

Password-Authenticated Key Exchange Protocols: A Survey

DING WANG, College of Cryptology and Cyber Science, Nankai University, Tianjin, China
GUANLING LI, College of Cryptology and Cyber Science, Nankai University, Tianjin, China
JINGWEI JIANG, College of Cryptology and Cyber Science, Nankai University, Tianjin, China

Password-authenticated key exchange (PAKE) protocols tackle the important problem of how to enable two parties, who share a low-entropy password, to establish a cryptographically strong session key for secure data communication. Although considerable research efforts have been devoted to designing hundreds of PAKE protocols, to the best of our knowledge, there have been few systematic reviews. In this work, we provide a comprehensive overview of PAKE research. We first propose a list of 13 desirable properties of PAKE protocols in terms of security and usability, enabling PAKE protocols to be systematically rated across a common spectrum. We then provide a taxonomy for PAKE protocols, and classify them into seven types according to their underlying design strategies. For each type, we investigate the inner working mechanisms of various representative protocols, and identify their pros, and cons. We further classify existing PAKE protocols from five other key perspectives (i.e., symmetry, number of participants, hardness assumptions, security goals, and round complexity) and review their development history under each classification, aiming at providing an in-depth and thorough understanding of the status quo of PAKE research. Based on 13 properties and six perspectives, we conduct a large-scale comparative evaluation of 71 representative PAKE protocols in a systematic manner. Finally, we highlight a few potential directions for the future design of PAKE protocols.

CCS Concepts: • **Security and privacy → Authentication**; • **Theory of computation → Cryptographic protocols**; • **General and reference → Surveys and overviews;**

Additional Key Words and Phrases: Password-authenticated key exchange, design strategy, evaluation criteria

ACM Reference Format:
Ding Wang, Guanling Li, and Jingwei Jiang. 2025. Password-Authenticated Key Exchange Protocols: A Survey. *ACM Comput. Surv.* 58, 6, Article 140 (December 2025), 37 pages. https://doi.org/10.1145/3774642

1 Introduction

In both traditional user-server environments and modern cloud/edge computing settings, user authentication constitutes one of the most critical lines of defense, ensuring that the communicating parties are intended entities. Existing methods for user authentication fall into three broad categories: something the user knows (e.g., passwords), something the user has (e.g., smart cards, USB tokens),

This work was supported in part by the National Natural Science Foundation of China under Grant 62222208, and by the Natural Science Foundation of Tianjin, China under Grant No. 25JCJQJC00230, and by the CCF-ANT Research Fund under Grant CCF-AFSGRF20250203.
Authors' Contact Information: Ding Wang (corresponding author), College of Cryptology and Cyber Science, Nankai University, Tianjin, Tianjin, China; e-mail: wangding@nankai.edu.cn; Guanling Li, College of Cryptology and Cyber Science, Nankai University, Tianjin, China; e-mail: liguanling@mail.nankai.edu.cn; Jingwei Jiang, College of Cryptology and Cyber Science, Nankai University, Tianjin, Tianjin, China; e-mail: jiangjingwei@nankai.edu.cn.
Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. Copyrights for components of this work owned by others than the author(s) must be honored. Abstracting with credit is permitted. To copy otherwise, or republish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee. Request permissions from permissions@acm.org.
© 2025 Copyright held by the owner/author(s). Publication rights licensed to ACM.
ACM 0360-0300/2025/12-ART140
https://doi.org/10.1145/3774642

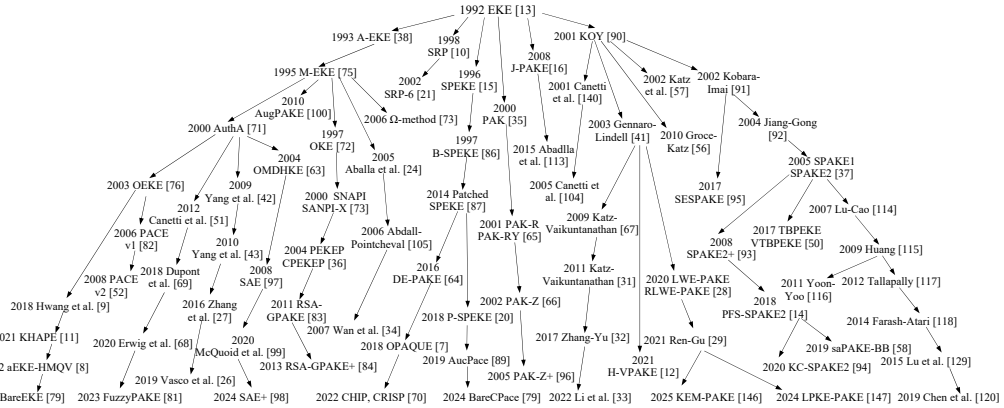


Fig. 1. A brief history of PAKE protocols, where $A \rightarrow B$ indicates that the authors explicitly state that the protocol B is an improvement based on the protocol A, or that B essentially uses a similar design strategy with A. For information on design strategies, see Section 3.

and something the user is (e.g., fingerprint, iris, and facial images) [1]. Among them, password-based authentication requires users only to remember and type a (username, password) pair. Thus, the user incurs almost no cost and can move conveniently without carrying hardware tokens. Although the latter two types of authentication may offer higher security or usability, they generally suffer from poor deployability [2]. In fact, due to the advantages of simplicity and convenience, none of these alternatives proposed in recent years can replace passwords [2–4], and passwords are expected remain the dominant user authentication method in the foreseeable future [5, 6].

The most widely used password authentication scheme in practice is "password-over-TLS" [7]. More specifically, the client-to-server channel is protected by TLS. During registration, the user sends her password to the server, which generally hashes the user's password together with a randomly selected salt and stores this salted-hash in a password file. During login, the user sends her password to the server over a TLS connection. The server then verifies the user's identity by applying a salted hash to the password. One disadvantage is that the password-over-TLS method relies crucially on the security of TLS. If the TLS channel is established with a compromised server's public key, the attacker can immediately learn passwords without performing an exhaustive offline dictionary attack [7, 8].

Password-authenticated key exchange (**PAKE**) protocols have been extensively studied in academia for over thirty years, encompassing both the symmetric setting and the asymmetric (clientserver) setting. A symmetric PAKE protocol allows two parties who share only a password to establish a cryptographically strong session key [8, 9]. An **asymmetric PAKE** (**aPAKE**) protocol extends this concept by allowing the server to store only a password-derived verifier rather than the password itself [10, 11]. When the password file is exposed, the attacker cannot directly obtain the password and has to perform an offline guessing attack to find the correct one [7, 12]. At IEEE S&P'92, Bellovin and Merritt [13] proposed the first PAKE protocol. Since then, considerable efforts have been given to the design of secure and efficient PAKE protocols (e.g., [7, 14–16]), and the development of some notable PAKE protocols is summarized in Figure 1. Particularly, a few of them have been used in the real world. For example, the SRP [10] is used in 1Password [17] and Apple HomeKit [18]; the J-PAKE [16] is used in Firefox Sync [19].

Motivations. Most of these works focus on revealing specific attacks/defects on existing PAKE protocols and proposing corresponding countermeasures (e.g., [11, 14, 20, 21]). Generally, protocol

designers assert the advantages of their protocol, while (unconsciously) overlooking dimensions in which their protocols fare poorly. Only limited effort have been made to provide systematic research on PAKE protocols from the perspective of design strategies. To the best of our knowledge, Hao-Oorschot's work [22] may be the closest to this article: They divided PAKE protocols into five types, i.e., (1) password as an encryption key; (2) password-derived generator; (3) trusted setup; (4) secure two-party computation; and (5) password-derived exponent, and compared the performance of seven representative PAKE protocols. But our work differs from it [22] in four aspects.

First, the classification criteria are different. The taxonomy proposed by Hao and Oorschot [22] is mainly based on the way to use a password, and it combines security models, major assumptions, and construction approaches, resulting in five classification types. In contrast, the classification of this article emphasizes the design strategies that are reused across family of protocols (see Figure 2) and results in seven types, because a password may play multiple roles in a PAKE protocol, confusion may occur when classifying PAKE protocols based on how the password is used. For example, in Jarecki et al.'s OPAQUE protocol [7], the password pw is used to derive the generator (i.e., $\mathsf{rw} = \mathsf{H}(pw, \mathsf{H}'(pw)^k)$), and the mapping of the password (i.e., rw) is also used as an encryption key (i.e., $c = \mathsf{Enc}_{\mathsf{rw}}(a, A, B)$), where $\mathsf{H}(\cdot)$ and $\mathsf{H}'(\cdot)$ are hash functions, $\mathsf{Enc}(\cdot)$ is encryption function, k is the server's key, and (a, A, B) are the server's long-term HMQV keys. According to the way of using passwords, the OPAQUE protocol belongs to type-1 and 2 in [22]. But from our perspective of the similar design strategy, OPAQUE will be classified into only one type (see Table 3).

Second, we propose comprehensive set of criteria consisting of 13 desirable properties that encompass a wide range of security and usability goals, and conduct a large-scale comparative evaluation, which is much more comprehensive than that of Hao-Oorschot's work [22]. It is known that it is incredibly difficult to design a PAKE protocol correctly. For example, the seminal EKE [13] and SRP [10] are found to be vulnerable to undetectable online guessing attacks [21, 23]; SPEKE [15] is pointed out to be vulnerable to impersonation attacks and key-malleability attacks in 2018 [20], twelve years after it was standardized by ISO/IEC. Moreover, various properties (e.g., perfect forward secrecy, user anonymity) desirable for varied application environments, but protocol designers often only illustrate the superior aspects over previous protocols (see Section 4.4). All this implies that an objective and comprehensive evaluation criterion set for PAKE protocols is necessary.

Third, in addition to the design strategy, we explore PAKE protocols from five complementary perspectives. Ideally, PAKE protocols could be classified using a unified framework to facilitate comparative analysis. Still, to solve practical problems in different applications, PAKE protocols are usually proposed with special structures, such as multi-party [24, 25], anonymous [26, 27], and quantum-resistant [28–30]. The large number and complex structure of PAKE protocols hinder the systematic analysis. It is virtually impossible to make a thorough comparison of all PAKE protocols based on one method alone. Using multiple classification methods will better illustrate the differences, pros, and cons of existing protocols. In order to supplement the information not covered by the design strategy, we understand the development of PAKE from five perspectives: symmetry, number of participants, hardness assumptions, security goals, and round complexity.

Fourth, we provide an overview of ten PAKE protocols that employ **smooth projective hash functions** (**SPHF**), a category not covered in the work of Hao-Oorschot [22]. The security of SPHF-based PAKE protocols typically relies on lattice-based assumptions, e.g., [31–33]. The analysis of such protocols not only enhances the completeness of the PAKE survey but also is helpful for guiding the design of quantum-secure PAKE protocols. Further, we explore newly proposed PAKE protocols (e.g., KHAPE [11] and aEKE-HMQV [8]), which demonstrate strong performance in terms of security, computation overhead, or round complexity (see Figure 3).

Scope. To keep the scope of the article manageable, we consider works that focus on the PAKE protocol, stemming from the pioneering PAKE protocol in 1992 [13]. It should be noted that, as

hundreds of PAKE protocols have been proposed, this survey mainly focuses on PAKE protocols that are highly representative or used to clarify a certain mainstream technique route (see Figure 2). Due to page limits, our work is not intended to be absolutely complete. Some subfields of PAKE (e.g., threshold PAKE and PAKE based on a chaotic map) are not discussed. We use Google Scholar and the database Web of Science as the main tools to discover related articles. In addition, we screen most of the related high-profile conferences, such as EUROCRYPT, CRYPTO, ASIACRYPT, ACM CCS, USENIX SEC, IEEE S&P, and NDSS, just to name a few, to find out recent works. The major keywords we have used include "password-authenticated key exchange", "password-based authentication key exchange", "password-only authentication and key exchange", and "PAKE".

Contributions. This article provides a systematic overview of password-authenticated key exchange protocols. In all, our contributions are summarized as follows:

— We provide a systematic framework for evaluating PAKE protocols, including a security model and a criterion set of 13 desirable properties in terms of security and usability.
— We conduct a comprehensive review of PAKE protocols, and for the first time classify them into seven types according to their underlying design strategies.
— We revisit existing PAKE protocols from five complementary perspectives (i.e., symmetry, number of participants, hardness assumptions, security goals, and round complexity) and discuss representative protocols of each classification to clarify their working mechanisms.
— We conduct a large-scale comparative evaluation of 71 representative PAKE protocols in a systematic manner based on our evaluation framework and six perspectives (i.e., round/flow, explicit authentication, computation/communication/storage overhead, and building blocks). In addition, we provide several insights into promising directions for future research on PAKE.

2 Preliminaries

2.1 Protocol Phases

To date, numerous PAKE protocols with diverse structures (e.g., symmetric [16], asymmetric [10], and multi-party [34]) have been published in the literature. Despite variations in structural design, the fundamental principle of PAKE protocols is to achieve authentication using only a low-entropy password. Hence, the phases of most PAKEs are largely the same: (1) The setup phase, which initializes the system's public parameters; (2) The registration phase, in which user identities and passwords are established; and (3) The login phase, which verifies the legitimacy of users.

— *Setup*. Setup is a preparatory phase where the server selects and publishes a set of necessary parameters for subsequent interactions with users. For example, the PAK protocol [35] publishes a multiplicative group $\mathbb{G}$ with generator g and prime order p, and the PEKEP protocol [36] publishes an RSA public key pair (N, e). Moreover, a series of PAKE protocols (e.g., PFS-SPAKE2 [14]) collectively refer to the setup and registration phases as initialization. But in this article, we only regard the computation without passwords (i.e., setup) as initialization.
— *Registration*. The registration phase is assumed to be executed over a channel that is secure against network adversaries (e.g., face-to-face, under TLS protection). Although some articles (e.g., [37]) describe the login phase in detail and omit the registration phase, the latter is an essential component of PAKE protocols. More specifically, a user chooses their identity (also called username) and password according to the server's policy (e.g., the username is different from existing ones, and the password length is greater than 8). Then the user sends a (username, password) pair to the server, and the server may store passwords in different ways.

Table 1. Capabilities of the Adversary

C-1	The adversary can offline enumerate the Cartesian product $D_{id} \times D_{pw}$, where D_{id} and D_{pw} denote the identity space and the password space, respectively.
C-2	The adversary can determine the victim's identity.
C-3	The adversary has full control of the communication channel.
C-4	The adversary can learn the victim's password.
C-5	The adversary can learn the previous session key(s).
C-6	The adversary can learn the server's long-term private key(s), as well as all other data stored on the server, but only when evaluating an eventual server compromise (e.g., forward secrecy).

For example, A-EKE [38] stores a hash of the password; AMP [39] stores a salted hash of the password and a random salt; OPAQUE [7] generates the password file with HMQV [40].

— *Login*. It is also called the login and authentication phase. In a PAKE protocol, both the user and the server prove their identity through a password: The user can type the correct password, and the server stores the password (or its verifier). This is different from the password-over-TLS scheme. Because in password-over-TLS, the user authenticates itself with a password, but the server authenticates itself with a certificate. It means that PAKE can achieve authentication between two parties without a public key infrastructure. Moreover, protocol designers employ various cryptographic primitives (e.g., oblivious pseudo-random functions [7], commitments [41], and public key juggling [16]) to design PAKE protocols.

2.2 Security Model

Generally, it is difficult for authors to make a comprehensive evaluation of their protocols. On the one hand, authors overlook certain weaknesses in their designs (e.g., [13, 15, 37]), and assign overly favorable assessments to particular properties their protocols satisfy (e.g., the anonymity [27, 42, 43]). On the other hand, it is challenging to clearly evaluate PAKE protocols by considering multiple factors, such as differences in application scenarios, assumptions on the adversary's capabilities, and requirements for the security goals (see Section 4). Hence, a rigorous security model capable of accurately simulating practical protocol execution is essential.

To simulate PAKEs more realistically, we need to clarify the capabilities of an adversary before defining the security model. Our adversary model is based on **Bellare-Pointcheval-Rogaway** (**BPR**) model [44] and other models, particularly those presented in [45–49]. The BPR model is widely used in the PAKE literature (e.g., [29, 33, 50–52]). We summarize the capacities of the adversary $\mathcal{A}$ for PAKE protocols in Table 1. Each capability is realistic and reasonable.

Specifically, the capability C-1 in Table 1 comes from the vulnerable behaviors of selecting passwords. Bonneau's work [53] indicates $\mid D_{id} \mid \leq \mid D_{pw} \mid \approx 10^6$. $\mathcal{A}$ can offline enumerate the Cartesian product $D_{id} \times D_{pw}$ in polynomial time. The capabilities of C-2 and C-3 are from the inherent vulnerabilities of network communication. In a non-anonymous network, $\mathcal{A}$ can easily obtain the source address, the destination address, and the user's identity through IP packets [54]. In addition, an attacker can eavesdrop, insert, delete, and modify any transmitted messages over the public channel by using network packet analysis tools. For the capability of C-4, if the password-entry interface is accidentally infected with viruses [55], an attacker can intercept the password entered by the user. The capability of C-5 may be due to the improper erasure of session keys after use, or the compromise of a host computer [45, 47]. Moreover, for the capability of C-6, when an attacker breaks into a computer system, the password file stored on the server side may be leaked (see [168]).

The BPR model [44] simulates the capabilities of an adversary by querying oracles. The security model presented here is applied to the basic client-server setting, which can be easily extended to other settings, such as the two-server setting and wireless sensor network.

Participants. We assume a user (also called client) $U \in \mathcal{U}$ and a server $S \in \mathcal{S}$, which participate in the PAKE $\mathcal{P}$. $\mathcal{U}$ and $\mathcal{S}$ are disjoint, i.e., $\mathcal{U} \cap \mathcal{S} = \emptyset$. We denote client instances by $U^i, i \in \mathbb{Z}$ and server instances by $S^j, j \in \mathbb{Z}$. In addition, we denote any kind of instance by $I \in \mathcal{U} \cup \mathcal{S}$.

Password and long-term keys. Each user U holds an identifier ID_U and a password PW_U. We assume that each PW_U is from a dictionary $\mathcal{D}$ that obeys the Zipf's law [49]. The size $|\mathcal{D}|$ is independent of the security parameter. Each server S stores a long-term key x and a password file (e.g., ID_U and a hash of the password PW_U in [38]).

Queries. The adversary $\mathcal{A}$ is a **probabilistic polynomial time** (PPT) algorithm with capabilities defined in Table 1. $\mathcal{A}$ has access to the following oracles:

- Execute(U^i, S^j): This query models the ability of an adversary $\mathcal{A}$ to perform passive attacks. That is, $\mathcal{A}$ eavesdrops on honest executions between U^i and S^j. This query returns the protocol transcript (e.g., the complete ordered messages exchanged) to $\mathcal{A}$.
- Send(I, m): This query models the ability of $\mathcal{A}$ to perform active attacks. $\mathcal{A}$ sends a message m to the instance I. After receiving m, I runs according to the protocol specification and updates its state. Finally, the message output by I is given to $\mathcal{A}$.
- Test(I): This query defines the session key's semantic security. If the instance I has computed a session key sk, then a private coin c is flipped. If $c = 0$, it returns a random key of the same size chosen uniformly at random. If $c = 1$, it returns the session key sk of the instance I. If no session key is defined for instance I, it returns the undefined symbol $\perp$.
- Reveal(I): This query models the corruption of session keys. If the instance I has computed a session key sk, and neither I nor its partner was asked by a Test query, it returns the current session key sk of the instance I. Otherwise, it returns $\perp$.
- Corrupt(I): This query models the corruption of the instance I. If $I = U$, it outputs the password PW_U of U; if $I = S$, it outputs the secret key and the password file of the server S.

Accepting. If an instance I has run the protocol successfully and computed a session key sk, the instance I is considered to be accepted. Otherwise, it is considered to be rejected.

Partnering. We denote the session identifier by sid, which may be a concatenation of all messages sent and received by the instance [31, 32, 56]. Instances U^i and S^j are partnered if: (1) Both instances have been accepted, i.e., $sk_U^i = sk_S^j$; (2) Both instances shared the same session identifier, i.e., $sid_U^i = sid_S^j$; (3) The partner identifier of U^i is $pid_U^i = S$, and the partner identifier of S^j is $pid_S^j = U$.

Freshness. The notion of freshness captures the fact that a session key can't be trivially known to the adversary $\mathcal{A}$. An instance I is considered to be fresh if: (1) I has accepted a session key sk; (2) Neither I nor its partner has experienced the Reveal-query or Corrupt-query.

Security. The security notion requires that $\mathcal{A}$ cannot distinguish the real session key from a random string of the same size in polynomial time. $\mathcal{A}$ breaks the security of the PAKE protocol $\mathcal{P}$ if it makes a Test query to a fresh instance, and outputs a bit c' with $c' = c$. We define this event by Succ. The advantage of $\mathcal{A}$ in breaking this game is $\mathrm{Adv}_{\mathcal{P}}^{\mathrm{ake}}(n)(\mathcal{A}) = 2\Pr[c' = c] - 1$. We say that $\mathcal{P}$ is semantically secure if for all dictionaries $\mathcal{D}$ and any PPT adversary $\mathcal{A}$ making at most q_{send} online guessing attacks, $\mathrm{Adv}_{\mathcal{P}}^{\mathrm{ake}}(\mathcal{A})$ is negligible after excluding the advantages of online password guessing.

2.3 Evaluation Criteria

At IEEE S&P'12, Bonneau et al. [2] conducted a comparative evaluation of 35 representative web authentication schemes (including textual passwords, hardware tokens, and biometrics) based on 25 properties, and pointed out the irreplaceability of passwords. In the field of PAKE, some desirable properties have well attracted the attention of researchers, such as forward secrecy [14, 57], user anonymity [27, 43], resistance to pre-computation attacks [7, 58], and round optimality [31, 33],

while others are not (e.g., resistance to known-key attacks). Building upon prior work (e.g., [2, 8, 14, 20, 45]), we propose an evaluation metric with 13 desirable properties in terms of security and usability, enabling PAKEs to be systematically rated across a common spectrum.

2.3.1 *Security Benefits.*

S1. Key agreement. The user and the server jointly establish a common session key upon completion of the protocol. The consistency of a session key implies the authenticity of both parties, and it will protect subsequent data communication.

S2. Mutual authentication. The user and the server can verify the authenticity of each other. Considering the possible malicious/compromised authentication server, only user-to-server authentication is not enough, and may lead to irreparable losses for users.

S3. Explicit authentication. It is often mentioned together with implicit authentication. Implicit authentication means that both parties may not notice the failure of the protocol until they use their session keys to interact with errors. Explicit authentication means that an honest party can immediately know whether the protocol succeeded or not, at the end of PAKE.

S4. Perfect forward secrecy. It can be divided into **weak forward secrecy** (**wFS**) and **perfect forward secrecy** (**PFS**) according to whether the attacker has actively participated in the protocol. PFS means that even if the long-term private key of one or more parties is stolen by an attacker, the security of previous session keys is still not threatened [14].

S5. User anonymity. The protocol protects user identity and achieves the untraceability of user activities [47]. In other words, an attacker cannot confirm the user's identity from the protocol messages or distinguish whether two independent sessions are from the same user.

S6. Resistance to known attacks. The PAKE protocol needs to resist various known attacks. It is a basic requirement for security. We summarize eleven possible attacks.

(1) *Replay attack*. This attack exploits the lack of message freshness. An attacker can replay (possibly modify) previous messages to spoof legitimate users. If the previous message replayed by the attacker has not been detected by legitimate users until the end of the PAKE protocol execution, the replay attack is deemed successful.

(2) *Man-in-the-middle attack*. An active "man-in-the-middle" attacker can manipulate the session key without being detected by intercepting, modifying, delaying, and interleaving messages between honest communication entities.

(3) *Offline password guessing attack*. An attacker can verify the correctness of the guessing password offline by utilizing protocol messages between legitimate users. The assumption of the attacker's ability here does not involve the compromise of long-term private keys and password files, but only focuses on the eavesdropping of protocol messages.

(4) *Undetectable online password guessing attacks*. An attacker can exclude multiple wrong passwords in a single online attempt. In addition, there are two special attacks against RSA-based PAKE protocols that combine offline guessing with online guessing, i.e., *e*-residue attacks [23] and separation attacks [59].

(5) *Unknown key-share attack*. An attacker forces honest parties into establishing a session key where at least one party does not know that the session key is shared with others.

(6) *Parallel-session attack*. This attack works when the user is engaged in several sessions in parallel with another user. In two parallel sessions with the same user, the attacker acts as the recipient of the first session and the initiator of the second session. The attacker can pass authentication by relaying the sender's message back to the sender.

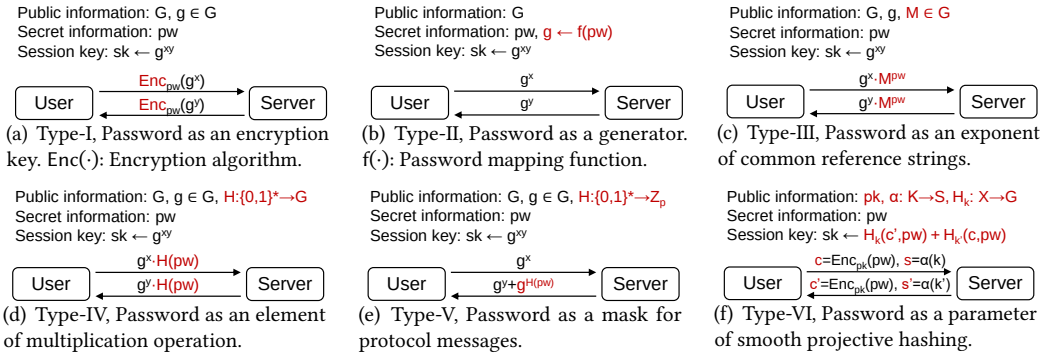
(7) *Known-key attack*. An attacker who knows a session key established between legitimate users can compute other secrets (e.g., the password). A known-key attack emphasizes the compromise of the session key, not the long-term private key or password file.

(8) *Key compromise impersonation attack*. This attack captures the compromise of the long-term private key of parties. After the long-term key of the victim is compromised, an attacker will inevitably be able to impersonate the victim, which is also known as the impersonation attack. But **key compromise impersonation** (**KCI**) attacks highlight that the attacker can cheat the victim by impersonating other entities.
(9) *Pre-computation attack*. An attacker can pre-compute all passwords in the dictionary (such as a reverse lookup table from the hash to the plaintext password), so that they can find the password immediately if the server is compromised.
(10) *Quantum attack*. This attack reflects the capability of quantum adversaries to efficiently solve the classical hardness assumptions (e.g., integer factorization and discrete logarithm) underlying PAKE. Although significant progress has been made in developing quantum algorithms with exponential speedups [60], the most impactful quantum attacks to date are those leveraging Shor's algorithm [61] and Grover's algorithm [62]. A quantum adversary equipped with a large-scale quantum computer can run Shor's algorithm to break PAKEs (e.g., OPAQUE [7] and KHAPE [11]) based on classical hardness assumptions, and employ Grover's algorithm to accelerate brute-force search and hash collision attacks.

In practice, **denial-of-service** (**DoS**) attacks represent a common threat in which adversaries flood servers with authentication requests in an attempt to exhaust system resources. To mitigate such attacks, Bresson et al. [63] proposed the OMDHKE, which incorporates a client-side proof-of-work mechanism. However, this defense is a general mechanism for a broad range of cryptographic protocols rather than an inherent feature of PAKE design. Moreover, proof-of-work imposes significant computational overhead on clients, thereby reducing usability. Fundamentally, resistance to DoS attacks is a broader concern for all interactive systems and is not unique to PAKE protocols. Service providers are expected to implement non-cryptographic protections (such as rate limiting, CAPTCHA challenges, or account lockout mechanisms) that lie outside the scope of cryptographic protocol design. For this reason, we exclude DoS resistance from our evaluation criteria for PAKEs.

2.3.2 *Usability Benefits.*

U1. Password friendly. A user can freely choose their password and have no other burden besides memorizing it, including the ability to update the password at any time after initial registration. The high-entropy key generated by the system offers higher security, but it compromises the vital motivation of users to choose password authentication, i.e., memorability and deployability. Hence, weak passwords are a fact of life that we must face. The goal of password authentication researchers is not to prevent users from using weak passwords, but to find suitable methods for achieving authentication using weak passwords.
U2. Login device friendly. The practical deployment of PAKE requires a client, such as a browser plugin, a mobile software [14, 64]. Users should be able to log in on any client configured with a PAKE. It means that the client should only store system public parameters (e.g., group elements in J-PAKE [16] and common reference strings in SPAKE2 [37]).
U3. Support pre-computation. The client or the server can pre-compute parameters that don't need to wait until the other party's message is received. For example, the PAKE using the DH key exchange (e.g., [7, 14, 15]) can pre-compute g^x, g^y, where g is the public base for exponentiation and $\{x, y\}$ are random numbers of the client and the server, respectively.
U4. Storage overhead. The storage overhead is quantified as the bit-lengths of the necessary public parameters, key materials, identifiers, password files, and so on. Note that when two users share a password directly (e.g., J-PAKE [16]), neither party needs to store it. Considering the memory constraints of the participants, the storage overhead should be as low as possible.



(a) Type-I, Password as an encryption key. Enc(·): Encryption algorithm.

(b) Type-II, Password as a generator. f(·): Password mapping function.

(c) Type-III, Password as an exponent of common reference strings.

(d) Type-IV, Password as an element of multiplication operation.

(e) Type-V, Password as a mask for protocol messages.

(f) Type-VI, Password as a parameter of smooth projective hashing.

Fig. 2. Overview of design strategies for Types I~VI. We provide four parts: public information, secret information, session keys, and protocol messages. The characteristics of each type are marked in red. Note that for simplicity and intuition, both parties share a password, and the details of the session key are omitted.

U5. Computation overhead. The computation overhead is quantified as the time cost for the participants to execute the protocol algorithm, which can affect the total running time. In particular, these values that can be pre-computed have no effect. As a result, if a PAKE protocol supports pre-computation, we only need to consider the execution of the protocol algorithm after pre-computation. Typically, low computation overhead is expected.

U6. Transmission overhead. The transmission overhead is quantified as the bit length of the transmitted data for the participants in executing the protocol. Note that some PAKE protocols (e.g., [8]) omit the explicit transmission of participant identifiers, but it is still part of the protocol. Heavy communication overhead should be avoided.

U7. Round complexity. A protocol flow refers to the data stream sent by a participant each time. If two participants can each send a protocol flow in no order, the two flows are considered as one round (e.g., [15, 16]). Otherwise, they are considered as two rounds (e.g., [7, 10]). A large number of protocol rounds will increase the running time. Round complexity is affected by explicit/implicit authentication. For example, if one-round PAKE achieves mutually implicit authentication (e.g., PPK [35] and SPAKE2 [37]), it is round-optimal.

3 Taxonomy of PAKE Protocols

We now provide a taxonomy of PAKE protocols in terms of the design strategy (see Figure 2). Our taxonomy captures the general evolutionary trend of PAKE development and emphasizes the common design principles shared across different protocols. To keep our survey general and simple, we only select representative PAKE protocols to be included in Tables 2~7. In addition, we only discuss how such protocols achieve authentication and key exchange here, and leave the discussion of all aspects of each protocol individually later (see Tables 9 and 10). For simplicity, we employ "password" in Types I~VI instead of "the password itself or the mapping of the password". For example, in Type-I Password as an encryption key, the KHAPE protocol [11] uses the password as an encryption key, and the OKAPE-HMQV protocol [8] employs a hash of the password for the same purpose (see Table 2).

Type-I. Password as an encryption key. The password is used to derive an encryption key. The message to be encrypted is usually an ephemeral public key. Only the party holding the password can decrypt the public key. Examples: KHAPE [11], OKAPE-HMQV [8].

Table 2. PAKE Protocols using the Password as an Encryption Key (Type-I)[†]

Protocol	Year	Reference	Sym./Asym.	Plaintext	Key for encryption/decryption	Ciphertext
BareEKE	2024	CRYPTO [79]	Symmetric	$m = g^x$	$k = pw$	$c = (E_k(m), id)$
SweetPAKE	2024	AisaCCS [80]	Symmetric	$m = g^x$	$k = pw$	$c = E_k(m)$
FuzzyPAKE	2023	AISACRYPT [81]	Symmetric	$m = \text{YGC}(x)$	$k = pw$	$c = E_k(\text{OT}(m))$
OKAPE-HMQV	2022	EUROCRYPT [8]	Asymmetric	$m = g^y$	$k = \text{H}(pw)$	$c = E_k(m)$
aEKE-HMQV	2022	EUROCRYPT [8]	Asymmetric	$m = g^y$	$k = \text{H}(pw)$	$c = E_k(m)$
KHAPE	2021	CRYPTO [11]	Asymmetric	$m = (a, g^b)$	$k = pw$	$c = E_k(m)$
PACE v2	2008	BSI report [52]	Symmetric	$m = s \in \{0,1\}^{t_1}$	$k = pw$	$c = E_k(m)$
PACE v1	2006	BSI report [82]	Symmetric	$m = s \in \mathbb{Z}_q$	$k = \text{H}(pw\|0)$	$c = E_k(m)$
Ω-method	2006	CRYPTO [78]	Asymmetric	$m = sk_{sign}$	$k_1 = sk, k_2 = pw$	$c = E_{k_1}(E'_{k_2}(m))$
OEKE	2003	ACM CCS [76]	Symmetric	$m = g^y$	$k = \text{H}(pw)$	$c = E_k(m)$
AuthA	2000	IEEE P1363 [71]	Asymmetric	$m_1 = g^x, m_2 = g^y$	$k = \text{H}(pw)$	$c_1 = E_k(m_1), c_2 = E_k(m_2)$
M-EKE	1995	ACM SIGOPS [75]	Symmetric	$m = g^x$	$k = pw$	$c = E_k(m)$
A-EKE	1993	ACM CCS [38]	Asymmetric	$m_1 = g^x, m_2 = g^y$	$k = \text{H}(pw)$	$c_1 = E_k(m_1), c_2 = E_k(m_2)$
DH-EKE	1992	IEEE S&P [13]	Symmetric	$m_1 = g^x, m_2 = g^y$	$k = pw$	$c_1 = E_k(m_1), c_2 = E_k(m_2)$
CPEKEP	2004	ASIACRYPT [36]	Symmetric	$m = a \in_R \mathbb{Z}_n^*$	$k = \text{H}(n\|e\|r_U\|r_S\|pw)$	$c = E'^{r-1}(k \cdot E(m)), r \in (0, 80]$
PEKEP	2004	ASIACRYPT [36]	Symmetric	$m = a \in_R \mathbb{Z}_n^*$	$k = \text{H}(n\|e\|r_U\|r_S\|pw)$	$c = E'(k \cdot E(m)), r = \lfloor log_2 n \rfloor$
SNAPI-X	2000	ASIACRYPT [73]	Asymmetric	$m = a \in_R \mathbb{Z}_n^*$	$k = \text{H}(n\|e\|r_U\|r_S\|g^{\text{H}(ID_U\|ID_S\|pw)})$	$c = k \cdot E(m)$
SNAPI	2000	ASIACRYPT [73]	Symmetric	$m = a \in_R \mathbb{Z}_n^*$	$k = \text{H}(n\|e\|r_U\|r_S\|pw)$	$c = k \cdot E(m)$
RSA-EKE	1992	IEEE S&P [13]	Symmetric	$m = e$	$k = pw$	$c = k \cdot E(m)$

[†]This table shows the plaintext (i.e., m, m_1, m_2), key (i.e., k, k_1, k_2), and ciphertext (i.e., c, c_1, c_2). $E(\cdot), E'(\cdot)$ are encryption algorithms. n, e is the RSA public key. YGC$(\cdot)$ denotes Yao's garbled circuits, and OT$(\cdot)$ denotes oblivious transfer. *sid* is the session identifier, and *sk* is the session key. $\{x, r_U\}$ and $\{y, r_S\}$ are the ephemeral secret keys selected by the user U and the server S, respectively. a, b are private keys used in HMQV [40]. pw is the user's password. H$(\cdot)$ is a hash function. $\|$ denotes string concatenation. The DH-based PAKEs appear above the dash, the RSA PAKEs below.

Type-II. Password as a generator. The password is used to derive a generator. Each party raises it to the power of an ephemeral private key. Participants prove their knowledge of the password by deriving the common generator. Examples: SPEKE [15], P-SPEKE [20].

Type-III. Password as an exponent of common reference strings. In this type, there are one or more common reference strings. Each party raises it to the power of the password, and then multiplies it by an ephemeral public key. Examples: SPAKE2 [37], PFS-SPAKE2 [14].

Type-IV. Password as an element of the multiplication operation. The protocol uses a hash-to-group (or hash-to-curve) function to map the password to a group element, then multiply by an ephemeral public key. Examples: PAK [35], PAK-R [65], and PAK-Z [66].

Type-V. Password as a mask for messages. The password is used as an exponent to mask the ephemeral public key. There is no need for a common reference string or a hash-to-group function, and only public key technology is used. Examples: SRP [10], AMP [39], TP-AMP [**?**].

Type-VI. Password as a parameter of smooth projective hashing. In this type, the session key is computed based on the properties of the smooth projective hash function, where the password is used as a parameter. Examples: Katz-Vaikuntanathan [67], Groce-Katz [56].

Type-VII. Others. The design strategy differs from the above six types, and there is also no uniform or regular usage of the password. Roughly speaking, the password is treated as a bit string (but with different roles) in [51, 68, 69] and to replace the private key of a trusted third party [70].

3.1 Type-I: Password as an Encryption Key

Bellovin and Merritt [13] presented three variants of EKE: **Diffie-Hellman EKE** (**DH-EKE**), RSA-EKE, and ElGamal-EKE. Subsequent research has proposed numerous extensions and enhancements to DH-EKE (e.g., [8, 11, 71]) and RSA-EKE (e.g., [36, 72, 73]), while ElGamal-EKE is less common. A plausible reason is that the defined group $\mathbb{G}$ usually can be a finite group, or an elliptic curve group [74]. Since exponentiation in finite fields and the point multiplication operation in elliptic

curve cryptosystems play similar roles, a PAKE protocol in the DH setting can generally also be implemented in the ECC setting. Table 2 summarizes PAKE protocols that use the same paradigm as EKE [13].

3.1.1 DH-EKE. In Bellovin-Merritt's work [13], the EKE protocol requires both parties to hold a plaintext password. This causes passwords to be exposed immediately after the server is compromised. To address this problem, Bellovin and Merritt [38] extended the EKE protocol to the **Augmented Encrypted Key Exchange** (**A-EKE**) protocol, the first verifier-based/asymmetric PAKE protocol. In order to reduce the relative complexity of EKE [13], Steiner et al. [75] proposed the **minimal EKE** (**M-EKE**) protocol. M-EKE only requires three messages and three encryption operations over a single data block. Similarly, considering the complexity, Bellare and Rogaway [71] proposed AuthA with lower cost in communications and greater versatility than EKE [13]. Later, Bresson et al. [76] modeled AuthA as the **One-Encryption Key-Exchange** (**OEKE**) protocol. Then, the German Federal Office for Information Security (BSI) proposed the **Password-Authenticated Connection Establishment** (**PACE v1**) protocol and its updated version, PACE v2 [77]. The participants of PACE v1/v2 are a chip contained in a machine-readable travel document and a reader. The chip is required to send a random nonce encrypted with the password to the terminal.

Gentry et al. [78] proposed the Ω-method to convert a symmetric PAKE to an asymmetric PAKE. The Ω-method requires two additional communication rounds and one signature computation/verification. More specifically, the server sends a private key sk_{sign} that is double encrypted with a password pw and a temporary session key sk, i.e., $E_{sk}(E'_{pw}(sk_{sign}))$. Then the client decrypts the sk_{sign} and sends the protocol transcript signed with sk_{sign}. At CRYPTO'21, Gu et al. [11] proposed the **Key-Hiding Asymmetric PAKE** (**KHAPE**) protocol. KHAPE [11] and EKE [13] are different in encryption: EKE [13] uses the password to encrypt each *unauthenticated* **key exchange** (**KE**) message and requires the KE messages must be random in some message space; KHAPE [11] uses the password to encrypt an ***authenticated*** **key exchange** (**AKE**) message and requires the underlying AKE to have the key-hiding property. Motivated by low round complexity, Santos et al. [8] proposed aEKE-HMQV and OKAPE-HMQV, which use the password to encrypt a *unilaterally authenticated* key exchange message. The aEKE-HMQV uses an unsalted password hash and completes in single simultaneous flow, while OKAPE-HMQV uses a salted password hash and takes two rounds.

3.1.2 RSA-EKE. The RSA-EKE protocol proposed by Bellovin and Merritt [13] has two apparent constraints: (1) A new RSA key pair (i.e., n, e, and d) needs to be generated for each protocol execution; (2) Users need first to encrypt the RSA public key e before transmitting it. To overcome these issues, Lucks [72] introduced the OKE protocol, so that the generated RSA key pair can once be reused and the public key e is sent openly. Then, Mackenzie et al. [73] pointed out that an e-residue attack is feasible for the OKE protocol, since the receiver cannot verify whether the RSA public exponent e satisfies the equation $\gcd(e, \phi(n)) = 1$ without knowing the factorization of the RSA modulus n. To thwart this attack, Mackenzie et al. [73] introduced SNAPI and SNAPI-X, which require the exponent $e(e > n)$ to be a prime number ensuring that $\gcd(e, \phi(n)) = 1$ holds. Later, Zhang [36] presented PEKEP and CPEKEP, which use multiple RSA encryption (i.e., r times in Table 2) to resist e-residue attacks, thereby eliminating the assumption that the RSA public exponent must be a large prime number. Then, Wei et al. [83] extended PEKEP to a gateway-oriented PAKE protocol, named RSA-GPAKE. RSA-GPAKE establishes an authenticated session key between the user and the gateway. Subsequently, Wang et al. [84] pointed out that the RSA-GPAKE protocol is vulnerable to separation attacks [59], and put forward the RSA-GPAKE+ protocol. RSA-GPAKE+ is inspired by the idea of avoiding information leakage in Zhang's work [36].

Table 3. PAKE Protocols using the Password as a Generator (Type-II)[†]

Protocol	Year	Reference	Sym./Asym.	Generator	User→Server[*]	Server→User
BareCPace	2024	CRYPTO [79]	Symmetric	$g = \mathsf{Map2Point}(\mathsf{H}(sid \Vert g_1^{\mathsf{PBKDF}(ID_U \Vert pw)}))$	$X = g^x$	$Y = g^y$
AucPace	2019	IACR Trans. [89]	Asymmetric	$g = \mathsf{Map2Point}(\mathsf{H}(sid \Vert g_1^{x \cdot \mathsf{PBKDF}(ID_U \Vert pw \Vert salt)}))$	$X = g^x$	$Y = g^{xy}$
CPace	2019	IACR Trans. [89]	Symmetric	$g = \mathsf{Map2Point}(\mathsf{H}(sid \Vert g_1^{\mathsf{PBKDF}(ID_U \Vert pw)}))$	$X = g^x$	$Y = g^y$
P-SPEKE	2018	IEEE TIFS [20]	Symmetric	$g = \mathsf{H}(pw)^2$	$X = g^x$	$Y = g^y$
OPAQUE	2018	EUROCRYPT [7]	Asymmetric	$g = \mathsf{H}(pw)$	$X = g^x$	$Y^* = g^{xy}$
SPEKE	1996	CCR [15]	Symmetric	$g = pw^2$	$X = g^x$	$Y = g^y$

[†]$\mathsf{H}(\cdot)$ is a hash-to-group function. The base g is computed by each party and is not public, while g_1 is a public generator. Due to space constraints, we omit protocol messages that are not related to g. For example, we only show a part of the OPAQUE protocol [7], namely, the OPRF interaction. This is because the use of OPRF to resist pre-computation attacks is the most significant feature of the OPAQUE protocol.

[*]User→Server denotes the message sent from the user to the server, while Server→User denotes the message sent from the server to the user. We use $X = g^x$ and $Y = g^y$ to denote the ephemeral Diffie-Hellman public keys, and X^*, Y^* to denote the values after further operations (such as exponents). Table 4~7 also uses the above symbols.

3.2 Type-II: Password as a Generator

In 1996, Joblon [15] proposed a **simple password exponential key exchange** (**SPEKE**) protocol, and discussed three attacks related to the SPEKE protocol: discrete log computation, leaking information, and small subgroup confinement. Informally, both SPEKE [15] and DH-EKE [13] can be regarded as slight modifications to the basic Diffie-Hellman key exchange [85]. However, in the protocol design, even the slightest modification can have profound effects [16]. DH-EKE [13] encrypts key exchange messages by using a password as a symmetric key. Yet, SPEKE [15] converts the password into a base for exponentiation, which is the common feature of Type-II.

Then, Jablon [86] proposed an extension of SPEKE (i.e., B-SPEKE) to prevent user passwords from being exposed directly after the server is compromised. In addition to hashing the password, B-SPEKE [86] adds a second Diffie-Hellman exchange to prove the participant's knowledge of the password. Later, Hao and Shahandashti [87] presented two attacks on the SPEKE protocol [86]: an impersonation attack and a key malleability attack. To resist these two attacks, Hao and Shahandashti [87] suggested adding communication messages to the computation of session keys, where the communication messages include not only the identifiers of both parties, but also the ephemeral Diffie-Hellman exchange values. Then, Hao et al. [20] proposed a P-SPEKE protocol based on the idea from [87], which has been included in the ISO/IEC 11770-4 [88].

In 2019, Haase and Labrique [89] proposed cPACE and AucPace tailored for the Internet of Things. The generation of the base in [89] includes the session identifier, the user's identity, and the password. Besides, both DE-PAKE [64] and OPAQUE [7] use an **Oblivious Pseudo-Random Function** (**OPRF**) to provide security upon server compromise. To retrieve the OPRF value $\mathsf{H}'(pw, \mathsf{H}(pw)^y)$, the client raises the hashed value $\mathsf{H}(pw)$ to a random exponent x, and sends $X = \mathsf{H}(pw)^x$ to the server (see Table 3). The server checks that $X \in \mathbb{G}$ and raises it to the secret exponent y, i.e., $Y^* = g^{xy}$. Finally, the client completes the OPRF computation by computing $\mathsf{H}'(pw, Y^{1/x})$.

3.3 Type-III: Password as an Exponent of Common Reference Strings

To eliminate the reliance on the random oracle assumption, Katz et al. [90] proposed a PAKE protocol, commonly referred to as KOY, at EUROCRYPT 2001, which operates solely in the standard model. The KOY protocol assumes a shared set of common parameters accessible to all participating entities. Then, Kobara and Imai [91] reduced the computational cost of KOY: Only about 2.34 modular exponentiations are required in [91], while more than 6.5 modular exponentiations are required in KOY [90]. Jiang and Gong [92] optimized the communication round of KOY: Only

Table 4. PAKE Protocols using the Password as an Exponent of Common Reference String (Type-III)[†]

Protocol	Year	Reference	Sym./Asym.	Public para.	User→Server	Server→User
KC-SPAKE2+	2020	TCC [94]	Asymmetric	$g, M \in \mathbb{G}$	$X^* = g^x \cdot M^{\mathsf{H}(ID_U \| ID_S \| pw)}$	$Y = g^y$
KC-SPAKE2	2020	TCC [94]	Symmetric	$g, M \in \mathbb{G}$	$X^* = g^x \cdot M^{pw}$	$Y = g^y$
saPAKE-BB	2019	CRYPTO [58]	Asymmetric	$g_1, g_2, M \in \mathbb{G}$	$X_1^* = g_1^x \cdot M^{\mathsf{H}(pw)}, X_2^* = g_2^x$	$Y^* = (g_2^{\mathsf{H}(pw)/(salt+\mathsf{H}(pw))})^{y_1} \cdot (X_1^*)^{y_2} \cdot (X_2^*)^{y_3}$
PFS-SPAKE2	2018	ProvSec [14]	Symmetric	$g, M \in \mathbb{G}$	$X^* = g^x \cdot M^{pw}$	$Y = g^y$
VTBPEKE	2017	AsiaCCS [50]	Asymmetric	$g, M \in \mathbb{G}$	$X^* = g^x \cdot M^{x \cdot \mathsf{H}(salt \| pw)}$	$Y^* = g^y \cdot M^{y \cdot \mathsf{H}(salt \| pw)}$
TBPEKE	2017	AsiaCCS [50]	Symmetric	$g, M \in \mathbb{G}$	$X^* = g^x \cdot M^{x \cdot pw}$	$Y^* = g^y \cdot M^{y \cdot pw}$
SESPAKE	2017	RFC 8133 [95]	Asymmetric	$P, Q \in E$	$X^* = x^*P - \mathsf{H}(ID_U \| pw)^*Q$	$Y^* = y^*P + \mathsf{H}(ID_U \| pw)^*Q$
SPAKE2+	2008	EUROCRYPT [93]	Asymmetric	$g, M, N \in \mathbb{G}$	$X^* = g^x \cdot M^{\mathsf{H}(ID_U \| ID_S \| pw)}$	$Y^* = g^y \cdot N^{\mathsf{H}(ID_U \| ID_S \| pw)}$
SPAKE2	2005	CT-RSA [37]	Symmetric	$g, M, N \in \mathbb{G}$	$X^* = g^x \cdot M^{pw}$	$Y^* = g^y \cdot N^{pw}$
SPAKE1	2005	CT-RSA [37]	Symmetric	$g, M, N \in \mathbb{G}$	$X^* = g^x \cdot M^{pw}$	$Y^* = g^y \cdot N^{pw}$
KOY	2001	EUROCRYPT [90]	Symmetric	$g, M \in \mathbb{G}$	$X^* = g^x \cdot M^{pw}$	$Y^* = g^y \cdot M^{pw}$

[†]For simplicity and better intuition, we omit other values in the protocol message that are not related to the common reference string, e.g., [50, 58, 95]. $\{x\}$ and $\{y, y_1, y_2, y_3\}$ are the ephemeral secret keys selected by the user U and the server S, respectively. In addition, P and Q are two points on the elliptic curve E.

three rounds are required to achieve explicit mutual authentication in [92], while four rounds are required in KOY [90].

Abdalla and Pointcheval [37] argued that, although KOY [90] completely eliminates the reliance on random oracles, it is less efficient than EKE [13] and is vulnerable to man-in-the-middle attacks. Therefore, for efficiency reasons, Abdalla and Pointcheval [37] presented two protocols: SPAKE1 and SPAKE2 (see Table 4). Both protocols are comparable in efficiency to any existing EKE protocol and only require one random oracle instance. In order to provide security upon server compromise, Cash et al. [93] proposed an improved version (i.e., SPAKE2+) which uses a password-derived value ($\mathsf{H}(ID_U \| ID_S \| pw)$, instead of pw) as the exponent (see Table 4). Additionally, in order to satisfy perfect forward secrecy, Becerra et al. [14] proposed a PFS-SPAKE2 protocol by incorporating key-confirmation codes in SPAKE2 [37]. Then, Shoup [94] proposed two protocols: KC-SPAKE2 and KC-SPAKE2+. KC-SPAKE2 is essentially the PFS-SPAKE2 protocol [14], and Shoup's [94] contribution is to prove the security of KC-SPAKE2 in the **Universal Composable** (**UC**) framework. KC-SPAKE2+ incorporates standard key confirmation flows in SPAKE2+ [93].

Inspired by SPAKE2 [37], Pointcheval and Wang [50] proposed the TBPEKE and VTBPEKE protocols. Both protocols achieve user authentication to the server with three messages. In 2017, Smyshlyaev [95] specified the Security Assessment Standardized PAKE protocol (i.e., SESPAKE). Both parties use password-based coefficients multiplied by one of the predefined elliptic curve points (i.e., $\mathsf{H}(ID_U \| pw)^*Q$) to mask the Diffie-Hellman key exchange. At CRYPTO'19, Bradley et al. [58] proposed the saPAKE-BB protocol that satisfies the notion of strong asymmetric PAKE [7].

3.4 Type-IV: Password as an Element of Multiplication Operation

At EUROCRYPT'00, Boyko et al. [35] proposed a PAK protocol. Compared to AuthA [44], PAK only relies on random oracles, while AuthA relies on random oracles and ideal block ciphers. PPK and PAK-X are two variants of PAK: PPK can achieve implicit authentication within two rounds; PAK-X requires the server to store a verifier for the password and is resilient to server compromise. Then, MacKenzie [65] proposed a revised PAK protocol called PAK-R, with a client-side computation only half that of PAK [35]. PAK-RY is an asymmetric PAKE protocol used to replace PAK-X [35]: the first two messages are the same as PAK-X, while in the third message, the client of PAK-RY simply proves knowledge of the password. Later, MacKenzie [66] proposed a general method for making PAK protocols (e.g., PAK, PPK) resilient to server compromise. Applying this method to PAK [35] resulted in the PAK-Z protocol. But Gentry et al. [96] pointed out that in fact, PAK-Z is not resilient to server compromise and then proposed an improved version, i.e., PAK-Z+.

Table 5. PAKE Protocols using the Password as an Element of Multiplication Operation (Type-IV)[†]

Protocol	Year	Reference	Sym./Asym.	Hash function	User→Server	Server→User
McQuoid et al.	2020	ACM CCS [99]	Symmetric	$\mathsf{H}: \{0,1\}^* \to E$	$X^* = g^x \cdot \mathsf{H}(pw\|r_U)^{-1}$	$Y^* = g^y \cdot \mathsf{H}(pw\|r_S)^{-1}$
SAE	2008	SensorComm [97]	Asymmetric	$\mathsf{H}: \{0,1\}^* \to E$	$X^* = x \cdot \mathsf{H}(ID_U\|ID_S\|pw)$	$Y^* = y \cdot \mathsf{H}(ID_U\|ID_S\|pw)$
PAK-Z+	2005	IEEE P1363 [96]	Asymmetric	$\mathsf{H}: \{0,1\}^* \to \mathbb{G}$	$X^* = g^x \cdot \mathsf{H}(ID_U\|ID_S\|pw)$	$Y = g^y$
PAK-Z	2002	IEEE P1363 [66]	Asymmetric	$\mathsf{H}: \{0,1\}^* \to \mathbb{G}$	$X^* = g^x \cdot \mathsf{H}(ID_U\|pw)$	$Y = g^y$
PAK-RY	2001	CT-RSA [65]	Asymmetric	$\mathsf{H}: \{0,1\}^* \to \{0,1\}^\eta$	$X^* = g^{x_1} \cdot x_2{}^q \cdot \mathsf{H}(ID_U\|ID_S\|g^{\mathsf{H}(minID\|max1D\|pw)})$	$Y = g^y$
PAK-R	2001	CT-RSA [65]	Symmetric	$\mathsf{H}: \{0,1\}^* \to \{0,1\}^\eta$	$X^* = g^{x_1} \cdot x_2{}^q \cdot \mathsf{H}(ID_U\|ID_S\|pw)$	$Y = g^y$
PAK-X	2000	EUROCRYPT [35]	Asymmetric	$\mathsf{H}: \{0,1\}^* \to \{0,1\}^\eta$	$X^* = g^x \cdot \mathsf{H}(ID_U\|ID_S\|pw)^r$	$Y = g^y$
PPK	2000	EUROCRYPT [35]	Symmetric	$\mathsf{H}: \{0,1\}^* \to \{0,1\}^\eta$	$X^* = g^x \cdot \mathsf{H}(ID_U\|ID_S\|pw)^r$	$Y^* = g^y \cdot (\mathsf{H}(ID_U\|ID_S\|pw)^r)$
PAK	2000	EUROCRYPT [35]	Symmetric	$\mathsf{H}: \{0,1\}^* \to \{0,1\}^\eta$	$X^* = g^x \cdot \mathsf{H}(ID_U\|ID_S\|pw)^r$, where $p = rq + 1$	$Y = g^y$

[†]The protocol message is usually the product of an ephemeral Diffie-Hellman public key with a hash of the password. g is a generator of a subgroup of $\mathbb{Z}_q^*$ of size q, which is public to all parties. $p = rq + 1$ for some value r co-prime to q. $\{x, x_1, x_2, r_U\}$ and $\{y, r_S\}$ are the ephemeral secret keys selected by the user U and the server S, respectively. $minID/maxID$ is the min/max value between the user's identity ID_U and the server's identity ID_S.

At PKC'04, Bresson et al. [63] for the first time proved that the product of the message with a hash of the password (e.g., $X^* = g^x \cdot \mathsf{H}(pw)$) is secure. In the same work, they also proposed the OMDHKE protocol, which provides resistance to denial-of-service attacks. Then, Harkins [97] proposed the **Simultaneous Authentication of Equals** (**SAE**) protocol. SAE has been implemented for IEEE 802.11-based mesh networks. SAE requires a finite cyclic group: prime modulus groups or elliptic curve groups [98]. At CCS'20, McQuoid et al. [99] proposed a symmetric PAKE with optimal round complexity and computational efficiency: one round to achieve implicit authentication, and only two exponentiations per party. They employ the primitive of programmable once public function.

3.5 Type-V: Password as a Mask for Messages

At NDSS'98, Wu [10] introduced the **Secure Remote Password** (**SRP**) protocol, where the server stores a verifier derived from the user's password, namely $V = g^{\mathsf{H}(ID_U\|pw\|salt)}$. Notably, the SRP protocol uses a mathematical expression $Y^* = g^y + V$, a novel construction that has never appeared in previous schemes. A user needs to compute $g^y \leftarrow Y^* - g^{\mathsf{H}(ID_U\|pw\|salt)}$ to prove the knowledge of the password. Wu [21] pointed out two issues with the SRP protocol [10]. The first is a security concern. An attacker impersonates the server to send the exchanged key value, and replaces the original verifier and the ephemeral key value with two guessed passwords, thereby testing two possible passwords in an impersonation attack. The second one is an operational problem. That is, the server needs to wait for the user to send the exchanged key value before sending the random parameter u. To solve these two problems, Wu [21] proposed the SRP-6 protocol as an improved version of SRP [10], which uses the mathematical expression $Y^* = g^y + k \cdot V$ (e.g., $k = 3$) to eliminate symmetry, and redefines $u = \mathsf{H}(X^*\|Y^*)$ to remove the message order limitation.

At NDSS'01, Kwon [39] proposed the AMP protocol, where the server stores $V = g^{\mathsf{H}(ID_U\|pw)}$. The server sends $Y^* = (g^x \cdot V)^y$ to the user and requires the user to compute $g^y \leftarrow (Y^*)^{1/(x+\mathsf{H}(ID_U\|pw))}$. The AugPAKE protocol [100] adopts a design strategy similar to AMP [39]. That is, the server requires the user to compute $g^{\mathsf{H}(y)} \leftarrow (Y^*)^{1/(x+pw\cdot\mathsf{H}(ID_U\|ID_S\|X))}$. For more details, see Table 6.

At SPW'08, Hao and Ryan [16] proposed the password-authenticated key exchange by juggling (J-PAKE) protocol. J-PAKE was later standardized in ISO/IEC 11770-4 (2017) [88] and formally published in RFC 8236 [101]. In particular, J-PAKE was implemented in OpenSSL and OpenSSH as an experimental authentication protocol [102]. In addition, it was used by Firefox Sync version 1.1 [19], where it continues to serve as the foundation for the Pale Moon Sync service [103]. In Hao-Ryan's work [16], participants first exchange their generated ephemeral public keys, such as standard Diffie-Hellman key exchange, and then combine these ephemeral public keys with a

Table 6. PAKE Protocols using the Password to Mask Protocol Messages (Type-V)†

Protocol	Year	Reference	Sym./Asym.	User→Server	Server→User	Session key material
AugPAKE	2010	RFC 6628 [100]	Asymmetric	$X = g^{x}$	$Y^{*} = g^{x \cdot \mathrm{H}(y)} \cdot g^{pw \cdot \mathrm{H}(ID_U \| ID_S \| X) \cdot \mathrm{H}(y)}$	$\sigma = g^{\mathrm{H}(y)}$
J-PAKE	2008	SPW [16]	Symmetric	$X^{*} = (g^{y_1} g^{y_2} g^{x_1})^{x_2 \cdot pw}$	$Y^{*} = (g^{x_1} g^{x_2} g^{y_1})^{y_2 \cdot pw}$	$\sigma = g^{(x_1+y_1) \cdot x_2 \cdot y_2 \cdot pw}$
SRP-6	2002	IEEE P1363 [21]	Asymmetric	$X = g^{x}$	$Y^{*} = g^{y} + 3 \cdot g^{\mathrm{H}(ID_U \| pw \| salt)}$	$\sigma = g^{y \cdot (x + \mathrm{H}(X^{*} \| Y^{*}) \cdot \mathrm{H}(ID_U \| pw \| salt))}$
AMP	2001	NDSS [39]	Asymmetric	$X = g^{x}$	$Y^{*} = g^{xy} \cdot g^{\mathrm{H}(ID_U \| pw) \cdot y}$	$\sigma = g^{y \cdot (x + \mathrm{H}(ID_U \| X^{*} \| Y^{*}))}$
SRP	1998	NDSS [10]	Asymmetric	$X = g^{x}$	$Y^{*} = g^{y} + g^{\mathrm{H}(ID_U \| pw \| salt)}$	$\sigma = g^{y \cdot (x + u \cdot \mathrm{H}(ID_U \| pw \| salt))}$
CHIP	2020	CRYPTO [70]	Asymmetric	$X = g^{x}, A = g^{a}$	$Y = g^{y}, B = g^{b}$	$\sigma = g^{(a+x+\mathrm{H}(pw) \cdot \mathrm{H}(ID_U \| A)) \cdot (b+y+\mathrm{H}(pw) \cdot \mathrm{H}(ID_S \| Y))}$

†In Figure 2(e), we use SRP [10] as an example (see the symbol "+" in Figure 2(e)) to show the protocol flows of Type-V. This table shows all methods to mask protocol messages. g is a generator of a subgroup of $\mathbb{Z}_q^*$ of size q, which is public to all parties. $\{x, x_1, x_2\}$ and $\{u, y, y_1, y_2\}$ are the ephemeral secret keys selected by U and S, respectively.

Table 7. PAKE Protocols using the Password as a Parameter of Smooth Projective Hashing (Type-VI)†

Protocol	Year	Reference	Sym./Asym.	Session key material	Building blocks
Li et al.	2022	IEEE TDSC [33]	Asymmetric	$\sigma = H_k(c', \mathrm{H}(salt \| pw)) + H_{k'}(c, r)$	SPHF + Micciancio-Peikert enc. [106]
H-VPAKE	2021	Comput. J. [12]	Asymmetric	$\sigma = H_k(c', pw) + H_{k'}(c, r)$	SPHF + Cramer-Shoup enc. [107]
Ren-Gu	2021	ICICS [29]	Asymmetric	$\sigma = \mathrm{Rec}(\mathbf{s}_C{}^{T}\mathbf{y}_S, v)$	SPHF + Error reconciliation
LWE-PAKE	2020	PKC [28]	Symmetric	$\sigma = \pounds(\mathbf{E}'^{T}\mathbf{A}\mathbf{s}, v)$	SPHF + Key reconciliation
Zhang-Yu	2017	ASIACRYPT [32]	Symmetric	$\sigma = H_k(u', pw) + H_{k'}(u, pw)$	SPHF + Peikert enc. [108]
Katz-Vaikuntanathan	2011	TCC [31]	Symmetric	$\sigma = H_k(label', c', pw) + H_{k'}(label, c, pw)$	SPHF + Naor-Yung enc. [109]
Groce-Katz	2010	ACM CCS [56]	Symmetric	$\sigma = H_k(c', pw)$	SPHF + Kurosawa-Desmedt enc. [110]
Katz-Vaikuntanathan	2009	ASIACRYPT [67]	Symmetric	$\sigma = H_k(label', c', pw) + H_{k'}(label, c, pw)$	SPHF + Peikert enc. [108]
Canetti et al.	2005	EUROCRYPT [104]	Symmetric	$\sigma = H_{k'}(c', pw) + H_{k'}(c, pw)$	SPHF + Shoup enc. [111]
Gennaro-Lindell	2003	EUROCRYPT [41]	Symmetric	$\sigma = H_k(c', pw) + H_{k'}(c, pw)$	SPHF + Cramer-Shoup enc. [107]

†We present the session key material σ in order to intuitively show the characteristics of session keys, which are different from Types I~V. Due to space constraints, the parameters will not be explained one by one. A SPHF is a collection of a keyed function $\{H_k : X \to \{0,1\}^n\}_{k \in K}$, along with a projection function $\alpha : K \to S$, satisfying notions of correctness and smoothness [31]. $\pounds$ is the key reconciliation scheme used in [28]. The error reconciliation scheme used in [29] contains a conciliate function Con(·) and a reconcile function Rec(·).

shared password in an additional round. The technique adopted by the J-PAKE protocol is called "public key juggling", that is, rearranging the public keys in a particular way, and encrypting the data by introducing quadratic terms on the exponents in a verifiable way.

3.6 Type-VI: Password as a Parameter of Smooth Projective Hashing

At EUROCRYPT'03, Gennaro and Lindell [41] found a novel application of SPHF for password-based key exchange. Specifically, the client sends $\{c, s\}$ and the server sends $\{c', s'\}$, where c and c' are randomly generated commitments of pw, s and s' are the corresponding projection keys. Then, based on the (approximate) correctness and smoothness of SPHF, both parties then compute the shared session key material $H_k(c', pw) + H_{k'}(c, pw)$. Later, Canetti et al. [104] found that when simulating the Gennaro-Lindell protocol [41], it is impossible to ensure that the key computed by the adversary is the same as the key obtained from the functionality. To overcome this problem, Canetti et al. [104] added a zero-knowledge proof to Gennaro-Lindell's work [41]. Katz and Vaikuntanthan [67] introduced a CCA-secure (i.e., chosen-ciphertext attack secure) public-key encryption scheme based on the **Learning With Errors** (**LWE**) assumption, and then incorporated it into the Gennaro-Lindell framework [41] to present a generic three-round PAKE protocol. Subsequently, Katz and Vaikuntanathan [31] introduced a one-round lattice-based PAKE protocol by combining the **Non-Interactive Zero-Knowledge** (**NIZK**) and the **Approximate Smooth Projective Hashing** (**ASPH**). Table 7 summarizes the representative PAKE protocols.

At CCS'10, Groce and Katz [56] proposed a new framework that relies on potentially weaker assumptions than the Gennaro-Lindell framework [41]: (1) A CCA-secure encryption scheme, and (2) A CPA-secure (i.e., chosen-plaintext attack security) encryption scheme equipped with an associated smooth projective hash function. Then, Zhang and Yu [32] proposed a two-round

lattice-based PAKE protocol by combining splittable public-key encryption and an adaptive ASPH. Shin et al. [12] further proposed a two-round verifier-based PAKE using tamper-proof hardware. Recently, Li et al. [33] proposed an asymmetric PAKE protocol that employs smooth projective hash functions together with commitment-based password hashing within a lattice-based cryptographic setting. At PKC'20, Jiang et al. [28] put forward two PAKE protocols based on the plain LWE and the Ring-LWE assumptions, which do not require any CCA-secure encryption or its variant. Later, Ren and Gu [29] proposed an asymmetric PAKE based on the Module-LWE assumption, which is a horizontal extension of the PAK protocol [66] in the field of module lattice.

3.7 Type-VII: Others

Three PAKEs [51, 68, 69] belong to *passwords as bit strings*. In addition, Cremers et al. [70] proposed two identity-based PAKEs that belong to *password as the private key of a trusted third party* (see detailed analysis in Section 3.7 of the full version of our article http://bit.ly/4nGwdLW).

4 Further Classification of PAKE

Besides the design strategy, there are classified in several other ways for PAKE, and here we focus on the five most important ones: symmetry, number of participants, hardness assumptions, security goals, and round complexity. Each classification is described as follows: (1) According to symmetry, PAKE can be divided into symmetric PAKE (e.g., [16, 20, 37, 99]) and asymmetric PAKE (e.g., [8, 11, 21, 94]); (2) According to the number of participants, PAKE can be divided into two-party (e.g., [8, 11]), three-party (e.g., [24, 75]), and multi-party (e.g., [34, 105]) PAKE; (3) According to the hardness assumption, PAKE can be divided into classic PAKE (e.g., [16, 20, 72, 73]) and quantum-safe (e.g., [28, 29]) PAKE; (4) According to the security goals, PAKE can be divided into forward-secure PAKE (e.g., [14, 57]), provably secure PAKE (e.g., [7, 8]), anonymous PAKE (e.g., [26, 27]), and so on; (5) According to the number of rounds, PAKE can be divided into one-round PAKE (e.g., [37, 91]), two-round PAKE (e.g., [58, 63], three-round PAKE (e.g., [11, 94]), and others.

4.1 Symmetric and Asymmetric PAKE

To provide a clearer conceptual view we give the following general definition. *Symmetric PAKE* is a PAKE in which both parties play identical roles (e.g., [15, 16, 37, 94]). *Strong* ***symmetric PAKE (sPAKE)*** is a symmetric PAKE protocol that resists to pre-computation attacks (e.g., CRISP [8]). *aPAKE* is designed to remain secure even if the server is compromised (e.g., [10, 11, 21, 35]). ***Strong asymmetric PAKE (saPAKE)*** is an asymmetric PAKE protocol that is resistant to pre-computation attacks (e.g., OPAQUE [7] and saPAKE-BB [58]).

4.1.1 Symmetric PAKE protocol. Bellovin and Merritt [13] were the first to propose a symmetric PAKE, named EKE. After that, many famous symmetric PAKE protocols have been proposed, such as SPEKE [15], SPAKE2 [37], and J-PAKE [16]. Most symmetric PAKE protocols (except CRISP [8]) assume that both participants hold the password in the clear. The disadvantages of such protocols (e.g., [13, 15, 37]) are inherent. That is, an attacker who corrupts one party can perform key compromise impersonation attacks [7, 38]. Recently, Cremers et al. [70] proposed two symmetric PAKE protocols (i.e., CHIP and CRISP) which avoid storing plaintext passwords by instead maintaining structured password files. Although CHIP and CRISP are symmetric PAKE protocols, they provide security upon server compromise. In particular, CRISP resists pre-computation attacks, which is the highest security that asymmetric PAKE protocols (e.g., [7, 58]) can provide.

Moreover, motivated by the advantages of asymmetric PAKE, Gentry et al. [78] proposed a method (called the Ω-method) which convert any UC secure symmetric PAKE protocol into an asymmetric PAKE protocol by adding two additional message rounds and a signature computation/verification.

Then, Hwang et al. [9] reduced the overhead to a single additional message round. After that, Hesse [112] presented a formal UC-security proof of the Ω-method.

4.1.2 Asymmetric PAKE protocol. aPAKE protocols work in the more common client-server setting, where the server stores a mapping of the password [38, 78]. After EKE [13] was published, Bellovin and Merritt [38] soon proposed its asymmetric version, named A-EKE, to prevent direct exposure of passwords upon server compromise. However, if the mapping is deterministic (e.g., a hash of the password), attackers can compute them offline in advance, and then find passwords almost instantaneously when breaking into the server.

To address this problem, Jareki et al. [7] introduced a *strong* asymmetric PAKE notion and provided a general construction (i.e., OPAQUE). Later, Cremers et al. [70] further summarized four methods to construct a password file in asymmetric PAKE protocols. We find that there are at least six types of password files: (1) Hashed password, e.g., A-EKE [38]; (2) Hashed password with identities, e.g., SNAPI-X [73]; (3) Hashed password with a public salt, e.g., VTBPEKE [50]; (4) Hashed password with identities and a public salt, e.g., SRP [15]; (5) Hashed password with a secret salt, e.g., saPAKE-BB [58]; (6) Hashed password with identities and a secret salt (no examples). A plausible reason why Cremers et al. [70] only summarized four types is that, if a secret salt is used, the public identifier does not affect the resistance to pre-computation attacks.

4.2 Two-party, Three-party, and Multi-party PAKE

4.2.1 Two-party PAKE Protocol. The number of two-party PAKE protocols is the largest (over 70) in the literature. Therefore, it is difficult to analyze all relevant typical protocols based solely on the concept of "two-party". The research focus within two-party PAKE is diverse. For example, as we discussed in Section 4.1, researchers focus on the impact of the eventual failure of the server and design symmetric, asymmetric, strong symmetric, and strong asymmetric PAKE protocols. There are other examples: Katz et al. [57] and Becerra et al. [14] focus on satisfying forward secrecy; Gentry et al. [96] and Abdalla et al. [113] focus on providing provable security; Hwang et al. [9] and Katz-Vaikuntanathan [31] focus on optimizing protocol rounds. At EUROCRYPT'22, Cramer et al. [107] presented two two-party PAKE protocols (i.e., aEKE-HMQV and OKAPE-HMQV) with minimal computational cost and minimal round complexity. In short, the field of two-party PAKE is highly active, with numerous outstanding works being developed.

4.2.2 Three-party PAKE Protocol. **Three-party PAKE (3PAKE)** protocols have emerged to meet the demands of practical applications. As discussed earlier, **two-party PAKE** (**2PAKE**) protocols are well-suitable for the standard client-server setting. But in large-scale network communication environments, the 2PAKE model is no longer applicable. The requirement for each client to remember the passwords of all its communicating parties seriously affects the storage capacity and scalability of the client. To address this issue, 3PAKE protocols are developed as a natural and effective solution. A trusted third party (which acts as the server) is introduced to centrally keep the passwords for all users in 3PAKE protocols. Thus, each user only needs to share one password with the server. In addition to traditional two-user settings, the 3PAKE protocol can also be applied in the wireless sensor setting and the two-server setting. In particular, 3PAKE protocols suffer from the threat of password guessing attacks as well, following a recurring 'breakfixbreakfix' cycle of design and revision, as partially illustrated in the bottom right corner of Figure 1.

In 1995, Steiner et al. [75] proposed a three-party PAKE for the first time. The PAKE of [75] is an extension of EKE [13]. Abdalla et al. [24] formalized a security model for 3PAKE protocols. Then, Lu and Cao [114] extended the SPAKE2 [37] into a three-party setting without the requirement for any server's public key. Later, Huang [115] proposed a new 3PAKE without the server's public key, which is claimed to be more efficient than Lu-Cao's PAKE [114]. Then Yoon and Yoo [116] pointed

out that Huang's 3PAKE [115] is vulnerable to undetectable online guessing attacks and offline guessing attacks. Subsequently, Tallapally [117] showed that Huang's protocol also suffers from an unknown key-share attack [115], and then proposed an alternative protocol with four message transmission rounds, which was breached by Farash and Attari [118]. Lu et al. [119] presented an improved 3PAKE to erase the security flaws found in [118]. However, Chen et al. [120] found that the 3PAKE proposed in [119] is still vulnerable to offline password attacks. Thus, a fundamental challenge with the PAKE is to resist offline guessing attacks, regardless of being two- or three-party.

4.2.3 Multi-party PAKE Protocol. Multi-party PAKE is also known as the **group password-authenticated key exchange** (**GPAKE**). Distributed group communication applications, such as remote teaching and multimedia conferencing, motivate the need for secure and practicable multi-party authentication protocols. Generally, GPAKE was originally introduced as an extension of two-party PAKE. However, large-scale communication and unreliable data transmission put forward more requirements on the GPAKE. On the one hand, considering the uncertainty of the number of participants in the application scenario, GPAKE should achieve constant rounds [121]. Namely, the number of communication rounds is independent of the cardinality of group members, so it will not increase linearly or logarithmically. On the other hand, GPAKE should support dynamic updates of members. In group communication, newly joined members should not have access to previous conversation messages, and quit members should not continue to receive new messages.

Moreover, GPAKE protocols proposed in the literature can be mainly divided into two categories depending on the presence of a trusted third party (TTP). In the GPAKE protocol with a trusted third party (also called a central server) in the system, each client shares an independent password with the server. Then, when users wish to create a session key, they resort to a trusted third party for authenticating each other. For example, an authentication server was added in the GPAKE protocol proposed by Abdalla and Pointcheval [105]. This protocol combines the **Completely Automated Public Turing Test to Tell Computers and Humans Apart** (**CAPTCHA**) technique and federated signature verification to thwart undetectable online password guessing attacks. In contrast, in the GPAKE protocol without any trusted third party, all participants achieve authentication in a self-organized manner and collaborate to calculate the session key. A typical GPAKE protocol without a trusted server was proposed by Wan et al. [34]. Which enhances computational efficiency by integrating the basic Diffie-Hellman exchange and the hierarchical key tree structure.

4.3 Hardness Assumptions of PAKE

4.3.1 PAKE Based on the Discrete Logarithm Problem. Diffie and Hellman [85] for the first time proposed a one-way trapdoor function with excellent asymmetry: modular exponentiation, and used it to demonstrate the famous cryptographic protocol: the Diffie-Hellman key exchange protocol. The classical Diffie-Hellman key exchange provides no authentication at all, and only focuses on the establishment of the session key. The confidentiality of the shared key in [85] is guaranteed by the **Computational Diffie-Hellman** (**CDH**) problem. Then Miller [122] and Koblitz [123] introduced the elliptic curves over finite field, respectively. ElGamal [124] further proposed a public key cryptosystem. Notably, ECC offers a significant advantage in key length efficiency. Under the same security strength, its key strength is much smaller than that of the integer-factorization based cryptography, implying smaller bandwidth and storage requirements [125].

The seminal work applying the above problems to password authentication protocols is Bellovin and Merritt [13]. Since then, researchers have devoted considerable effort to constructing the security of the PAKE protocol on the intractability of solving the **discrete logarithm** (**DL**) problem. In addition to the wide application in PAKE protocols, these hardness problems are also common in two-factor or multi-factor authentication protocols. On the other hand, the Diffie-Hellman key

exchange technique is one of the promising methods to achieve perfect forward secrecy [126], which makes it more attractive to researchers. Some classic protocols (e.g., [15, 16, 21, 39]) have also been included in international standards ISO/IEC 11770-4 [88]. Among these standardized PAKE protocols, AMP [39], J-PAKE [16] and P-SPEKE [20] work in both the DL and EC settings, but SRP-6 [21] only works in the DL setting. The underlying reason is that the mathematical expression $Y^* = g^y + 3 \cdot g^{\mathrm{H}(ID_U \| pw \| salt)}$ used in SRP-6 (see Table 6) cannot be defined in the EC setting.

4.3.2 PAKE Based on the Integer Factorization Problem. The best-known **integer factorization** (**IF**) problem is the RSA, named after its inventors Rivest, Shamir and Adleman [127]. The security of RSA lies in the difficulty of computing the eth root of a ciphertext c ($c = m^e \bmod N$) modulo a composite integer N. As a relatively mature public key cryptosystem, RSA has been employed in password-authenticated key exchange as early as the seminal work of Bellovin and Merritt [13]. The basic idea of constructing a PAKE protocol based on RSA consists of three steps. First, the sender generates and sends the RSA public key (n, e). Then the receiver uses the public key to encrypt the key material with the shared password as a parameter. According to the RSA algorithm, the party accepting the public key (n, e) should check the validity of the equation $\gcd(e, \varphi(n)) = 1$.

The purpose of verifying this equality is to ensure that regardless of whether the password parameter is correct, the ciphertext encrypted by RSA remains an e-residue. Thus, an attacker cannot rule out wrong passwords in one attempt. Authors generally take other measures to avoid a direct primality test, such as requiring exponent e to be greater than modulus n [73], or using multiple RSA encryption [36], thereby reducing the computational cost (see Section 3.1.2). The final phase nvolves decryption, often followed by a key confirmation step to achieve explicit authentication. As shown in Table 2, few RSA-based PAKE protocols have been proposed recently, and similar situations occur in two-factor and multi-factor password authentication. This trend can be reasonably explained in terms of security and usability: (1) The RSA-based PAKE protocol is vulnerable to e-residue attacks and does poorly in forward secrecy (see Table 9). By contrast, these security issues are rare in DL-based PAKE protocols; (2) Massive modular exponentiation and multiplication inverse make RSA-based PAKE protocols insufficiently efficient for practical use.

4.3.3 PAKE Based on the Lattice Problem. Regev [128] reduced the worst-case lattice problems (e.g., GAPSVP and SIVP) to the LWE problem for the first time, and presented a classical public-key cryptosystem whose security is based on the hardness of the **Learning With Errors** (**LWE**) problem. Compared to traditional DL problems and the IF problem, the hard problems over lattices can resist quantum computing attacks [33, 129]. The quantum computer has faster parallel computing and simulation capabilities, allowing them to solve the (elliptic-curve) discrete logarithm and integer factorization in polynomial time [130]. In contrast, the lattice problem is a mathematically hard problem that cannot be solved by a quantum computer, so it can provide security in the context of quantum-safe cryptography. The lattice-based PAKE protocol is currently considered to be the most promising research direction of quantum-resistant cryptographic protocols.

Interestingly, a research route shows the pursuit of the optimal round of the PAKE protocol over lattices. Specifically, at ASIACRYPT'09, Katz and Vaikuntanathan [67] introduced a generic *three*-round symmetric PAKE protocol over lattices using a public-key encryption scheme and SPHF. Then Zhang and Yu [32] proposed a *two*-round symmetric PAKE based on the hardness of the LWE assumption. Later, Li et al. [33] proposed a *one*-round asymmetric PAKE over lattices, eliminating the need for NIZK and the random-oracle assumption.

4.4 Security Goals of PAKE

4.4.1 Forward-secure PAKE Protocol. Forward secrecy is a desirable security property of PAKE protocols, ensuring that each session key remains independent of the long-term key. Forward

secrecy can be distinguished as **Perfect Forward Secrecy** (**PFS**) and wFS according to whether an adversary actively participates. The commonly referred to forward secrecy, or perfect forward secrecy, means that even if the long-term private key of one or more parties is stolen by an attacker, the security of the session key established in the previous session is still not threatened [45]. But weak forward secrecy only protects the session key created without the active participation of attackers after the long-term private key material is compromised [40]. Recall that the capability of C-6 in Table 1 captures the leakage of long-term private keys, corresponding to forward secrecy here. To formally capture this capability within the security model, corrupt oracles are introduced into the security model to simulate it (see Section 2.2).

The notion of forward secrecy was first proposed by Diffie [131], and was formally applied to password authentication protocols in [44] and [57]. Subsequently, Park et al. [126] proposed two methods to achieve forward secrecy in the PAKE protocol. One method is based on the classic Diffie-Hellman key exchange technique, and the other is based on the confidentiality of a random number selected by the server. Similarly, Ma et al. [132] put forward a general principle, namely that at least two exponentiations (respectively point multiplication in elliptic curve cryptosystems) operations conducted on the server side are necessary for achieving forward secrecy. In addition to basic cryptographic components, some researchers have focused on the relationship between protocol flow and forward secrecy. For example, Krawczyk [40] pointed out that a two-flow PAKE protocol using a public-key primitive as an authentication mechanism cannot achieve perfect forward secrecy, and proposed the notion of wFS. Following this idea, Becerra et al. [14] proved that the SPAKE2 protocol [37] only satisfies wFS, and proposed the PFS-SPAKE2 protocol by incorporating key-confirmation codes into the original SPAKE2 protocol. Recently, forward secrecy has become a basic goal of security protocol design. For example, all public-key based key exchange mechanisms in TLS 1.3 have provided this property [133].

4.4.2 Provably Secure PAKE Protocol. When a password authentication protocol is proposed, authors typically provide a security proof to demonstrate its soundness and practical viability (e.g., [7, 8, 58]). Most PAKE protocols published earlier performed only heuristic and informal security cryptanalysis. As time goes on, the field has shifted toward the adoption of formal security analysis methods [134], such as complexity-based provable security [44], symbol-based logic [20], and automated formal verification tools [135]. Among them, complexity-based provable security in cryptography refers to reducing the task of breaking the protocol to solving hard problems (e.g., CDH, RSA) or cryptographic primitives (e.g., the discrete logarithm problem, the integer factorization problem) [136]. Because the advantage of computers in solving these problems in polynomial time is negligible, the possibility of an attacker cracking the PAKE protocol is also negligible.

A prerequisite for a PAKE protocol to be provably secure is a formal, well-defined security model, such as the *random oracle model* and the *standard model*. Early provably secure protocols (e.g., KOY [90]) is mainly based on the standard model, where the simulation of the protocol is realized by constructing a special trapdoor function without random oracle assumptions. Conversely, the random oracle model assumes that some cryptographic hash functions are replaced by a publicly accessible random function (i.e., a random oracle). An adversary can only obtain the result of the hash function by querying the random oracle, instead of by calculating. In addition, the quantum random oracle is proposed to resist quantum computer attacks. The random oracle model requires that the emulator can record and query all queries executed by adversaries. However, unlike the classical setting, the adversary's query cannot be recorded and read, resulting in the ordinary random oracle being unable to work. To overcome this issue, researchers have proposed multiple methods (see [137, 138]) to build a scientific quantum random oracle model.

Moreover, *the **ideal cipher model (ICM)*** has also emerged as an important paradigm in the evolution of security model. Instead of having a publicly accessible random function, it assumes that there is a publicly accessible random block cipher (or ideal cipher). For any given key, all parties, including an adversary, can simultaneously perform encryption or decryption queries on the ideal block cipher. Coron et al. [139] further proved that the random oracle model and the ideal cipher model are inherently equivalent. Table 9 shows the security model adopted by representative PAKE protocols.

However, the traditional provable security theory ignores the security when multiple protocols are executed in combination. Canetti et al. [140] addressed this issue by proposing a new paradigm for defining the security of PAKE protocols, called **Universal Composable (UC)** security. This framework requires security to be assured, even when the cryptographic protocol is composed of any set of protocols. Commonly, the UC framework is combined with assumptions such as random oracles to provide rigorous security guarantees for a PAKE protocol. Recently, Abdalla et al. [141] introduced the lazy-extraction UC PAKE and relaxed UC PAKE functionalities, and showed several PAKE protocols (e.g., SPAKE2 [37], TBPEKE [50], and SPEKE [15]) which had been proven secure under game-based definitions, also realize the lazy-extraction UC PAKE functionality.

4.4.3 Anonymous PAKE Protocol. Anonymity was not considered in early PAKE protocols, and all participants were publicly identifiable (e.g., [15, 35, 38]). Concretely, the identifiers are sent in the clear, as part of the PAKE protocol. With the increasing awareness of privacy in society, anonymous PAKE protocols that can protect data access information are receiving more and more attention. An anonymous PAKE protocol simultaneously achieves identity authentication, session key establishment, and participant anonymity. From the perspective of participants, anonymity can be divided into user anonymity and server anonymity. User anonymity ensures that a protocol protects the user's identity and prevents behavioral tracing [45]. Specifically, an attacker can neither confirm the identity of the user nor distinguish whether any two sessions are from the same user. Server anonymity means that the password database information will not be disclosed to users. Except for whether the user password is stored in the server database, no information about the password or the set of valid passwords should leak to the other party [26]. It is worth noting that, beyond anonymous PAKE protocols, achieving any form of network anonymous communication inherently requires the encryption of data packets to hide the source and destination.

Generally, most anonymous PAKE protocols either require the client to remember an index associated with its password in the server's password database (e.g., [142, 143]), or require the client to store password-wrapped anonymous credentials (e.g., [27, 42, 43]). Considering the storage requirement of the client, Vasco et al. [26] proposed two anonymous PAKE protocols that assume the client holds only passwords. The first protocol leverages oblivious pseudo-random functions, and the second one builds upon a special type of identity-based encryption scheme. Vasco et al.'s work [26] breaks the constraints of the previous two types, and truly enables the construction of anonymous PAKE protocols using only passwords. In addition, public-key techniques are not only essential for constructing password authentication protocols that resist offline password guessing attacks [144], but also essentially indispensable to constructing an anonymous PAKE protocol [145]. Building an anonymous PAKE protocol using only basic symmetric cryptographic primitives (e.g., hash functions, symmetric encryptions, and XOR operations) appears highly implausible from both theoretical and practical standpoints.

4.5 Round Complexity of PAKE

The primary criteria for evaluating a PAKE protocol are its round complexity, computation overhead, and security. The research trend in PAKE design are largely driven by the pursuit of better trade-offs

Table 8. 23 PAKE Protocols with Minimal Round Complexity (i.e., Implicit Authentication in 1-round/2-flow; Unilaterally Explicit Authentication in 2-round/2-flow; Mutually Explicit Authentication in 3-round/3-flow)

Protocol	Year	Reference	R/F	Start	EA	Symm.	Client	Server	Assump.	Model
aEKE-HMQV	2022	EUROCRYPT [8]	1/2	-	None	**aPAKE**	1fb+1mvb	1fb+1mvb	gapDH	RO/IC
McQuoid et al.	2020	ACM CCS [99]	1/2	-	None	PAKE	1fb+1vb	1fb+1vb	DDH	RO
P-SPEKE	2018	IEEE TIFS [20]	1/2	-	None	PAKE	2vb	2vb	CDH	-
SPAKE2	2005	CT-RSA [37]	1/2	-	None	PAKE	3fb+1vb	3fb+1vb	CDH	RO
SPAKE1	2005	CT-RSA [37]	1/2	-	None	PAKE	3fb+1vb	3fb+1vb	CDH	RO
Kobara-Imai	2002	IEICE Trans. [91]	1/2	-	None	PAKE	2fb+1vb	2fb+1vb	DDH	Std
PPK	2000	EUROCRYPT [35]	1/2	-	None	PAKE	1fb+2vb	1fb+2vb	DDH	RO
OKAPE-HMQV	2022	EUROCRYPT [8]	2/2	S	C	aPAKE	1fb+1mvb	1fb+1mvb	gapDH	RO/IC
saPAKE-BB	2019	CRYPTO [58]	2/2	C	S	**saPAKE**	10fb+1bv	2fb+2vb	SDH, DDH	RO
OMDHKE	2004	PKC [63]	2/2	C	S	PAKE	1fb+1vb	1fb+1vb	CDH	RO
AuthA	2000	IEEE P1363 [71]	2/2	C	S	aPAKE	1fb+1vb	1f+1vb	CDH	RO/IC
KHAPE	2021	CRYPTO [11]	3/3	S	C+S	aPAKE	1fb+1mvb	1fb+1mvb	gapCDH	IC
KC-SPAKE2+	2020	TCC [94]	3/3	C	C+S	aPAKE	2fb+2vb	2fb+2vb	CDH	RO
KC-SPAKE2	2020	TCC [94]	3/3	C	C+S	PAKE	2fb+1vb	2fb+1vb	CDH	RO
OPAQUE	2018	EUROCRYPT [7]	3/3	C	C+S	**saPAKE**	1fb+2vb+1mvb	1fb+1vb+1mvb	OMDH	RO
PFS-SPAKE2	2018	ProvSec [14]	3/3	C	C+S	PAKE	2fb+1vb	2fb+1vb	CDH	RO
PAK-Z+	2005	IEEE P1363 [96]	3/3	C	C+S	aPAKE	1fb+1vb	1fb+1vb	CDH	RO
PAK-Z	2002	IEEE P1363 [66]	3/3	C	C+S	aPAKE	1fb+1vb	1fb+1vb	CDH	RO
PAK-RY	2001	CT-RSA [65]	3/3	C	C+S	aPAKE	2fb+2vb	1fb+3vb	CDH	RO
PAK-R	2001	CT-RSA [65]	3/3	C	C+S	PAKE	1fb+2vb	1fb+1vb	DDH	RO
PAK-X	2000	EUROCRYPT [35]	3/3	C	C+S	aPAKE	2fb+3vb	2fb+3vb	DDH	RO
PAK	2000	EUROCRYPT [35]	3/3	C	C+S	PAKE	1fb+2vb	1fb+2vb	DDH	-
M-EKE	1995	ACM SIGOPS [75]	3/3	C	C+S	PAKE	1fb+1vb	1fb+1vb	CDH	RO/IC

The PAKE protocol with the best overall security and performance in each round is highlighted in purple.

R/F: Round/Flow
EA: Explicit Authenticaiton
SDH: Strong Diffie-Hellman
C, S: Client, Server
IC: Ideal Cipher
fb: Fixed-based Exponentiation
CDH: Computational Diffie-Hellman
gapDH: gap Diffie-Hellman
RO: Random Oracle
vb: Variable-based Exponentiation
DDH: Decision Diffie-Hellman
aPAKE: Asymmetric PAKE
Std: Standard Model
mvb: Multi-exponentiation
OMDH: One-More Diffie-Hellman
saPAKE: Strong asymmetric PAKE

among these three criteria. For example, the saPAKE-BB protocol [58] attains the same level of strong asymmetric security as OPAQUE [7], and further reduces the computational cost and round complexity; The aEKE-HMQV protocol [70] achieves asymmetric security and computational cost as KHAPE [11], and further reduces round complexity (see Figure 3). Table 8 summarizes 23 PAKE protocols with the minimum round complexity.

Minimal round complexity. The number of rounds and implicit/explicit authentication are related. A one-round PAKE can only provide implicit authentication (e.g., SPAKE2 [93]). There are two methods to reduce round complexity: (1) Merge messages from different flows, e.g., M-EKE [75]; (2) Turn two ordered flows into unordered ones, e.g., P-SPEKE [20]. Specifically, the M-EKE [75] functionally equivalent to EKE [13], except that EKE has five rounds of communication and M-EKE has three rounds of communication. M-EKE [75] merges the fourth message of EKE (i.e., the key confirmation code from the server) into the second message, and the fifth message (i.e., the key confirmation code from the client) into the third message, thereby reducing the number of rounds. The P-SPEKE [20] only has two rounds (four flows): the first round is used for key exchange, and the second round is used for key confirmation. The messages sent by the client and server are completely symmetrical, and one doesn't need to wait for the other party to send its own message.

Minimal computation overhead. Generally, unauthenticated **key exchange** (**KE**), i.e., Diffie-Hellman, serves as the baseline for symmetric PAKE and aPAKE, namely, aPAKE ⇒ PAKE ⇒ KE [8, 11]. The computational cost of the Diffie-Hellman key exchange is one offline fixed-base and one variable-base exponentiation per party. Note that the computation costs in Table 8 only involve exponentiation operations and do not include hashing, encryption, and so on.

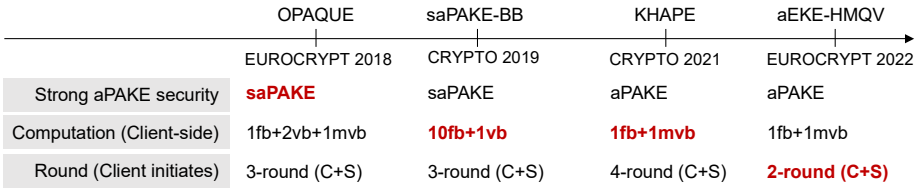


Fig. 3. An example that illustrates the relationship between security, computation overhead, and round complexity. The four PAKE protocols are OPAQUE [7], saPAKE-BB [58], KHAPE [11], and aEKE-HMQV [8]. For a fair comparison, we give the number of rounds required for each protocol to implement mutually explicit authentication. The value highlighted in red indicates the improvement of the current protocol over the previous one. fb = fixed-based exponentiation, where the base is fixed and precomputation techniques apply to accelerate computation (e.g., computing g^x for a fixed generator g); vb = variable-based exponentiation, where the base is determined at runtime and cannot be precomputed (e.g., computing g^x with $g = H(pw)$); mvb = multi-exponentiation, referring to computing multiple exponentiations (e.g., computing $g^x h^y$).

The computational cost of the five symmetric PAKE protocols (i.e., [20, 44, 63, 75, 99]) is comparable to that of the DiffieHellman key exchange. More specifically, AuthA [71] and M-EKE [75] employ password-encryption to Diffie-Hellman key exchange messages (see Table 2). OMDHKE [63] and McQuoid et al. [99] apply a mask generation function as the product of a Diffie-Hellman value with a hash of the password (see Table 5). P-SPEKE [20] applies a hash function, which uses the password to derive a secret group generator for performing Diffie-Hellman key exchange (see Table 3). The three asymmetric PAKE protocols (i.e., [8, 11]) achieve near-optimal computational efficiency of one offline fixed-base exponentiation and one multi-exponentiation (at the cost of 1.17 regular exponentiations) per party. More specifically, KHAPE [11], aEKE-HMQV [8], and OKAPE-HMQV [8] use (the hash value of) the password to the public key and/or private key of HMQV.

Strong aPAKE security. At EUROCRYPT'18, Jarecki et al. [7] strengthened the definition of aPAKE to eliminate vulnerabilities to pre-computation attacks (i.e., strong aPAKE security). OPAQUE [7] offers resistance to pre-computation attacks by employing OPRF; saPAKE-BB [58] employs a new form of conditional key **encapsulation mechanism** (**CKEM**).

To gain deeper insight into the interplay between security, computational cost, and round complexity, we use the latest four protocols as examples (see Figure 3). The most significant feature of OPAQUE [7] is its strong aPAKE security. However, the computational cost of OPAQUE is greater than that of unauthenticated key exchange. This is because OPAQUE uses both OPRF (1fb+1vb) and HMQV (1fb+1mvb). By employing CKEM, saPAKE-BB [58] uses more fixed-base exponentiation operations, thereby reducing the computational cost. KHAPE [11] improves performance by dispensing with the OPRF (hence also reducing security). Recently, aEKE-HMQV [8] achieves both minimal computational cost and minimal round complexity by using *one-pass* HMQV. In standard HMQV, $(a, A = g^a)$ and $(b, B = g^b)$ are long-term private-public key pairs for the client and server, respectively. But in one-pass HMQV, (b, B) plays a double role as a server's public key and its ephemeral Diffie-Hellman contribution. Therefore, even if an attacker decrypts B' using a candidate password pw' from a password dictionary, she cannot perform offline password guessing attacks. Because B is random every session and cannot be used by the attacker to rule out wrong passwords. As a result, the messages of users and servers in aEKE-HMQV [8] are synchronized.

5 Comparison

To demonstrate the practical effectiveness of our framework, we provide a comparative evaluation of 71 representative PAKE protocols in terms of security, usability, and efficiency. This assessment

is grounded in the security model proposed in Section 2.2 and 13 desirable properties proposed in Section 2.3. Table 9 offers a comparison of 71 PAKE protocols in terms of security and usability. Table 10 shows the efficiency comparison of 71 PAKE protocols. Note that the survey conducted in this article is not absolutely complete. A PAKE protocol is included in Tables 9 and 10 only because it is highly representative, or clarifies a certain design strategy in some way (see Section 3).

To select these 71 PAKE protocols, we proceed from two perspectives. First, we begin with a set of seed papers and trace subsequent improvement schemes proposed by researchers for these PAKE protocols. Seed articles are representative works that we know are related to password authentication, e.g., the seminal EKE protocol [13], the SRP protocol [10], and the SPEKE protocol [15]. Most of them are widely recognized in the cryptographic community and have even been incorporated into the international standard. Then, we trace back from excellent works in the past three years, such as the CRISP protocol [70] that is resilient to pre-computation attacks, and Vasco et al.'s [26] anonymous protocol that is truly password-only. We believe that recent works indicate the hot issues in PAKE protocols research and suggest future directions. To identify more works, we further examine PAKE protocols with special attributes from the perspective of protocol classification as discussed in Section 4. The resulting selection of protocols is organized chronologically, creating a timeline for PAKE protocol development. Furthermore, articles are identified by the name of the proposed PAKE protocol if available (e.g., SRP [10]), otherwise by the authors' name (e.g., Vasco et al. [26]). These 71 works are published in 59 venues. The top-3 venues for publishing articles on PAKE protocols are EUROCRYPT (8 articles), ASIACRYPT (8 articles), and ACM CCS (5 articles).

5.1 Comparison of Security and Usability

Our analysis of Table 9 follows the order from left to right, that is, from "Security model" to "U7 Round complexity". From the perspective of the security model, the use of UC frameworks increases over time: UC PAKE accounts for 17/25 in the past eight years. For example, Erwig et al. [68] present a security definition for asymmetric fuzzy PAKE in the UC model; Jarecki et al. [7] formalize strong asymmetric PAKE in the UC model; Gu et al. [11] present a formalization of the key-hiding AKE UC functionality. In addition to these new security definitions, the interaction with the environment in the UC model can capture the actual deployment of PAKE. Because the entire network activity consists of many executions, PAKE is just one of them (e.g., OPAQUE can be used in conjunction with TLS. For details on their combination, see [7]). This indicates that researchers pay more attention to the security of the protocol combination execution.

Diffie-Hellman (**DH**)-like assumptions (e.g., CDH, DDH, and SDH) constitute the security foundation of most PAKE protocols (76.1%=54/71). One reason DH-based assumption are prevalent is their simplicity (only one round of communication) and efficiency (only one offline fixed-base exponentiation and one online variable-based exponentiation per party). In addition, one can observe that lattice-based PAKE shows an upward trend, while RSA-based PAKE is decreasing. As we discussed above, the interest in the lattice-based PAKE protocol is largely due to its post-quantum security properties. RSA-based PAKE (e.g., [36, 72, 73]) tend to perform poorly in terms of transmission overhead, round complexity, and support for pre-computation (see Table 9).

Security analysis. Among the six security-related criteria, S1 (key agreement) and S2 (mutual authentication) are met by all 71 PAKE protocols. This is because key agreement and mutual authentication are the inherent nature of a PAKE protocol. There are two main methods of key establishment, namely, key agreement and key distribution. The inherent shortcoming of key distribution is the absolute trust in a single entity, so that if the entity is compromised, the security of the entire system will be destroyed. Hence, PAKE protocols commonly adopt key agreement to generate session keys, which is also more suitable for distributed environments or ad hoc

Table 9. Security and usability Evaluation of 71 Representative PAKE Protocols

Protocol		Reference	Security model	Cryptographic assumption	Participant	(Strong) (a)symmetric PAKE	Security													Usability							
							S1: *Key agreement*	S2: *Mutual authentication*	S3: *Explicit authentication*	S4: *Perfect forward secrecy*	S5: *User anonymity*	S6-1: *Resist. replay attacks*	S6-2: *Resist. man-in-the-middle*	S6-3: *Resist. offline guessing attacks*	S6-4: *Resist. undetec. online guessing*	S6-5: *Resist. unknown key-share*	S6-6: *Resist. parallel-session attacks*	S6-7: *Resist. known-key attacks*	S6-8: *Resist. key compromise impersn.*	S6-9: *Resist. pre-computation attacks*	S6-10: *Resist. quantum attacks*	U1: *Password friendly*	U2: *Login device friendly*	U3: *Support pre-computation*	U4: *Storage overhead*	U5: *Computation overhead*	U6: *Transmission overhead*	U7: *Round complexity*
KEM-PAKE	2025	[146]	UC, RO	DLWE	2-pary	aPAKE	●	●	●	●	○	●	●	●	●	●	●	●	●	○	●	●	○	●	●	○	○
LPKE-PAKE	2024	[147]	UC, RO	LWE	2-pary	PAKE	●	●	●	●	○	●	●	●	●	●	●	●	○	○	○	●	○	●	○	●	○
BarePAKE	2024	[79]	UC, RO/IC	CDH	n-pary	PAKE	●	●	●	●	○	●	●	●	●	●	●	●	○	○	○	●	●	●	○	○	○
SweetPAKE	2024	[80]	UC, RO	CDH	2-pary	PAKE	●	●	●	●	○	●	●	●	●	●	●	●	●	●	○	●	●	●	○	●	○
FuzzyPAKE	2023	[81]	UC, RO	gapDH	2-pary	PAKE	●	◐	●	●	○	●	●	●	●	●	●	●	◐	○	○	●	●	●	●	●	○
OKAPE-HMQV	2022	[8]	UC, RO/IC	gapDH	2-pary	aPAKE	●	●	◐	◐	○	●	●	●	●	●	●	●	◐	○	○	●	●	●	●	●	●
aEKE-HMQV	2022	[8]	UC, RO/IC	gapDH	2-pary	aPAKE	●	●	○	◐	○	●	●	●	●	●	●	●	◐	○	○	●	●	●	●	●	●
CRISP	2022	[70]	UC, GG	–	2-pary	sPAKE	●	●	○	◐	○	●	●	●	●	●	●	●	○	●	○	●	●	●	○	○	●
CHIP	2022	[70]	UC, RO	SCDH	2-pary	PAKE	●	●	○	◐	○	●	●	●	●	●	●	●	○	○	○	●	●	●	○	○	○
KHAPE	2021	[11]	UC, IC	gapCDH	2-pary	aPAKE	●	●	●	●	○	●	●	●	●	●	●	●	●	◐	○	●	●	●	●	●	○
H-VPAKE	2021	[12]	Std	–	2-pary	aPAKE	●	●	○	◐	○	●	●	●	●	●	●	●	◐	◐	○	●	●	●	○	○	○
Vasco et al.	2021	[26]	RO	–	2-pary	saPAKE	●	●	◐	◐	●	●	●	●	●	●	●	●	◐	●	○	●	●	●	○	●	○
KC-SPAKE2+	2020	[94]	RO, CRS	CDH	2-pary	aPAKE	●	●	●	●	○	●	●	●	●	●	●	●	○	◐	○	●	●	●	●	●	●
KC-SPAKE2	2020	[94]	RO, CRS	CDH	2-pary	PAKE	●	●	●	●	○	●	●	●	●	●	●	●	○	○	○	●	●	●	●	●	●
McQuoid et al.	2020	[99]	UC, RO	DDH	2-pary	PAKE	●	●	●	●	○	●	●	●	●	●	●	●	○	○	○	●	●	●	○	○	●
RLWE-PAKE	2020	[28]	Std	R-DLWE	2-pary	PAKE	●	●	●	●	○	●	●	●	●	●	●	●	○	○	●	●	●	●	○	○	●
LWE-PAKE	2020	[28]	Std	DLWE	2-pary	PAKE	●	●	●	●	○	●	●	●	●	●	●	●	○	○	●	●	●	●	○	○	●
Erwig et al.	2020	[68]	UC, IC, GG	–	2-pary	aPAKE	●	●	○	◐	○	●	●	●	●	●	●	●	◐	◐	○	●	●	●	○	○	○
AucPace	2019	[89]	UC	CDH	2-pary	aPAKE	●	●	●	●	○	●	●	●	●	●	●	●	◐	◐	○	●	●	●	○	○	○
saPAKE-BB	2019	[58]	UC, RO, GG	DDH, SDH	2-pary	saPAKE	●	●	◐	◐	○	●	●	●	●	●	●	●	◐	◐	○	●	●	●	○	○	●
Hwang et al.	2018	[9]	UC	CDH	2-pary	aPAKE	●	●	◐	◐	○	●	●	●	●	●	●	●	◐	○	○	●	●	●	○	○	○
Dupont et al.	2018	[69]	UC	CDH	2-pary	PAKE	●	●	○	●	○	●	●	●	●	●	●	●	○	○	○	●	●	●	○	○	○
OPAQUE	2018	[7]	UC, RO	OMDH	2-pary	saPAKE	●	●	●	●	○	●	●	●	●	●	●	●	◐	●	○	●	●	●	○	○	●
PFS-SPAKE2	2018	[14]	RO, CRS	CDH	2-pary	PAKE	●	●	●	●	○	●	●	●	●	●	●	●	○	○	○	●	●	●	●	●	●
P-SPEKE	2018	[20]	–	CDH	2-pary	PAKE	●	●	○	◐	○	●	●	●	●	●	●	●	○	○	○	●	●	○	●	○	●
VTBPEKE	2017	[50]	RO	GSDH	2-pary	aPAKE	●	●	◐	◐	○	●	●	●	●	●	●	●	◐	◐	○	●	●	●	○	○	○
TBPEKE	2017	[50]	RO	GSDH	2-pary	PAKE	●	●	○	◐	○	●	●	●	●	●	●	●	○	○	○	●	●	●	○	●	○
SESPAKE	2017	[95]	–	CDH	2-pary	aPAKE	●	●	●	●	○	●	●	●	●	●	●	●	○	◐	○	●	●	●	○	○	○
Zhang-Yu	2017	[32]	UC, Std	LWE	2-pary	PAKE	●	●	○	◐	○	●	●	●	●	●	●	●	○	○	●	●	●	●	○	○	○
Zhang et al.	2016	[27]	RO	q-SDH	2-pary	aPAKE	●	●	○	◐	●	●	●	●	●	●	●	●	◐	●	○	●	○	●	○	○	○
DE-PAKE	2016	[64]	RO	OMGDH	3-pary	saPAKE	●	●	●	●	○	●	●	●	●	●	●	●	●	●	○	●	●	●	○	○	○
Abdalla et al.	2015	[113]	RO, AA	DSDH	2-pary	PAKE	●	●	●	●	○	●	●	●	●	●	●	●	○	○	○	●	○	●	○	○	○
Patched SPEKE	2014	[87]	–	CDH	2-pary	PAKE	●	●	●	●	○	●	●	●	●	●	●	●	○	○	○	●	●	○	●	○	●
Canetti et al.	2012	[51]	UC, CRS	CDH	2-pary	PAKE	●	●	●	●	○	●	●	●	●	●	●	●	○	○	○	●	●	●	○	○	○
Groce-Katz	2010	[56]	UC, CRS	LWE	2-pary	PAKE	●	●	◐	◐	○	●	●	●	●	●	●	●	○	○	●	●	●	●	○	○	○
AugPAKE	2010	[100]	RO	SDH	2-pary	aPAKE	●	●	●	●	○	●	●	●	●	●	●	●	◐	○	○	●	●	●	●	●	○
Yang et al.	2010	[43]	–	CDH	2-pary	aPAKE	●	●	●	●	●	●	●	●	●	●	●	●	●	●	○	●	○	●	○	○	○
Yang et al.	2009	[42]	–	CDH	2-pary	aPAKE	●	●	●	●	●	●	●	●	●	●	●	●	●	●	○	●	○	●	○	○	○
Katz-Vaikuntanathan	2009	[67]	Std	LWE	2-pary	PAKE	●	●	○	◐	○	●	●	●	●	●	●	●	○	○	●	●	●	●	○	○	○
PACE v2	2008	[52]	RO, IC	GCBDH	2-pary	PAKE	●	●	●	●	○	●	●	●	●	●	●	●	○	○	○	●	●	○	○	○	○
SAE	2008	[97]	RO	CDH	2-pary	aPAKE	●	●	●	●	○	●	●	●	●	●	●	●	○	◐	○	●	●	●	○	○	○
J-PAKE	2008	[16]	–	CDH, DDH	2-pary	PAKE	●	●	●	●	○	●	●	●	●	●	●	●	○	○	○	●	●	●	○	○	○
SPAKE2+	2008	[93]	RO, CRS	CDH	2-pary	aPAKE	●	●	○	◐	○	●	●	●	●	●	●	●	○	◐	○	●	●	●	○	●	●
Wan et al.	2007	[34]	RO, IC	CDH	n-pary	PAKE	●	●	●	●	○	●	●	●	●	●	●	●	○	●	○	●	●	●	○	○	○
PACE v1	2006	[82]	RO, IC	gPACE-DH	2-pary	PAKE	●	●	●	●	○	●	●	●	●	●	●	●	○	○	○	●	●	○	●	○	○
Abdalla-Pointcheval	2006	[105]	CRS	DDH, QR	n-pary	PAKE	●	●	●	●	○	●	●	●	●	●	●	●	○	○	○	●	●	○	●	○	○
Abdalla et al.	2005	[24]	Std	DDH	3-pary	PAKE	●	●	●	●	○	●	●	●	●	●	●	●	○	○	○	●	●	●	○	○	●
SPAKE2	2005	[37]	RO, CRS	CDH	2-pary	PAKE	●	●	○	◐	○	●	●	●	●	●	●	●	○	○	○	●	●	●	●	●	●
SPAKE1	2005	[37]	RO, CRS	CDH	2-pary	PAKE	●	●	○	◐	○	●	●	●	●	●	●	●	○	○	○	●	●	●	●	●	●
Canetti et al.	2005	[104]	UC, CRS	DDH, QR	2-pary	PAKE	●	●	○	●	○	●	●	●	●	●	●	●	○	○	○	●	●	●	○	○	○
Jiang-Gong	2004	[92]	CRS	DDH	2-pary	PAKE	●	●	●	●	○	●	●	●	●	●	●	●	○	○	○	●	●	●	●	○	●
PEKEP	2004	[36]	RO	RSA	2-pary	PAKE	●	●	●	●	○	●	●	●	●	●	●	●	○	○	○	●	●	○	○	○	○
OEKE	2003	[76]	IC	CDH	2-pary	PAKE	●	●	◐	◐	○	●	●	●	●	●	●	●	○	○	○	●	●	●	●	○	○
Gennaro-Lindell	2003	[41]	CRS	DDH, QR	2-pary	PAKE	●	●	○	◐	○	●	●	●	●	●	●	●	○	○	○	●	●	●	○	○	○
Kobara-Imai	2002	[91]	Std	DDH	2-pary	PAKE	●	●	●	●	○	●	●	●	●	●	●	●	○	○	○	●	●	●	●	●	●
PAK-Z	2002	[66]	RO	CDH	2-pary	aPAKE	●	●	●	●	○	●	●	●	●	●	●	●	◐	◐	○	●	●	●	○	○	●
SRP-6	2002	[21]	–	–	2-pary	aPAKE	●	●	●	●	○	●	●	●	●	●	●	●	◐	◐	○	●	●	●	●	○	○
AMP	2001	[39]	–	–	2-pary	aPAKE	●	●	●	●	○	●	●	●	●	●	●	●	◐	◐	○	●	●	●	●	●	○
KOY	2001	[90]	CRS	DDH	2-pary	PAKE	●	●	◐	◐	○	●	●	●	●	●	●	●	○	○	○	●	●	●	○	○	○
PAK-RY	2001	[65]	RO	DDH	2-party	aPAKE	●	●	●	●	○	●	●	●	●	●	●	●	○	◐	○	●	●	●	○	○	●
PPK	2000	[35]	RO	DDH	2-party	PAKE	●	●	○	◐	○	●	●	●	●	●	●	●	○	○	○	●	●	●	○	○	●
PAK	2000	[35]	RO	DDH	2-party	PAKE	●	●	●	●	○	●	●	●	●	●	●	●	○	○	○	●	●	●	○	○	●
AuthA	2000	[71]	–	CDH	2-party	aPAKE	●	●	◐	◐	○	●	●	●	●	●	●	●	○	◐	○	●	●	●	●	●	●
SNAPI	2000	[73]	RO	RSA	2-party	PAKE	●	●	●	●	○	●	●	○	○	●	●	●	○	○	○	●	●	○	●	○	○
SRP	1998	[10]	–	–	2-party	aPAKE	●	●	●	●	○	●	●	●	○	●	●	●	◐	◐	○	●	●	●	○	○	○
OKE	1997	[72]	–	RSA	2-party	PAKE	●	●	●	●	○	●	●	○	○	●	●	●	○	○	○	●	●	○	●	○	○
B-SPEKE	1997	[86]	–	CDH	2-party	aPAKE	●	●	●	●	○	●	●	●	●	○	○	●	○	○	○	●	●	○	●	○	○
SPEKE	1996	[15]	–	CDH	2-party	PAKE	●	●	●	●	○	●	●	●	●	○	○	●	○	○	○	●	●	○	○	●	○
M-EKE	1995	[75]	–	CDH	2-party	PAKE	●	●	●	●	○	●	●	○	●	●	●	●	○	○	○	●	●	●	●	●	●
A-EKE	1993	[38]	–	CDH	2-party	aPAKE	●	●	●	●	○	●	●	○	●	●	●	●	○	○	○	●	●	●	●	○	○
EKE	1992	[13]	IC	CDH	2-party	PAKE	●	●	●	●	○	●	●	○	●	●	●	●	○	○	○	●	●	●	●	○	○

○: Unsupported
◐: Partial supported
●: Fully supported
IC: Ideal Cipher
RO: Random Oracle
Std: Standard Model
GG: Generic Group

UC: Universally Composable
QR: Quadratic Residuosity
AA: Algebraic-Adversary Model
CRS: Common Reference String
SDH: Strong Diffie-Hellman
DDH: Decision Diffie-Hellman
CDH: Computational Diffie-Hellman

OMDH: One-More Diffie-Hellman
OMGDH: One-More Gap Diffie-Hellman
DSDH: Decision Square Diffie-Hellman
GCBDH: Gap Chosen-Base Diffie-Hellman
SCDH: Strong Computational Diffie-Hellman
GSDH: Gap Strong Computational Diffie-Hellman
gPACE-DH: General Password-Based Chosen-Element DH

PAKE: Symmetric PAKE (in this table)
aPAKE: Asymmetric PAKE
sPAKE: Strong symmetric PAKE
saPAKE: Strong asymmetric PAKE
LWE: Learning With Errors
DLWE: Decision Learning With Errors
R-DLWE: Decision Ring-Learning With Errors

networks. S3 (explicit authentication) and S4 (perfect forward secrecy = PFS) are related. Namely, PFS cannot be achieved by any implicitly authenticated 2-message protocols based on public-key authentication [40]. As shown in Table 9, 24 implicitly authenticated protocols only satisfy weak forward secrecy.

It is encouraging to observe that S6-1 (resistance to replay attacks) and S6-2 (resistance to man-in-the-middle attacks) are achieved by 71 PAKE protocols. A method to prevent replay attacks is to use random numbers (implicitly adopted by all 71 PAKE protocols) or time stamps. The typical victim of man-in-the-middle attacks is the MQV protocol proposed by Menezes et al. [148], which was later improved by Krawczyk [40] into the famous HMQV protocol. An effective solution to resist man-in-the-middle attacks is to bind session identifiers to the session key, e.g., SPAKE2 [37] and P-SPEKE [20], thus ensuring that the session key is established with the intended entity. Another method is to bind messages to the identifier using zero-knowledge proof techniques, e.g., J-PAKE [16].

In addition, three RSA-based PAKE protocols (i.e., OKE [72], SNAPI [73], and SNAPI-X [73]) are vulnerable to partition attacks, which combine online guessing and offline guessing [84]. Three PAKE protocols (i.e., EKE [13], A-EKE [38], and M-EKE [75]) are susceptible to number-theoretic attacks, making offline password guessing feasible [23]. SRP [10] is vulnerable to undetectable online guessing attacks [21]. SPEKE [15] and B-SPEKE [86] are vulnerable to unknown key-share attacks and parallel-session attacks [20]. Finally, EKE [13] is vulnerable to a known-key attack. That is, attackers who know the session key in EKE [13] can learn the user's password by performing offline guessing attacks [75].

When it comes to the compromise of the long-term keys, the security property of resistance to **key compromise impersonation** (**KCI**) attacks has been ignored. But it is of practical importance and deserves attention. Considering the cloud-storage scenario, if the long-term private key of the server is compromised, it is unavoidable for the attacker to impersonate the server, but impersonating users on the server (to access cloud files) should be avoided. Table 9 reveals that none of the symmetric PAKE protocols achieve resistance to KCI attacks, owing to the fact that participants all possess plaintext passwords or password files with the same structure. But even in an asymmetric structure, 24.1%=8/29 of protocols fail to provide resistance to KCI attacks. For example, in the SPAKE2+ protocol [93], a malicious user who has the server's password file can cheat the server to pass authentication, and a server that has an honest user's password can fake the user's login request. This is because SPAKE2+ only requires the user to prove the knowledge of $\mathrm{H}(ID_U \| ID_S \| pw)$ instead of pw, while the value $\mathrm{H}(ID_U \| ID_S \| pw)$ also exists in the password file.

Moreover, a subtle connection exists between protocol asymmetry and S6-9 (resistance to pre-computation attacks). That is, (1) symmetric PAKE protocols that only share passwords (e.g., [15, 16]) cannot resist server compromise; (2) strong (a)symmetric PAKE protocols (e.g, [7, 58]) all fully satisfy S6-9; (3) asymmetric PAKE protocols achieve a higher proportion of server compromise resilience. The reason is that the asymmetric PAKE protocol was first proposed in [38] to address the potential insecurity of key servers. However, not all asymmetric PAKE protocols can successfully resist server compromise. Of the 30 asymmetric PAKE protocols, eight are unable to resist server compromise (i.e., using deterministic mapping values of passwords), 15 protocols partially resist server compromise (i.e., using mapping values of passwords and identifiers/public random salts), and only eight fully resist server compromise (i.e., using mapping values of passwords and secret random salts). In addition, only OMDHKE [63] tackles DoS attacks. A possible reason is that DoS attacks are more effectively mitigated through defense mechanisms from system security, e.g., throttling and rate limiting (see [149, 150]). With the rapid advancement of quantum computing, resistance to quantum attacks (S6-10) has become increasingly critical. PAKEs need to gradually migrate toward post-quantum designs to ensure their security in the post-quantum era. Since the launch

of the NIST **Post-Quantum Cryptography** (**PQC**) standardization process, numerous quantum-safe symmetric/asymmetric PAKEs [28, 146, 147] have been proposed. However, constructing a quantum-safe strong asymmetric PAKE remains an open problem.

Usability analysis. All 71 PAKE protocols satisfy U1 (password friendly). However, there are three anonymous protocols (i.e., Yang et al. [42], Yang et al. [43], and Zhang et al. [27]) that do not satisfy U2 (login device friendly). This is because their clients of these three protocols need to store password-based credentials. This forces the user to log in on the device where the credential is stored (rather than any device). Almost all PAKE protocols assume that the user only holds a password, which aligns with real-world usage. However, as an independent natural individual, the user needs software composed of code to execute the protocol algorithm. Therefore, the property of U2 is concerned with whether the client stores parameters related to the user. It is acceptable to store the necessary public parameters, such as group elements and common reference strings, but additional credentials (see [42, 43]) are redundant burdens and should be avoided.

In addition to RSA-based PAKE protocols [36, 72, 73], protocols that convert passwords to generators (e.g, SPEKE [15]) are also prone to not supporting pre-computation. This is because before the user enters the password pw, the client cannot compute the generator $g \leftarrow f(pw)$, and therefore cannot pre-compute g^x (see Table 3). Actually, there may be different benchmarks for achieving the remaining four usability-related criteria (i.e., U4~U7). The specific values of storage, communication, and transmission overhead are shown in Table 10. We just try to provide an intuitive comparison of 71 PAKE protocols in Table 9.

- A PAKE protocol is said to satisfy U4 (storage overhead) if the storage overhead is less than $3\mathbb{G}$. As with [14], we denote a group element by $\mathbb{G}$ and a κ-bit string by κ.
- A PAKE protocol is said to satisfy U5 (computation overhead) if the computation overhead per party is less than one multi-exponentiation. We ignore fixed-based exponentiations because they can be pre-computed and therefore do not increase the protocol execution time.
- A PAKE protocol is said to satisfy U6 (transmission overhead) if the communication overhead per party is less than $1\mathbb{G}$ for implicit authentication, or less than $1\mathbb{G} + 1\kappa$ for unilaterally/mutually explicit authentication.
- A PAKE protocol is said to satisfy U7 (round complexity) if it can achieve implicit authentication in 1-round/2-flow, unilaterally explicit authentication in 2-round/2-flow, or mutually explicit authentication in 3-round/3-flow.

In all, we can deduce that *no* scheme can fulfill all these 13 criteria – the only five PAKE protocols that can achieve 11 criteria (i.e., except for S5 user anonymity and S6 resistance to known attacks) are KC-SPAKE2+ [94], KC-SPAKE2 [94], PFS-SPAKE2 [20], Kobara-Imai [91], and M-EKE [75].

5.2 Comparison of Efficiency

A total of 14 PAKEs support asynchronous communication: eight protocols with 1-round/2-flow, five protocols with 2-round/4-flow, and two protocols with 3-round/6-flow (see Table 10). We strictly follow the original article and do not optimize the number of rounds and flows of PAKEs. For example, the key exchange phase of SPEKE [15] is 1-round/2-flow, and the key confirmation phase is 2-round/2-flow, resulting in a total round complexity of 3-round/4-flow. Although the key confirmation phase can be optimized to 1 round, we maintain the original description.

Table 10 reports only the number of exponentiation operations. Hashing, XOR, and encryption are omitted. We adopt the computation, communication, and storage overhead of Katz-Vaikuntanathan [153], Groce-Katz [56] in Jiang et al's work [28]. Fixed-base exponentiations (fb) can be pre-computed offline (e.g., $X = g^x, Y = g^y$ in SPAKE2 [37]). Variable-based exponentiations (vb) need to be computed online (e.g., X^y, Y^x in SPAKE2 [37]). Generally, PAKE protocols using

Table 10. Efficiency Comparison of 71 Representative PAKE Protocols

Protocol	Year	Ref.	R/F	EA	Computation overhead		Communication overhead		Storage overhead	Building blocks
					Client side	Server side	Client side	Server side		
KEM-PAKE	2025	[146]	3/3	C+S	1pke+1sig+2ae	3pke+2ae	1M + 1κ	1M + 1κ	1M + 1κ	KEM, AE
LPKE-PAKE	2024	[147]	3/3	C+S	2pke	2pke	1M + 1κ	2κ	0	LPKE, PKE
BarePAKE	2024	[79]	2/4	C	2fb+2mvb	2fb+2mvb	2G + 2κ	2G	2G + 1κ	Password-enc
SweetPAKE	2024	[80]	3/3	C	(2n+2)fb+2nmvb	4nfb+2nmvb	2nG + (n + 1)κ	2nG + (n + 1)κ	n	PKE, PRF, Honeyword
FuzzyPAKE	2023	[81]	2/2	C	2nfb	(2n+2)fb	n$\mathbb{G}$ + 1κ	n$\mathbb{G}$	0	ECC, RSS
OKAPE-HMQV	2022	[8]	2/2	C	1fb+1mvb	1fb+1mvb	1$\mathbb{G}$ + 1κ	1$\mathbb{G}$	1$\mathbb{G}$ + 1κ	Password-enc, HMQV
aEKE-HMQV	2022	[8]	1/2	None	1fb+1mvb	1fb+1mvb	1$\mathbb{G}$	1$\mathbb{G}$	1$\mathbb{G}$ + 1κ	Password-enc, HMQV
CRISP	2022	[70]	1/2	None	3vb	3vb	2$\mathbb{G}$	2$\mathbb{G}$	3$\mathbb{G}$	Pairing, PAKE, Htg
CHIP	2022	[70]	1/2	None	1fb+1vb+1mvb	1fb+1vb+1mvb	2$\mathbb{G}$	2$\mathbb{G}$	3$\mathbb{G}$	Pairing, PAKE
KHAPE	2021	[11]	3/3	C+S	1fb+1mvb	1fb+1mvb	1$\mathbb{G}$ + 1κ	1$\mathbb{G}$ + 2κ	2$\mathbb{G}$ + 1κ	Password-enc, HMQV
H-VPAKE	2021	[12]	2/2	None	2vb+4mvb	2vb+4mvb	6$\mathbb{G}$	6$\mathbb{G}$	1$\mathbb{G}$	SPHF, Hardware
Vasco et al.	2021	[26]	5/5	C	1vb	1vb	1$\mathbb{G}$ + 1κ	(2n + 1)$\mathbb{G}$	(3n + 3)$\mathbb{G}$	OPRF
KC-SPAKE2+	2020	[94]	3/3	C+S	2fb+2vb	2fb+2vb	1$\mathbb{G}$ + 1κ	1$\mathbb{G}$ + 1κ	0	–
KC-SPAKE2	2020	[94]	3/3	C+S	2fb+1vb	2fb+1vb	1$\mathbb{G}$ + 1κ	1$\mathbb{G}$ + 1κ	0	–
McQuoid et al.	2020	[99]	2/4	C+S	1fb+1vb	1fb+1vb	1$\mathbb{G}$ + 3κ	1$\mathbb{G}$ + 3κ	0	Hash-to-curve, OT
RLWE-PAKE	2020	[28]	3/3	C+S	$O(\frac{L'n}{\log n} + n\log^2 n)$zb	$O(L'n)$zb	$O(\frac{L'n}{\log n} + n\log n)\mathbb{M}$	$O(\frac{L'n}{\log n} + n\log n)\mathbb{M}$	0	ASPH
LWE-PAKE	2020	[28]	3/3	C+S	(nm)zb	$O(L'nm/\log q)$zb	$O(\frac{L'n}{\log n} + n\log n)\mathbb{M}$	$O(\frac{L'n}{\log n} + n\log n)\mathbb{M}$	0	ASPH
Erwig et al.	2020	[68]	3/3	None	(2n + 1)fb+(2n)vb	(2n + 1)fb+(4n)vb	(6n)$\mathbb{G}$	(4n + 1)$\mathbb{G}$	(2n + 2)$\mathbb{G}$	RSS, OT
AucPace	2019	[89]	4/6	C+S	1fb+2vb	1fb+2vb	1$\mathbb{G}$ + 1κ	2$\mathbb{G}$ + 3κ	1$\mathbb{G}$ + 2κ	PBKDF, Map2Point
saPAKE-BB	2019	[58]	2/2	S	10fb+1vb	2fb+2vb	2$\mathbb{G}$	2$\mathbb{G}$ + 2κ	1$\mathbb{G}$ + 1κ	CKEM, STOWF
Hwang et al.	2018	[9]	3/3	C	1fb+2vb	2fb+2vb	1$\mathbb{G}$ + 2κ	2$\mathbb{G}$	1$\mathbb{G}$ + 1κ	Password-enc
Dupont et al.	2018	[69]	5/5	None	n$\mathbb{fb}$+(2n + 2)vb	(n)fb+(2n + 2)vb	(4n + 2)$\mathbb{G}$	(4n + 2)$\mathbb{G}$	0	Garbled Circuits, OT
OPAQUE	2018	[7]	3/3	C+S	1fb+2vb+1mvb	1fb+1vb+1mvb	2$\mathbb{G}$ + 1κ	3$\mathbb{G}$ + 1κ	5$\mathbb{G}$	Password-enc, OPRF
PFS-SPAKE2	2018	[14]	3/3	C+S	2fb+1vb	2fb+1vb	1$\mathbb{G}$ + 1κ	1$\mathbb{G}$ + 1κ	0	–
P-SPEKE	2018	[20]	1/2	None	2vb	2vb	1$\mathbb{G}$	1$\mathbb{G}$	0	Hash-to-group, KDF
VTBPEKE	2017	[50]	3/3	C	2fb+2vb	1fb+3vb	1$\mathbb{G}$ + 2κ	1$\mathbb{G}$ + 2κ	1$\mathbb{G}$ + 1κ	Symmetric enc
TBPEKE	2017	[50]	2/2	None	1fb+2vb	1fb+2vb	1$\mathbb{G}$	1$\mathbb{G}$	0	–
SESPAKE	2017	[95]	6/6	C+S	1fb+2vb	1fb+1vb	1$\mathbb{G}$ + 1κ	1$\mathbb{G}$ + 2κ	3$\mathbb{G}$	MAC
Zhang-Yu	2017	[32]	2/2	None	$\omega(L'nm\log n)$zb	$\omega(L'nm\log n)$zb	$(L'n)\mathbb{M}$	$(L'n)\mathbb{M}$	0	SPHF, Peiker enc
Zhang et al.	2016	[27]	2/2	None	1fb+2vb+2mvb	1fb+2vb+1mvb	4$\mathbb{G}$	4$\mathbb{G}$	1$\mathbb{G}$ + 1κ	Password-enc, Sign
DE-PAKE	2016	[64]	5/6	C+S	1fb+4vb+1mvb	1fb+2vb+1mvb	3$\mathbb{G}$ + 1κ	7$\mathbb{G}$ + 1κ	6$\mathbb{G}$	OPRF, HMQV
Abdalla et al.	2015	[113]	3/6	C+S	2fb+11vb+1mvb	2fb+11vb+1mvb	3$\mathbb{G}$ + 3κ	3$\mathbb{G}$ + 3κ	κ	NIZK
Patched SPEKE	2014	[87]	2/4	C+S	2vb	2vb	1$\mathbb{G}$ + 1κ	1$\mathbb{G}$ + 1κ	0	Hash-to-group
Canetti et al.	2012	[51]	5/5	C+S	(n)fb+(n)vb	(n)fb+(n)vb	(n)$\mathbb{G}$ + 1κ	(3n)$\mathbb{G}$ + 1κ	0	SPHF, PVW enc
Groce-Katz	2010	[56]	3/3	C	(2nm)zb	$O(L'nm)$zb	$(1/2L'n)\mathbb{M}$	$(1/2L'n)\mathbb{M}$	0	SPHF, Shoup enc
AugPAKE	2010	[100]	4/4	C+S	1fb+1vb	1fb+1mvb	1$\mathbb{G}$ + 1κ	1$\mathbb{G}$ + 1κ	1$\mathbb{G}$	–
Yang et al.	2010	[43]	3/3	C+S	1fb+15vb	1fb+12vb	11$\mathbb{G}$	3$\mathbb{G}$	4$\mathbb{G}$	Password-enc, Sign
Yang et al.	2009	[42]	3/3	C+S	7fb+41vb+4mvb	2fb+53vb+2mvb	34$\mathbb{G}$	2$\mathbb{G}$	4$\mathbb{G}$	Password-enc, Sign
Katz-Vaikuntanathan	2009	[67]	3/3	None	$\omega(L'nm\log n)$zb	$\omega(L'nm\log n)$zb	$(L'n)\mathbb{M}$	$(L'n)\mathbb{M}$	0	SPHF, Peikert enc
PACE v2	2008	[52]	4/5	C+S	2vb	2vb	1$\mathbb{G}$ + 2κ	1$\mathbb{G}$ + 1κ	0	Password-enc, Htc
SAE	2008	[97]	2/4	C+S	2vb	2vb	2$\mathbb{G}$ + 1κ	2$\mathbb{G}$ + 1κ	0	Hash-to-curve
J-PAKE	2008	[16]	3/6	C+S	2fb+11vb+1mvb	2fb+11vb+1mvb	3$\mathbb{G}$ + 3κ	3$\mathbb{G}$ + 3κ	κ	NIZK
SPAKE2+	2008	[93]	1/2	None	3fb+2vb	4fb+1vb	1$\mathbb{G}$	1$\mathbb{G}$	2κ	–
Wan et al.	2007	[34]	3/3n	C+S	2fb+(log n)vb	1fb+(n)vb	(log n)$\mathbb{G}$ + 3κ	(n)$\mathbb{G}$ + 1κ	(n)κ	Password-enc, Tree
PACE v1	2006	[82]	4/5	C+S	2vb	2vb	1$\mathbb{G}$ + 2κ	1$\mathbb{G}$ + 1κ	0	Password-enc, Htc
Abdalla-Pointcheval	2006	[105]	4/4n	C+S	(n + 2)vb	(n + 2)vb	4$\mathbb{G}$ + 1κ	4$\mathbb{G}$ + 1κ	0	Hash-to-group, Sign
Abdalla et al.	2005	[24]	3/5	C+S	2fb+1vb	2fb+1vb	2$\mathbb{G}$ + 1κ	2$\mathbb{G}$ + 1κ	2κ	PAKE, MAC
SPAKE2	2005	[37]	1/2	None	3fb+1vb	3fb+1vb	1$\mathbb{G}$	1$\mathbb{G}$	0	–
SPAKE1	2005	[37]	1/2	None	3fb+1vb	3fb+1vb	1$\mathbb{G}$	1$\mathbb{G}$	0	–
PAK-Z+	2005	[96]	3/3	C+S	1fb+1vb	1fb+1vb	1$\mathbb{G}$ + 1κ	1$\mathbb{G}$ + 3κ	1$\mathbb{G}$ + 2κ	Hash-to-group, Sign
Canetti et al.	2005	[104]	4/4	None	3vb+6mvb	2vb+4mvb	9$\mathbb{G}$	7$\mathbb{G}$	0	SPHF, CS enc
Jiang-Gong	2004	[92]	3/3	C+S	3fb+1vb	3fb+2vb	2$\mathbb{G}$	1$\mathbb{G}$ + 1κ	0	Symmetric enc
PEKEP	2004	[36]	4/4	C+S	(m + 1)vb	(m + 1)vb	2$\mathbb{G}$ + 2κ	1$\mathbb{G}$ + 2κ	0	Password-enc
OEKE	2003	[76]	3/3	C	1fb+1vb	1fb+1vb	1$\mathbb{G}$ + 1κ	1$\mathbb{G}$	0	Password-enc
Gennaro-Lindell	2003	[41]	3/3	None	3vb+3mvb	3vb+3mvb	10$\mathbb{G}$	9$\mathbb{G}$	0	SPHF, CS enc
Kobara-Imai	2002	[91]	2/4	C+S	2fb+1vb	2fb+1vb	1$\mathbb{G}$ + 1κ	1$\mathbb{G}$ + 1κ	0	MAC
SRP-6	2002	[21]	6/6	C+S	2fb+1vb	1fb+1mvb	1$\mathbb{G}$ + 1κ	1$\mathbb{G}$ + 2κ	1$\mathbb{G}$ + 1κ	Mathematical func.
AMP	2001	[39]	4/4	C+S	1fb+1vb	2vb	1$\mathbb{G}$ + 1κ	1$\mathbb{G}$ + 1κ	1$\mathbb{G}$ + 1κ	–
KOY	2001	[90]	3/3	C	7fb+5vb+2mvb	7fb+5vb+2mvb	5$\mathbb{G}$ + 2κ	5$\mathbb{G}$	0	Sign
PAK-RY	2001	[65]	3/3	C+S	2fb+2vb	1fb+3vb	1$\mathbb{G}$ + 2κ	1$\mathbb{G}$ + 1κ	1$\mathbb{G}$	Hash-to-group, NIZK
PPK	2000	[35]	1/2	None	1fb+2vb	1fb+2vb	1$\mathbb{G}$	1$\mathbb{G}$	0	Hash-to-group
PAK	2000	[35]	3/3	C+S	1fb+2vb	1fb+2vb	1$\mathbb{G}$ + 1κ	1$\mathbb{G}$ + 1κ	0	Hash-to-group
AuthA	2000	[71]	2/2	S	1fb+1vb	1fb+1vb	1$\mathbb{G}$	1$\mathbb{G}$ + 1κ	1$\mathbb{G}$	Password-enc
SNAPI	2000	[73]	4/4	C+S	1vb	1vb	2$\mathbb{G}$ + 2κ	1$\mathbb{G}$ + 2κ	0	Password-enc
SRP	1998	[10]	6/6	C+S	2fb+1vb	1fb+1mvb	1$\mathbb{G}$ + 1κ	1$\mathbb{G}$ + 3κ	1$\mathbb{G}$ + 1κ	Mathematical func.
OKE	1997	[72]	4/4	C+S	1vb	1vb	2$\mathbb{G}$ + 2κ	1$\mathbb{G}$ + 2κ	0	Password-enc
B-SPEKE	1997	[86]	3/5	C+S	3vb	1fb+3vb	1$\mathbb{G}$ + 1κ	2$\mathbb{G}$ + 1κ	2$\mathbb{G}$	Hash-to-group
SPEKE	1996	[15]	3/4	C+S	2vb	2vb	1$\mathbb{G}$ + 1κ	1$\mathbb{G}$ + 1κ	0	Hash-to-group
M-EKE	1995	[75]	3/3	C+S	1fb+1vb	1fb+1vb	1$\mathbb{G}$ + 1κ	1$\mathbb{G}$ + 1κ	0	Password-enc
A-EKE	1993	[38]	5/5	C+S	1fb+1vb	1fb+1vb	1$\mathbb{G}$ + 2κ	1$\mathbb{G}$ + 2κ	0	Password-enc
EKE	1992	[13]	5/5	C+S	1fb+1vb	1fb+1vb	1$\mathbb{G}$ + 2κ	1$\mathbb{G}$ + 1κ	0	Password-enc

κ: κ-bit string
$\mathbb{G}$: Group element
$\mathbb{M}$: Matrix over lattice
ae: Authenticated encryption
zb: Multiplications in $\mathbb{Z}_q$
fb: Fixed-based exponentiation
vb: Variable-based exponentiation
mvb: Multi-exponentiation
sig: Signature
R/F: Round/Flow
pke: Public-key encryption
EA: Explicit authentication
OT: Oblivious transfer
Htg: Hash-to-group
Htc: Hash-to-curve
KEM: Key encapsulation mechanism
KDF: Key derivation function
SPHF: Smooth projective hash function
NIZK: Non-interactive zero-knowledge
OPRF: Oblivious pseudo-random function
ASPH: Approximate smooth projective hashing
PBKDF: Password-based key derivation function
CKEM: Conditional key encapsulation mechanism
LPKE: Lossy public key encryption
RSS: Robust secret sharing
MAC: Message authentication code
STOWF: Salted tight one-way function
Shoup enc: Shoup encryption [111]
Peiker enc: Peiker encryption [151]
PVW enc: Peikert et al. encryption [152]
CS enc: Cramer-Shoup encryption [107]
ECC: Error correcting codes

HMQV (e.g., OPAQUE [7] and aEKE-HMQV [8]) typically require multi-exponentiation (mvb). The computation cost of 1vb (or 1mvb) is enough to complete authentication and key exchange (see the OKAPE-HMQV protocol [8]). But anonymous PAKE protocols (e.g., [27, 42, 43]) and multi-party PAKE protocols (e.g., [34, 105]) often require more computation, communication, and storage overhead. They strengthen the basic guarantees of security properties at the expense of efficiency. In addition, in 1999, Halevi and Krawczyk [144] pointed out that it is unavoidable to use public-key technology to resist offline guessing attacks. This is also reflected in Table 10: All PAKE protocols use public-key techniques, such as Diffie-Hellman key exchange and RSA.

Among all building blocks, the most widely used cryptography tool (42.3%=30/71) is encryption, which includes password-based symmetric encryption, public-key encryption (e.g., Cramer-Shoup encryption [107] and Peiker encryption [151]). The second most used (15.5%=11/71) is hash-to-group/hash-to-curve. Other tools include Signature, NIZK, OPRF, and so on. In particular, OPRF has have attracted extensive attention as an efficient cryptography tool. For example, the OPRF (i.e., 2HashDH [154]) used in OPAQUE [7] only requires one variable-based exponentiation per party. The comparison between OPAQUE [7] and its variant KHAPE [11] highlights the trade-offs of using OPRF in a PAKE protocol. With OPRF, OPAQUE [7] is resilient to pre-computation attacks. By dispensing with the OPRF, KHAPE [11] forfeits this protection but gains resist pre-computation attacks, but it provides resilience to OPRF compromise, and improved computational performance (see Figure 3).

We emphasize that inefficient PAKEs (e.g., [7, 70]) are not necessarily useless. Performance constraints and security goals for a PAKE vary depending on the expected application environment. For highly critical environments such as government offices, security is the first requirement. To attain the desired level of protection, it may be acceptable to use a slightly inefficient authentication protocol, such as OPAQUE [7]. Conversely, lower-risk environments may focus more on efficiency. A very useful but less secure authentication scheme may also be acceptable to service providers. With this in mind, it is wise and feasible to choose a specific PAKE for target environments and applications. After all, from a usability and security point of view, it is unlikely that any single scheme will suit all environments and target users.

6 Discussion

Our review of PAKEs reveals a rich set of useful ideas, from which we distill several insights.

The neglected part of PAKE. PAKEs ignore the password update phase for a long time, which is a disadvantage for its application. In other authentication schemes, such as the smart-card-based password authentication mechanism (e.g., [45]), the operations performed on the password during the password update phase will be described in detail, which also matches the practical situation. Most PAKE protocols, however, pay little attention to this issue. Remarkably, in a PAKE, the password cannot be updated locally. Achieving authentication inherently requires some common information (e.g., passwords) to be shared in advance. When the user updates a new password locally and the server still possesses the old password, the prerequisite for authentication is broken. In a word, clarifying how the password update phase works is not only closer to the real world, but also allows researchers to have a broader understanding of the design of PAKEs.

For clarity of presentation, we propose a general framework for password updates in PAKE protocols (as shown in Figure 4) for password update in PAKE protocols, intended to provide a common reference point for systematic analysis and comparison. In the symmetric PAKEs [13, 147], aPAKEs [8, 146], and saPAKEs [7, 58], the most straightforward approach is to authenticate with a standard login procedure followed by a full re-registration. Consequently, the round complexity and computational cost of the update phase are bounded by the sum of these two steps. While

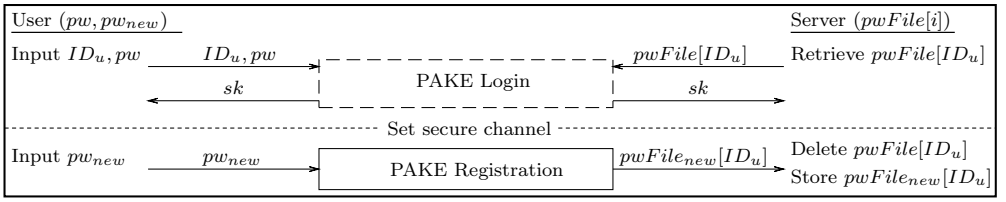


Fig. 4. A generic framework for the password update phase of PAKE. ID_u denotes the identity of user U. pw and pw_{new} denote the original and new passwords, respectively. $pwFile[i]$ denotes the password file stored on the server S, and sk is the session key. The dashed box indicates operations over the public channel, whereas the solid box indicates operations over the secure channel. U can establish a secure channel with S via PAKE using pw over the public channel, and then generate a new password file $pwFile_{new}[ID_u]$ for S.

dedicated constructions can reduce this overhead, the baseline strategy remains broadly applicable and reliably achieves secure password rotation.

Further work on the online password update without revealing the user's plain-text password to the server is needed, considering the disturbing fact that the user's plain-text password may be unconsciously logged, unauthorizedly accessed, and even misused by insider attackers. A notable real-world incident illustrating such risks occurred at Twitter [155]. Developing such privacy-preserving password update mechanisms remains a crucial step toward enhancing the overall resilience of PAKE systems in practice.

The dilemma of deploying PAKE. The deployment of PAKE has two basic requirements: (1) A secure channel is required during the registration phase, and (2) A user must be modified to support and execute a PAKE [22]. The fundamental assumption of PAKE is that both parties pre-share a password. It means that users either share a password face-to-face or send the password under the protection of a secure channel such as TLS. The deployment of PAKE on the user end usually requires a browser plugin or a software application [14, 64]. Currently, the widely accepted password authentication scheme (i.e., password-over-TLS) does not require any modifications to the client. One possible approach is to dynamically (and securely) load the PAKE execution code to the client when the user requests authentication. Then, the user enters the password and executes the PAKE with the server. But ensuring secure dynamic code loading is challenging [156].

Future trends of PAKE. Based on the above analysis of existing PAKEs and considering the development trends of computing and communication techniques, we identify four potential directions for the future design of PAKEs.

— *Anonymous PAKE protocol*. With more and more sensitive services and applications (that involve personal information) involving personal information protecting user privacy has become a major concern. For instance, the 2025 Verizon **Data Breach Investigations Report** (**DBIR**) [168] reveals that approximately 61% of data breaches involve **personally identifiable information** (**PII**) such as email, name, user ID, and mobile phone number. If these PII data are obtained by malicious network service providers rather than attackers, it is equally disturbing (e.g., activity tracking and targeted advertising). Therefore, it is crucial to adopt mechanisms that protect users' identities and associated activities from unnecessary exposure. However, traditional password-over-TLS authentication does not provide anonymity. Because the server authenticates the user's (user ID, password)-pair. In contrast, the anonymous PAKE can simultaneously achieve authentication and anonymity protection simultaneously, ensuring that a user's identity remains concealed not only hidden from the external adversary but also from the server itself.

— *Blockchain-based PAKE protocol*. Due to promising properties (e.g., decentralization and non-repudiation), researchers have attempted to apply blockchain to various protocols, e.g., blockchain-based self-tallying voting protocols [158, 159] and privacy-preserving payment protocols [160, 161]. Beyond its role in distributed authentication, blockchain can also be used to design anonymous authentication protocols. The research [162] proposes a distributed authentication protocol for the Internet of Things based on a public blockchain. Fully executing PAKEs on-chain may incur significant latency and privacy concerns. Therefore, the focus of blockchain-based PAKE designs is to leverage blockchain as a verifiable and tamper-resistant auxiliary component in password authentication systems. For example, blockchain can be used to enhance auditability [163] or record cryptographic commitments in a verifiable manner [164]. Still, how to incorporate blockchain as an auxiliary component into password authentication protocols remains an interesting and promising line of research.
— *Lattice-based PAKE protocol*. This class of protocols has attracted increasing research attention in recent years. With the growing threat posed by quantum computers to classical hardness assumptions (e.g., discrete logarithm and integer factorization problems), PAKE protocols based on classical hardness assumptions are facing increasing security risks. Quantum adversaries can efficiently break PAKEs (e.g., OPAQUE [7] and KHAPE [11]) using the Shor algorithm [61], and can apply the Grover algorithm [62] to accelerate brute-force search and hash collision attacks. As a result, quantum-resistant PAKE protocols have emerged as an important subclass of PAKE. Research on quantum-resistant PAKE protocols has become a hot topic [165]. Among all post-quantum algorithms (e.g., hash-based, code-based, lattice-based, and isogeny-based), lattice-based schemes appear to be the most promising general-purpose algorithms due to their provable security and relatively low computation cost [166]. Notably, three of the five post-quantum algorithms standardized by NIST are lattice-based [167]. Designing secure, efficient, and practical lattice-based PAKE protocols therefore remains a key direction for future research.
— *Identity-based PAKE protocol*. As mentioned above, (symmetric) PAKE may be more suitable in symmetric settings. However, when both parties share a plaintext password, compromising either party immediately exposes the shared secret. At CRYPTO'22, Cremers et al. [70] proposed the CRISP protocol combining symmetric PAKE with identity-based authentication, namely symmetric identity-based PAKE. CRISP is resistant to pre-computation attacks, which is the highest level of security that asymmetric PAKE (including aPAKE like OKAPE-HMQV [8] and saPAKE like OPAQUE [7]) can provide. Compared to non-identity based PAKE (particularly strong asymmetric PAKE, i.e., saPAKE), one critical defect of identity-based PAKE is that both parties need to store a password file. Thus, users can only log in on the device with this password file. Moreover, efficiency optimization for identity-based PAKEs remains an open research challenge. In all, identity-based PAKE protocols are likely to be a promising direction.

7 Conclusion

We have presented a comprehensive review of the most significant studies to date on PAKE protocols. We propose a list of 13 desirable properties, allowing PAKE protocols to be rated across a common spectrum for a fair evaluation. We categorize existing PAKE protocols into seven types according to their underlying design strategies, and investigate the pros and cons of each type. We further classify them from five other key perspectives (i.e., symmetry, number of participants, hardness assumptions, security goals, and round complexity), and performed a large-scale comparative analysis encompassing 71 PAKE protocols. We have also pointed out four promising future research

directions. We believe this work provides a thorough and in-depth understanding of how to design PAKE protocols that offer acceptable tradeoffs between security and usability.

References

[1] Lawrence O'Gorman. 2003. Comparing passwords, tokens, and biometrics for user authentication. *IEEE* 91, 12 (2003), 2021–2040.

[2] Joseph Bonneau, Cormac Herley, Paul C. van Oorschot, and Frank Stajano. The quest to replace passwords: A framework for comparative evaluation of web authentication schemes. In *Proceedings of the IEEE S&P 2012*. 553–567.

[3] Joseph Bonneau, Cormac Herley, Paul C. van Oorschot, and Frank Stajano. 2015. Passwords and the evolution of imperfect authentication. *Communications of the ACM* 58, 7 (2015), 78–87.

[4] V. Zimmermann. 2021. *From the Quest to Replace Passwords towards Support- ing Secure and Usable Password Creation*. Ph.D. Dissertation. Technical University of Darmstadt, Germany.

[5] Lily Hay Newman. 2021. *Why the Password Isn't Dead Quite Yet*. Retrieved 9 July 2023 from https://bit.ly/46xY886

[6] 2022. *The Password Isn't Dead Yet. You Need a Hardware Key*. Retrieved 9 July 2023 from https://bit.ly/43bXo5K

[7] Stanislaw Jarecki, Hugo Krawczyk, and Jiayu Xu. OPAQUE: An asymmetric pake protocol secure against pre-computation attacks. In *Proceedings of the EUROCRYPT 2018*. 456–486.

[8] Bruno Freitas Dos Santos, Yanqi Gu, Stanislaw Jarecki, and Hugo Krawczyk. Asymmetric PAKE with low computation and communication. In *Proceedings of the EUROCRYPT 2022*. 127–156.

[9] Jung Yeon Hwang, Stanislaw Jarecki, Taekyoung Kwon, Joohee Lee, Ji Sun Shin, and Jiayu Xu. Round-reduced modular construction of asymmetric password-authenticated key exchange. In *Proceedings of the SCN 2018*. 485–504.

[10] Thomas D. Wu. The secure remote password protocol. In *Proceedings of the NDSS 1998*. 97–111.

[11] Yanqi Gu, Stanislaw Jarecki, and Hugo Krawczyk. KHAPE: Asymmetric PAKE from key-hiding key exchange. In *Proceedings of the CRYPTO 2021*.

[12] Ji Sun Shin, Minjae Jo, Jung Yeon Hwang, and Jaehwan Lee. 2021. A verifier-based password-authenticated key exchange using tamper-proof hardware. *Computer Journal* 64, 8 (2021), 1293–1302.

[13] Steven M. Bellovin and Michael Merritt. Encrypted key exchange: Password-based protocols secure against dictionary attacks. In *Proceedings of the IEEE S&P 1992*. 72–84.

[14] José Becerra, Dimiter Ostrev, and Marjan Skrobot. Forward secrecy of SPAKE2. In *Proceedings of the ProvSec 2018*. 366–384.

[15] David P. Jablon. 1996. Strong password-only authenticated key exchange. *Computer Communication Review* 26, 5 (1996), 5–26.

[16] Feng Hao and Peter Y. A. Ryan. Password Authenticated Key Exchange by Juggling. In *Proceedings of the SPW 2008*. 159–171.

[17] Rick Fillion. 2018. *Secure Remote Password (SRP): How 1Password uses it*. Retrieved 5 July 2023 from https://bit.ly/3roH1FW

[18] Apple. 2021. *HomeKitADK*. Retrieved 5 July 2023 from https://github.com/apple/HomeKitADK

[19] 2023. *Firefox Sync*. Retrieved 5 July 2023 from https://www.mozilla.org/en-US/firefox/sync/

[20] Feng Hao, Roberto Metere, Siamak F. Shahandashti, and Changyu Dong. 2018. Analysing and patching SPEKE in ISO/IEC. *IEEE Transactions on Information Forensics and Security* 13, 11 (2018), 2844–2855.

[21] Thomas D. Wu. 2002. SRP-6: Improvements and refinements to the secure remote password protocol. Submission to IEEE P1363.2.

[22] Feng Hao and Paul C. van Oorschot. SoK: Password-authenticated key exchange - theory, practice, standardization and real-world lessons. In *Proceedings of the AsiaCCS 2022*. 697–711.

[23] Sarvar Patel. Number theoretic attacks on secure password schemes. In *Proceedings of the IEEE S&P 1997*. 236–247.

[24] Michel Abdalla, Pierre-Alain Fouque, and David Pointcheval. Password-based authenticated key exchange in the three-party setting. In *Proceedings of the PKC 2005*. 65–84.

[25] Jonathan Katz, Philip D. MacKenzie, Gelareh Taban, and Virgil D. Gligor. Two-server password-only authenticated key exchange. In *Proceedings of the ACNS 2005*. 1–16.

[26] Maria Isabel Gonzalez Vasco, Angel L. Pérez del Pozo, and Claudio Soriente. 2021. A key for john doe: Modeling and designing anonymous password-authenticated key exchange protocols. *IEEE Trans. Dependable Secur. Comput.* 18, 3 (2021), 1336–1353.

[27] Zhenfeng Zhang, Kang Yang, Xuexian Hu, and Yuchen Wang. Practical Anonymous Password Authentication and TLS with Anonymous Client Authentication. In *Proceedings of the ACM CCS 2016*. 1179–1191.

[28] Shaoquan Jiang, Guang Gong, Jingnan He, Khoa Nguyen, and Huaxiong Wang. PAKEs: New framework, new techniques and more efficient lattice-based constructions in the standard model. In *Proceedings of the PKC 2020*. 396–427.

[29] Peixin Ren and Xiaozhuo Gu. Practical post-quantum password-authenticated key exchange based-on module-lattice. In *Proceedings of the ICISC 2021*. 137–156.
[30] Jingwei Jiang, Ding Wang, and Guoyin Zhang. 2023. QPause: Quantum-resistant password-protected data outsourcing for cloud storage. *IEEE Transactions on Services Computing* 17, 3 (2023), 1140–1153.
[31] Jonathan Katz and Vinod Vaikuntanathan. Round-optimal password-based authenticated key exchange. In *Proceedings of the TCC 2011*.
[32] Jiang Zhang and Yu Yu. Two-round PAKE from approximate SPH and instantiations from lattices. In *Proceedings of the ASIACRYPT 2017*.
[33] Zengpeng Li, Ding Wang, and Eduardo Morais. 2022. Quantum-safe round-optimal password authentication for mobile devices. *IEEE Transactions on Dependable and Secure Computing* 19, 3 (2022), 1885–1899.
[34] Zhiguo Wan, Robert H. Deng, Feng Bao, and Bart Preneel. *n* PAKE $^{+}$: A hierarchical group password-authenticated key exchange protocol using different passwords. In *Proceedings of the ICICS 2007*. 31–43.
[35] Victor Boyko, Philip D. MacKenzie, and Sarvar Patel. Provably secure password-authenticated key exchange using diffie-hellman. In *Proceedings of the EUROCRYPT 2000*. 156–171.
[36] Muxiang Zhang. New approaches to password authenticated key exchange based on RSA. In *Proceedings of the ASIACRYPT 2004*.
[37] Michel Abdalla and David Pointcheval. Simple password-based encrypted key exchange protocols. In *Proceedings of the CT-RSA 2005*.
[38] Steven M. Bellovin and Michael Merritt. Augmented encrypted key exchange: A password-based protocol secure against dictionary attacks and password file compromise. In *Proceedings of the ACM CCS 1993*. 244–250.
[39] Taekyoung Kwon. Authentication and key agreement via memorable passwords. In *Proceedings of the NDSS 2001*. 1–21.
[40] Hugo Krawczyk. HMQV: A high-performance secure diffie-hellman protocol. In *Proceedings of the CRYPTO 2005*. 546–566.
[41] Rosario Gennaro and Yehuda Lindell. A framework for password-based authenticated key exchange. In *Proceedings of the EUROCRYPT 2003*.
[42] Yanjiang Yang, Jianying Zhou, Jian Weng, and Feng Bao. A New Approach for Anonymous Password Authentication. In *Proceedings of the ACSAC 2009*.
[43] Yanjiang Yang, Jianying Zhou, Jun Wen Wong, and Feng Bao. Towards practical anonymous password authentication. In *Proceedings of the ACSAC 2010*.
[44] Mihir Bellare, David Pointcheval, and Phillip Rogaway. Authenticated key exchange secure against dictionary attacks. In *Proceedings of the EUROCRYPT 2000*. 139–155.
[45] D. Wang and P. Wang. 2018. Two birds with one stone: Two-factor authentication with security beyond conventional bound. *IEEE Transactions on Dependable and Secure Computing* 15, 4 (2018), 708–722.
[46] D. Wang and P. Wang. On the implications of zipf's law in passwords. In *Proceedings of the ESORICS 2016*. 111–131.
[47] D. Wang, D. B. He, P. Wang, and C. H. Chu. 2015. Anonymous two-factor authentication in distributed systems: Certain goals are beyond attainment. *IEEE Transactions on Dependable and Secure Computing* 12, 4 (2015), 428–442.
[48] D. Wang, Z.J. Zhang, and P. Wang. Targeted online password guessing: An underestimated threat. In *Proceedings of the ACM CCS 16*.
[49] Ding Wang, Haibo Cheng, Ping Wang, Xinyi Huang, and Gaopeng Jian. 2017. Zipf's law in passwords. *IEEE Transactions on Information Forensics and Security* 12, 11 (2017), 2776–2791.
[50] David Pointcheval and Guilin Wang. VTBPEKE: Verifier-based two-basis password exponential key exchange. In *Proceedings of the AsiaCCS 2017*. 301–312.
[51] Ran Canetti, Dana Dachman-Soled, Vinod Vaikuntanathan, and Hoeteck Wee. Efficient password authenticated key exchange via oblivious transfer. In *Proceedings of the PKC 2012*. 449–466.
[52] Jean-Sébastien Coron, Aline Gouget, Thomas Icart, and Pascal Paillier. 2012. Supplemental access control (PACE v2): security analysis of PACE integrated mapping. *Cryptography and Security* (2012), 207–232.
[53] Joseph Bonneau. The science of guessing: Analyzing an anonymized corpus of 70 million passwords. In *Proceedings of the IEEE S&P 2012*. 538–552.
[54] Z. Chen, S. Pu, and S. Zhu. 2012. Traceback technology for anonymous network. *Journal of Computer Research and Development* 49, S2 (2012), 111–117.
[55] Peter Mayer, Nina Gerber, Benjamin Reinheimer, Philipp Rack, Kristoffer Braun, and Melanie Volkamer. I (don't) see what you typed there! shoulder-surfing resistant password entry on gamepads. In *Proceedings of the CHI 2019*. 1–12.
[56] Adam Groce and Jonathan Katz. A new framework for efficient password-based authenticated key exchange. In *Proceedings of the ACM CCS 2010*. 516–525.
[57] Jonathan Katz, Rafail Ostrovsky, and Moti Yung. Forward secrecy in password-only key exchange protocols. In *Proceedings of the SCN 2002*.

[58] Tatiana Bradley, Stanislaw Jarecki, and Jiayu Xu. Strong asymmetric PAKE based on trapdoor CKEM. In *Proceedings of the CRYPTO 2019*. 798–825.
[59] T. Y. Youn, Y. H. Park, C. H. Kim, and J. Lim. 2008. Weakness in a RSA-based password authenticated key exchange protocol. *Information Processing Letters* 108, 6 (2008), 339–342.
[60] John Preskill. 2018. Quantum computing in the NISQ era and beyond. *Quantum* 2 (2018), 79.
[61] Peter W Shor. Algorithms for quantum computation: discrete logarithms and factoring. In *Proceedings of the FOCS 1994*. 124–134.
[62] Lov K Grover. A fast quantum mechanical algorithm for database search. In *Proceedings of the STOC 1996*. 212–219.
[63] Emmanuel Bresson, Olivier Chevassut, and David Pointcheval. New security results on encrypted key exchange. In *Proceedings of the PKC 2004*.
[64] Stanislaw Jarecki, Hugo Krawczyk, Maliheh Shirvanian, and Nitesh Saxena. Device-enhanced password protocols with optimal online-offline protection. In *Proceedings of the AsiaCCS 2016*. 177–188.
[65] Philip D. MacKenzie. More efficient password-authenticated key exchange. In *Proceedings of the CT-RSA 2001*. 361–377.
[66] Philip MacKenzie. 2002. The PAK suite: Protocols for password-authenticated key exchange. Contribution to IEEE P1363.
[67] Jonathan Katz and Vinod Vaikuntanathan. Smooth projective hashing and password-based authenticated key exchange from lattices. In *Proceedings of the ASIACRYPT 2009*. 636–652.
[68] Andreas Erwig, Julia Hesse, Maximilian Orlt, and Siavash Riahi. Fuzzy asymmetric password-authenticated key exchange. In *Proceedings of the ASIACRYPT 2020*. 761–784.
[69] Pierre-Alain Dupont, Julia Hesse, David Pointcheval, Leonid Reyzin, and Sophia Yakoubov. Fuzzy password-authenticated key exchange. In *Proceedings of the EUROCRYPT 2018*. 393–424.
[70] Cas Cremers, Moni Naor, Shahar Paz, and Eyal Ronen. CHIP and CRISP: Protecting all parties against compromise through identity-binding PAKEs. In *Proceedings of the CRYPTO 2022*. 668–698.
[71] M. Bellare and P. Rogaway. 2000. The AuthA protocol for password-based authenticated key exchange. Contribution to IEEE P1363.
[72] Stefan Lucks. Open key exchange: How to defeat dictionary attacks without encrypting public keys. In *Proceedings of the SPW 1997*.
[73] Philip D. MacKenzie, Sarvar Patel, and Ram Swaminathan. Password-Authenticated key exchange based on RSA. In *Proceedings of the ASIACRYPT 2000*. 599–613.
[74] 2009. IEEE Standard specification for password-based public-key cryptographic techniques. Retrieved 29 October 2025 from https://standards.ieee.org/standard/1363_2-2008.html
[75] Michael Steiner, Gene Tsudik, and Michael Waidner. 1995. Refinement and extension of encrypted key exchange. *ACM SIGOPS Operating Systems Review* 29, 3 (1995), 22–30.
[76] Emmanuel Bresson, Olivier Chevassut, and David Pointcheval. Security proofs for an efficient password-based key exchange. In *Proceedings of the ACM CCS 2003*. 241–250.
[77] German Federal Offer for Information Security. 2012. Advanced security mechanisms for machine readable travel documents part 1 – eMRTDs with BAC/PACEv2 and EACv1. Retrieved 17 June 2023 from https://bit.ly/46BSUbx
[78] Craig Gentry, Philip D. MacKenzie, and Zulfikar Ramzan. A method for making password-based key exchange resilient to server compromise. In *Proceedings of the CRYPTO 2000*. 142–159.
[79] Manuel Barbosa, Kai Gellert, Julia Hesse, and Stanislaw Jarecki. Bare pake: Universally composable key exchange from just passwords. In *Proceedings of the CRYPTO 2024*. 183–217.
[80] Afonso Arriaga, Peter YA Ryan, and Marjan Skrobot. SweetPAKE: Key exchange with decoy passwords. In *Proceedings of the AsiaCCS 2024*. 1017–1033.
[81] Jonathan Bootle, Sebastian Faller, Julia Hesse, Kristina Hostáková, and Johannes Ottenhues. Generalized fuzzy password-authenticated key exchange from error correcting codes. In *Proceedings of the ASIACRYPT 2023*. 110–142.
[82] Jens Bender, Marc Fischlin, and Dennis Kügler. Security analysis of the PACE key-agreement protocol. In *Proceedings of the ISC 2009*. 33–48.
[83] F. Wei, C. Ma, and Q. Cheng. 2011. Gateway-oriented password-authenticated key exchange based on RSA. *Chinese Journal of Computers*. 34, 1 (2011), 38–46.
[84] D. Wang, P. Wang, and M. Li. 2015. Cryptanalysis and improvement of gateway-oriented password-authenticated key exchange based on RSA. *Acta Electronica Sinica*. 43, 1 (2015), 176–184.
[85] Whitfield Diffie and Martin E. Hellman. 1976. New directions in cryptography. *IEEE Transactions on Information Theory* 22, 6 (1976), 644–654.
[86] David P. Jablon. Extended password key exchange protocols immune to dictionary attacks. In *Proceedings of the WETICE 1997*.
[87] Feng Hao and Siamak Fayyaz Shahandashti. The SPEKE protocol revisited. In *Proceedings of the SSR 2014*. 26–38.

[88] 2017. *Information Technology – Security Techniques – Key Management – Part 4: Mechanisms based on Weak Secrets.* Retrieved from https://www.iso.org/standard/67933.html
[89] Björn Haase and Benoît Labrique. 2019. AuCPace: Efficient verifier-based PAKE protocol tailored for the IIoT. *IACR Transactions on Cryptographic Hardware and Embedded Systems* 2019, 2 (2019), 1–48.
[90] Jonathan Katz, Rafail Ostrovsky, and Moti Yung. Efficient password-authenticated key exchange using human-memorable passwords. In *Proceedings of the EUROCRYPT 2001.* 475–494.
[91] Kazukuni Kobara and Hideki Imai. 2002. Pretty-simple password-authenticated key-exchange protocol proven to be secure in the standard model. *IEICE Transactions on Fundamentals of Electronics, Communications and Computer Sciences* 85-A, 10 (2002), 2229–2237.
[92] Shaoquan Jiang and Guang Gong. Password based key exchange with mutual authentication. In *Proceedings of the SAC 2004.* 267–279.
[93] David Cash, Eike Kiltz, and Victor Shoup. The twin diffie-hellman problem and applications. In *Proceedings of the EUROCRYPT 2008.* 127–145.
[94] Victor Shoup. Security analysis of itSPAKE2+. In *Proceedings of the TCC 2020.* 31–60.
[95] Stanislav Smyshlyaev, Evgeny K. Alekseev, Igor B. Oshkin, and Vladimir Popov. 2017. The security evaluated standardized password-authenticated key exchange (SESPAKE) protocol. *RFC* 8133 (2017), 1–51.
[96] Craig Gentry, Philip MacKenzie, and Zulfikar Ramzan. 2005. PAK-Z+. Submission to IEEE P1363.2.
[97] Dan Harkins. Simultaneous authentication of equals: A secure, password-based key exchange for mesh networks. In *Proceedings of the SensorComm 2008.* 839–844.
[98] M. Qi, W. Hu, and Y. Tai. 2024. SAE+: One-round provably secure asymmetric sae protocol for client-server model. *IEEE Transactions on Information Forensics and Security* 19 (2024), 3906 – 3913.
[99] Ian McQuoid, Mike Rosulek, and Lawrence Roy. Minimal symmetric PAKE and 1-out-of-N OT from programmable-once public functions. In *Proceedings of the ACM CCS 2020.* 425–442.
[100] SeongHan Shin and Kazukuni Kobara. 2012. Efficient augmented password-only authentication and key exchange for IKEv2. *RFC* 6628.
[101] Feng Hao. 2017. J-PAKE: Password-authenticated key exchange by juggling. *RFC* 8236 (2017), 1–15.
[102] 2023. *OpenSSL Project.* Retrieved 30 July 2023 from https://www.openssl.org
[103] 2023. *Pale Moon Sync.* Retrieved 30 July 2023 from https://www.palemoon.org/sync/
[104] Ran Canetti, Shai Halevi, Jonathan Katz, Yehuda Lindell, and Philip D. MacKenzie. Universally composable password-based key exchange. In *Proceedings of the EUROCRYPT 2005.* 404–421.
[105] Michel Abdalla and David Pointcheval. A scalable password-based group key exchange protocol in the standard model. In *Proceedings of the ASIACRYPT 2006.* 332–347.
[106] Daniele Micciancio and Chris Peikert. Trapdoors for lattices: Simpler, tighter, faster, smaller. In *Proceedings of the EUROCRYPT 2012.*
[107] Ronald Cramer and Victor Shoup. Universal hash proofs and a paradigm for adaptive chosen ciphertext secure public-key encryption. In *Proceedings of the EUROCRYPT 2002.* 45–64.
[108] Chris Peikert. Lattice cryptography for the internet. In *Proceedings of the PQCrypto 2014.* 197–219.
[109] Moni Naor and Moti Yung. Universal one-way hash functions and their cryptographic applications. In *Proceedings of the ACM STOC 1989.*
[110] Kaoru Kurosawa and Yvo Desmedt. A new paradigm of hybrid encryption scheme. In *Proceedings of the CRYPTO 2004.* 426–442.
[111] Victor Shoup. 2001. A proposal for an ISO standard for public key encryption. *IACR Cryptol. ePrint Arch.*, 112.
[112] Julia Hesse. Separating symmetric and asymmetric password-authenticated key exchange. In *Proceedings of the SCN 2020.*
[113] Michel Abdalla, Fabrice Benhamouda, and Philip MacKenzie. Security of the J-PAKE password-authenticated key exchange protocol. In *Proceedings of the IEEE S&P 2015.* 571–587.
[114] Rongxing Lu and Zhenfu Cao. 2007. Simple three-party key exchange protocol. *Computers and Security* 26, 1 (2007), 94–97.
[115] Hui-Feng Huang. 2009. A simple three-party password-based key exchange protocol. *International Journal of Communication Systems* 22, 7 (2009), 857–862.
[116] Eun-Jun Yoon and Kee-Young Yoo. 2011. Cryptanalysis of a simple three-party password-based key exchange protocol. *International Journal of Communication Systems* 24, 4 (2011), 532–542.
[117] Shirisha Tallapally. 2012. Security enhancement on Simple Three Party PAKE Protocol. *Information Technology and Control* 41, 1 (2012), 15–22.
[118] Mohammad Sabzinejad Farash and Mahmoud Ahmadian-Attari. 2014. An enhanced and secure three-party password-based authenticated key exchange protocol without using server's public-keys and symmetric cryptosystems. *Information Technology and Control* 43, 2 (2014), 143–150.

[119] Yanrong Lu, Lixiang Li, Haipeng Peng, and Yixian Yang. 2015. A three-party password-based authenticated key exchange protocol for wireless communications. *Information Technology and Control* 44, 4 (2015), 404–409.
[120] Chien-Ming Chen, King-Hang Wang, Kuo-Hui Yeh, Bin Xiang, and Tsu-Yang Wu. 2019. Attacks and solutions on a three-party password-based authenticated key exchange protocol for wireless communications. *Journal of Ambient Intelligence and Humanized Computing* 10, 8 (2019), 3133–3142.
[121] Hu Xiong, Yan Wu, and Zhenyu Lu. 2019. A survey of group key agreement protocols with constant rounds. *ACM Computing Surveys* 52, 3 (2019), 57:1–57:32.
[122] Victor S. Miller. Use of elliptic curves in cryptography. In *Proceedings of the CRYPTO 1985*. 417–426.
[123] M. Koblitz. 1987. Elliptic curve cryptosystems. *Mathematics of Computation*. 48, 177 (1987), 203–209.
[124] Taher ElGamal. 1985. A public key cryptosystem and a signature scheme based on discrete logarithms. *IEEE Transactions on Information Theory* 31, 4 (1985), 469–472.
[125] E. Barker. 2020. *NIST special publication 800-57 part 1 revision 5*. DOI: https://doi.org/10.6028/NIST.SP.800-57pt1r5
[126] DongGook Park, Colin Boyd, and Sang-Jae Moon. Forward Secrecy and Its Application to Future Mobile Communications Security. In *Proceedings of the PKC 2000*. 433–445.
[127] Ronald L. Rivest, Adi Shamir, and Leonard M. Adleman. 1978. A method for obtaining digital signatures and public-key cryptosystems. *Communications of the ACM* 21, 2 (1978), 120–126.
[128] Oded Regev. On lattices, learning with errors, random linear codes, and cryptography. In *Proceedings of the STOC 2005*. 84–93.
[129] Jingwei Jiang and Ding Wang. 2024. Qpase: Quantum-resistant password-authenticated searchable encryption for cloud storage. *IEEE Transactions on Information Forensics and Security* 19 (2024), 4231–4246.
[130] Dimitrios Sikeridis, Panos Kampanakis, and Michael Devetsikiotis. Post-quantum authentication in TLS 1.3: A performance study. In *Proceedings of the NDSS 2020*. 1–16.
[131] Whitfield Diffie, Paul C. van Oorschot, and Michael J. Wiener. 1992. Authentication and authenticated key exchanges. *Des. Codes Cryptogr.* 2, 2 (1992), 107–125.
[132] Chunguang Ma, Ding Wang, and Sendong Zhao. 2014. Security flaws in two improved remote user authentication schemes using smart cards. *International Journal of Communication Systems* 27, 10 (2014), 2215–2227.
[133] E. Rescorla. 2018. *The Transport Layer Security Protocol Version 1.3*. Retrieved from https://www.rfc-editor.org/rfc/rfc8446.txt
[134] Ding Wang, Wenting Li, and Ping Wang. 2018. Measuring two-factor authentication schemes for real-time data access in industrial wireless sensor networks. *IEEE Transactions on Industrial Informatics* 14, 8 (2018), 4081–4092.
[135] Alan T. Sherman, Erin Lanus, Moses Liskov, Edward Zieglar, Richard Chang, Enis Golaszewski, Ryan Wnuk-Fink, Cyrus J. Bonyadi, Mario Yaksetig, and Ian Blumenfeld. Formal methods analysis of the secure remote password protocol. In *Proceedings of the Logic, Language, and Security 2020*. 103–126.
[136] Y. He, F. Sun, Q. Li, J. He, and L. Wang. A survey on public key mechanism in wireless sensor networks. *Chinese Journal of Computers*, 381–408.
[137] Jelle Don, Serge Fehr, Christian Majenz, and Christian Schaffner. Security of the Fiat-Shamir Transformation in the Quantum Random-Oracle Model. In *Proceedings of the CRYPTO 2019*. 356–383.
[138] Gorjan Alagic, Christian Majenz, Alexander Russell, and Fang Song. Quantum-access-secure message authentication via blind-unforgeability. In *Proceedings of the EUROCRYPT 2020*. 788–817.
[139] Jean-Sébastien Coron, Jacques Patarin, and Yannick Seurin. The random oracle model and the ideal cipher model are equivalent. In *Proceedings of the CRYPTO 2008*. 1–20.
[140] Ran Canetti. Universally composable security: A new paradigm for cryptographic protocols. In *Proc. FOCS 2001*.
[141] Michel Abdalla, Manuel Barbosa, Tatiana Bradley, Stanisław Jarecki, Jonathan Katz, and Jiayu Xu. Universally composable relaxed password authenticated key exchange. In *Proceedings of the CRYPTO 2020*. 278–307.
[142] SeongHan Shin, Kazukuni Kobara, and Hideki Imai. Very-efficient anonymous password-authenticated key exchange and its extensions. In *Proceedings of the AAECC 2009*. 149–158.
[143] Xuexian Hu, Jiang Zhang, Zhenfeng Zhang, and Jing Xu. 2017. Universally composable anonymous password authenticated key exchange. *Science China Information Sciences* 60, 5 (2017), 52107.
[144] Shai Halevi and Hugo Krawczyk. 1999. Public-key cryptography and password protocols. *ACM Transactions on Privacy and Security* 2, 3 (1999), 230–268.
[145] D. Wang and P. Wang. 2014. On the anonymity of two-factor authentication schemes for wireless sensor networks: Attacks, principle and solutions. *Computer Networks* 73, C (2014), 41–57.
[146] You Lyu, Shengli Liu, and Shuai Han. Efficient asymmetric PAKE compiler from KEM and AE. In *Proceedings of the ASIACRYPT 2025*. 34–65.
[147] You Lyu, Shengli Liu, and Shuai Han. Universal composable password authenticated key exchange for the post-quantum world. In *Proceedings of the EUROCRYPT 2024*. 120–150.

[148] A. Menezes, M. Qu, and S. Vanstone. 1995. Some new key agreement protocols providing mutual implicit authentication. (1995).
[149] Wei Chen, Derui Ding, Hongli Dong, and Guoliang Wei. 2019. Distributed resilient filtering for power systems subject to Denial-of-Service attacks. *IEEE Transactions on Systems, Man, and Cybernetics: Systems* 49, 8 (2019), 1688–1697.
[150] Songlin Hu, Dong Yue, Qing-Long Han, Xiangpeng Xie, Xiaoli Chen, and Chunxia Dou. 2020. Observer-based event-triggered control for networked linear systems subject to Denial-of-service attacks. *IEEE Transactions on Cybernetics* 50, 5 (2020), 1952–1964.
[151] Chris Peikert. Public-key cryptosystems from the worst-case shortest vector problem: Extended abstract. In *Proceedings of the STOC 2009*. 333–342.
[152] Chris Peikert, Vinod Vaikuntanathan, and Brent Waters. A framework for efficient and composable oblivious transfer. In *Proceedings of the CRYPTO 2008*. 554–571.
[153] Jonathan Katz, Rafail Ostrovsky, and Moti Yung. 2009. Efficient and secure authenticated key exchange using weak passwords. *Journal of the ACM* 57, 1, 3:1–3:39.
[154] Stanislaw Jarecki, Aggelos Kiayias, and Hugo Krawczyk. Round-optimal password-protected secret sharing and T-PAKE in the password-only model. In *Proceedings of the ASIACRYPT 2014*. 233–253.
[155] Chaim Gartenberg. 2018. *Twitter Advising All 330 Million Users to Change Passwords After Bug Exposed them in Plain Text*. Retrieved from https://bit.ly/3O8wHuE
[156] Ori Or-Meir, Nir Nissim, Yuval Elovici, and Lior Rokach. 2019. Dynamic malware analysis in the modern era - A state of the art survey. *ACM Computing Surveys* 52, 5 (2019), 88:1–88:48.
[168] Verizon. 2025. *2025 Data Breach Investigations Report*. Retrieved 29 October 2025. from https://www.verizon.com/business/resources/reports/2025-dbir-data-breach-investigations-report.pdf
[158] Yannan Li, Willy Susilo, Guomin Yang, Yong Yu, Dongxi Liu, Xiaojiang Du, and Mohsen Guizani. 2022. A blockchain-based self-tallying voting protocol in decentralized IoT. *IEEE Transactions on Dependable and Secure Computing* 19, 1 (2022), 119–130.
[159] Wenyi Xue, Yang Yang, Yingjiu Li, HweeHwa Pang, and Robert H. Deng. 2023. ACB-Vote: Efficient, flexible, and privacy- preserving blockchain-based score voting with anonymously convertible ballots. *IEEE Transactions on Information Forensics and Security* 18 (2023), 3720–3734.
[160] Muhammed F. Esgin, Ron Steinfeld, and Raymond K. Zhao. MatRiCT$^+$: More efficient post-quantum private blockchain payments. In *Proceedings of the IEEE S&P 2022*. 1281–1298.
[161] Stefan Dziembowski, Lisa Eckey, Sebastian Faust, and Daniel Malinowski. Perun: Virtual payment hubs over cryptocurrencies. In *Proceedings of the IEEE S&P 2019*. 106–123.
[162] G. Yuan, S. Wen, Q. Tang, Y. Ma, and S. Feng. 2021. Distributed authentication method of internet of things based on public blockchain. *Computer Engineering and Design* 51, 5 (2021), 1859–1866.
[163] A. W. Roscoe and Peter Y. A. Ryan. Auditable PAKEs: Approaching fair exchange without a TTP. In *Proceedings of the SPW 2017*. 278–297.
[164] Pawel Szalachowski. 2021. Password-authenticated decentralized identities. *IEEE Transactions on Information Forensics and Security* 16 (2021), 4801–4810.
[165] Nouri Alnahawi, David Haas, Erik Mauß, and Alexander Wiesmaier. SoK: PQC PAKEs - cryptographic primitives, design and security. In *Proceedings of the Cryptology ePrint Archive, Paper 2025/119*, Retrieved from https://eprint.iacr.org/2025/119
[166] Qingxuan Wang, Ding Wang, Chi Cheng, and Debiao He. 2023. Quantum2FA: Efficient quantum-resistant two-factor authentication scheme for mobile devices. *IEEE Transactions on Dependable and Secure Computing* 20, 1 (2023), 193–208.
[167] NIST. 2025. *Post-Quantum Cryptography: Selected Algorithms*. Retrieved 29 October 2025 from https://csrc.nist.gov/Projects/post-quantum-cryptography/selected-algorithms
[168] Taekyoung Kwon. 2004. Practical authenticated key agreement using passwords. In *International Conference on Information Security*. 1–12.

Received 10 July 2023; revised 8 July 2025; accepted 13 October 2025