. Jarecki, H. Krawczyk and J. Xu, OPAQUE: An asymmetric PAKE protocol secure against pre-computation attacks, In: Advances in Cryptology – EUROCRYPT'18, Lecture Notes in Computer Science Vol. 10822, 2018.

[23] J. Kelsey, B. Schneier, C. Hall and D. Wagner, Secure applications of low-entropy keys, In: Information Security, Lecture Notes in Computer Science, Vol. 1396, *Springer*, *Berlin–Heidelberg*, 1998, 121–134.

[24] F. Kiefer and M. Manulis, Zero-knowledge password policy checks and verifier-based PAKE, In: Computer Security – ESORICS 2014, Lecture Notes in Computer Science, Vol. 8713, *Springer*, *Cham*, 2014.

[25] F. Kiefer and M. Manulis, Blind password registration for verifier-based PAKE, In: Proceedings of the 3rd ACM International Workshop on ASIA Public-Key Cryptography, 2016, 39–48.

[26] F. Kiefer and M. Manulis, Blind password registration for two-server password authenticated key exchange and secret sharing protocols, In: International Conference on Information Security, Lecture Notes in Computer Science, Vol. 9866, *Springer*, *Cham*, 2016, 95–114.

[27] H. Krawczyk and E. Pasi, HMAC-based extract-and-expand key derivation function (HKDF), *RFC 5869* (2010).

[28] H. Krawczyk, The opaque asymmetric pake protocol, *Internet-Draft* (2020), https://datatracker.ietf.org/doc/pdf/draft-irtf-cfrg-opaque-08.

[29] J. Ma, W. Yang, M. Luo and N. Li, A study of probabilistic password models, In: 2014 IEEE Symposium on Security and Privacy, 2014, 689–704.

[30] P. MacKenzie, S. Patel and R. Swaminathan, Password-authenticated key exchange based on RSA, *Int. J. Inf. Secur.* **9** (2010), 387–410.

[31] C. Mainka, V. Mladenov and J. Schwenk, SoK: Single Sign-On security – An evaluation of OpenID connect, In: 2017 IEEE European Symposium on Security and Privacy, 2017, 251–266.

[32] R. C. Merkle, A digital signature based on a conventional encryption function, In: Advances in Cryptology – CRYPTO'18, Lecture Notes in Computer Science, Vol. 293, *Springer*, *Berlin–Heidelberg*, 1987, 369–378.

[33] OneLogin, https://www.entrepreneur.com/article/295831.

[34] Openwall, John the Ripper password cracker, http://www.openwall.com/john/.

[35] C. Percival and S. Josefsson, The scrypt password-based key derivation function, *IETF Draft*, http://tools.ietf.org/html/josefsson-scrypt-kdf-00.txt.

[36] R. S. Pippal, C. D. Jaidhar and S. Tapaswi, Robust smart card authentication scheme for multi-server architecture, *Wireless Personal Communications* **72.1** (2013), 729–745.

[37] N. Provos and D. Mazieres, Bcrypt algorithm, *USENIX* (1999).

[38] A. Shamir, Identity-based cryptosystems and signature schemes, In: Advances in Cryptology – CRYPTO'84, Lecture Notes in Computer Science, Vol. 196, *Springer*, *Berlin–Heidelberg*, 1984.

[39] N. P. Smart, An identity based authenticated key agreement protocol based on the Weil pairing, *Electronics Letters* **38** (2002), 630–632.

[40] S. K. Sood, A. K. Sarje and K. Singh, An improvement of Wang et al.'s authentication scheme using smart cards, In: 2010 National Conference On Communications (NCC), 2010, 1–5.

[41] M. Soria-Machado, D. Abolins, C. Boldea and K. Socha, Kerberos Golden Ticket protection, mitigating Pass-the-Ticket on Active Directory, *CERT-EU Security Whitepaper 2014-007* (2016).

[42] Á. Vécsi, A. Bagossy and A. Pethő, Cross-platform identity-based cryptography using WebAssembly, *Infocommunications* **31** (2019).

[43] G. Yang, D. S. Wong, H. Wang and X. Deng, Two-Factor mutual authentication based on smart cards and passwords, *J. Comput. Syst. Sci.* **74** (2008), 1160–1172.

[44] F. F. Yao and Y. L. Yin, Design and analysis of password-based key derivation functions, *IEEE Transactions on Information Theory* **51** (2005), 3292–3297.

csanád bertók[†‡], andrea huszti[†], szabolcs kovács[†*], norbert oláh[†]
†faculty of informatics
university of debrecen
26 kassai road, h-4028 debrecen
hungary

‡mta de equations, functions
and curves research group

∗cclab ltd.

E-mail: {bertok.csanad, huszti.andrea, kovacs.szabolcs, olah.norbert}@inf.unideb.hu