Wiley
IET Information Security
Volume 2024, Article ID 7557514, 14 pages
https://doi.org/10.1049/2024/7557514

Research Article

PATS: Let Parties Have a Say in Threshold Group Key Sharing

Adnan Kılıç,[1,2] **Cansu Betin Onur**,[3,4] **and Ertan Onur**[1,4,5]

[1]*Department of Computer Engineering, METU, Ankara, Türkiye*
[2]*ASELSAN, Ankara, Türkiye*
[3]*Institute of Applied Mathematics, METU, Ankara, Türkiye*
[4]*David R. Cheriton School of Computer Science, University of Waterloo, Waterloo, Canada*
[5]*ARLEON, Ankara, Türkiye*

Correspondence should be addressed to Adnan Kılıç; e160697@metu.edu.tr

Received 9 August 2023; Revised 12 April 2024; Accepted 23 April 2024

Academic Editor: Helena Rifà-Pous

We present a password-authenticated (2, 3)-threshold group key share (PATS) mechanism. Although PATS resembles threshold secret sharing schemes, it has a different structure. The innovative perspective of the PATS mechanism that makes a difference from the standard secret-sharing schemes is that it involves parties in the generation of the shares. PATS allows parties to communicate securely to establish their shares over insecure channels. Parties (shareholders) construct a secret (key) using shares obtained at the end of the protocol. PATS takes advantage of zero-knowledge proofs compared to well-known threshold key exchange schemes and will tolerate the existence of semi-trusted parties. We present two variants of PATS, centralized and distributed, and then generalize PATS to (t, n)-threshold scheme. PATS supports the distributed operation and optionally facilitates group key verification by a trusted third party, which may also partake in group key sharing. In this paper, we present PATS, which employs finite fields and elliptic curves, along with its security and complexity analyses.

1. Introduction

There is a dealer (a trusted third party, TTP) and n parties in legacy secret-sharing schemes. A secret value v, which the dealer knows, is split into n different pieces called shares again by the dealer. The shares are sent to the corresponding parties over a secure channel. The dealer knows and controls all of these n shares. A secret sharing scheme is called (t, n)-threshold if t or more parties can recover v, but $(t-1)$ or fewer parties cannot easily do it. A (t, n)-threshold secret sharing scheme is said to be perfect if no information is disclosed when fewer than t parties try to construct the secret value v.

In this work, we present a password-authenticated $(2, 3)$-threshold group key-sharing (PATS) mechanism that resembles secret sharing schemes, albeit having a different construction involving parties in the generation of the shares. The significant difference is the lack of prior knowledge of the secret used as a group key in this work. Our approach can be better understood with a simple use case. Assume that Alice, Bob, and Carol write their wishes about the future on a piece of paper, store it in a time capsule secured with a padlock, and promise each other that they will meet after 10 years, open the capsule together using the physical key to the padlock and check whether or not their wishes will have come true. They lock the time capsule so that none of them can change the bet alone without others knowing it. They also want to open the capsule if a majority is present after 10 years. They solve this problem by *jointly* generating a physical key from some components. Assume that there are three components, say c_1, c_2, and c_3, required to construct the physical key, and each component is cloned. Each party receives two distinct components; for example, Alice gets c_1, c_2; Bob gets c_1, c_3; and Carol gets c_2, c_3. All the unique components (original or cloned) c_1, c_2, and c_3 must be present to construct the physical key, which can be created by combining the components held by any two parties. None of the parties can create the key alone since there is always a missing component. This scenario is illustrated in Figure 1.

Several practical scenarios may illustrate the need for the involvement of parties to generate shares of a group key. This approach may be employed in the business world, the digital services industry, or the financial industry. Let us give

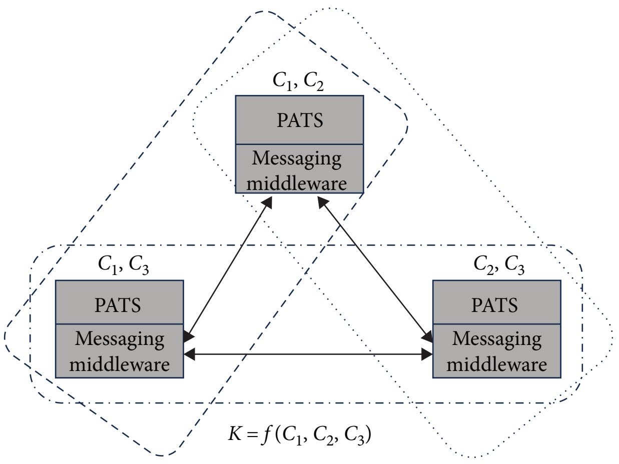

Figure 1: PATS scenario for the time capsule.

examples of centralized and distributed cases. In the distributed group key-sharing scenario, assume Alice, Bob, and Carol are critical infrastructure managers. Out of these three managers, at least two of them have to be present to execute a critical transaction. If Alice goes on a vacation, she must delegate her responsibility to Carol. In such a scenario, instead of Alice providing Bob with her secret, Carol will use her share of the key where a (2,3)-threshold group key sharing mechanism is implemented where shares involve input from all parties. None of the parties shall run the transaction alone. Carol and Bob can collaborate and run the transaction when Alice is off duty. The same scenario is applicable when any single party is off duty. In this scenario, parties do not need a centralized or trusted party while initializing the key shares or constructing the shared key using key shares. In the TTP-verified group key-sharing scenario, assume some transaction requires the approval of the TTP, such as a banking transaction. For example, Alice, Bob, and Carol, who have a say in a joint bank account, can only transfer money with the approval of the majority. If at least two out of three parties approve the transaction, the bank acting as the TTP approves and executes the transaction. Although the bank regulates the key shares, the users (not the bank) must determine shares jointly without knowing others' shares for secrecy. Another example is a password manager (keychain) that requires multiple devices to reveal a saved password, which is encrypted using PATS. In this case, a user who would like to reveal a saved password from a password manager has to interact with at least two out of the three predefined devices to collect two shares, generate the key, and decrypt the saved password. In all these scenarios, (i) none of the parties shall act alone, and (ii) the lack of a single party shall be tolerated.

In all these aforementioned scenarios, the challenge is to design a peer-to-peer threshold-based group key generation protocol where there is no central authority, and peers do not learn anything about the generated key other than their shares, although they are involved in the key generation. Furthermore, one of the peers should not act as an authority on behalf of the other peers. Based on this challenge, we can state our research question as to whether it is possible to construct a threshold-based group key involving only peers without any central authority or TTP and without any trusted setup. We address this research question and propose a password-authenticated (t, n)-threshold group key share (PATS) mechanism.

1.1. Challenges and Motivation. Yao et al. [1], Liu et al. [2], and Abdalla et al. [3] have proposed protocols that leverage a trusted server responsible for generating shares for each party. In contrast to performing operations on the client (or participant) side, Camenisch et al. [4] introduced an approach aimed at increasing the number of servers required for the verification step. Nevertheless, this type of solution is susceptible to phishing attacks.

Differing from the legacy key-sharing approaches, PATS does not require a centralized authority or trusted setup. Therefore, it can be employed in any peer-to-peer network, ad hoc networks, or in the Internet of Things (IoT) applications. As a simple example of an application area, assume that sensors in an IoT application are equipped with a low-entropy password that reflects that they belong to a group. However, using this password for securing communications will not be robust since its entropy is low. PATS allows us to secure this application by generating shares of a high-entropy threshold-based group key employing the low-entropy password in a distributed manner without entitling higher privileges to any one of the sensors (peers). In scenarios that require a centralized authority or trusted setup, the existing solutions give extra capabilities to this authority to generate the shares of parties. This approach will cause concerns in the case of a breach happening to the centralized authority.

The idea behind the different versions of PATS relies on the password-authenticated key agreement. There are different applications of password-authenticated protocols in the literature, such as password-authenticated group key exchange (PGKE) protocols, (t, n)-threshold secret sharing, or threshold password-authenticated key exchange (PAKE) protocols. A group key exchange protocol that is based on password authentication is called Group PAKE. There is "a fairy-ring dance" [5] construction to transform any PAKE protocol into a Group PAKE protocol. The common part of Group PAKE protocols is either the existence of a shared password with a common trusted server or the usage of pairwise passwords between group members. PATS also assumes a preconfigured common password known to all principals. To the best of our knowledge, Group PAKE protocols do not implement threshold-based key construction. Therefore, as a notable difference from the legacy Group PAKE protocols, PATS implements threshold-based group key construction without needing any TTP. On the other hand, the idea behind (t, n)-threshold secret sharing protocols is that each party has a portion of the secret, called a share, and the pieces belonging to at least t of the parties are sufficient for the construction of the secret. In these schemes, a secret that is known in advance is used to create shares, whereas we employ the reverse operation in PATS for different environments, either centralized or distributed. Moreover, there is always a TTP (dealer) in secret sharing protocols. In threshold-PAKE protocols, the concept of thresholding is different from that of secret sharing. In such schemes, a client is authenticated by multiple servers, and acquiring authentication from a subset of authentication servers is the idea behind thresholding. These are completely different constructions compared to PATS. Thresholding in PATS resembles the idea of (t, n)-threshold secret sharing.

1.2. Our Contributions. In this work, we present a password-authenticated $(2, 3)$-threshold group key-sharing mechanism that resembles secret-sharing schemes. Different than secret-sharing schemes, our construction involves parties in the generation of the shares. The significant difference is the lack of prior knowledge of the secret used as a group key in this work. Parties run a mutual agreement procedure, and pieces of information that help construct a group key are produced in a distributed fashion over insecure channels. The parties do not know others' shares. The shares are blended at the key usage stage. PATS does not require a fully TTP.

In summary, our contributions are four-fold:

(1) First, although PATS resembles threshold secret-sharing schemes, it has a different structure and approach, as demonstrated by the abovementioned use cases. There is no sharing of prior confidential information in PATS, and a trusted setup is not required.
(2) Second, in legacy secret sharing, there is a secret known to the dealer at the beginning of the protocol, and the shares of the secret are provided to the parties by the dealer. However, the secret (key) is not known in advance in PATS. PATS allows parties to construct a secret (key) using their shares obtained at the end of the protocol. In other words, legacy secret sharing starts from the secret and constructs the shares. PATS follows the opposite approach, where parties communicate to establish their shares securely over insecure channels. The shares will help them (or a TTP) construct the group key. When there is a data breach in a dealer of a legacy secret sharing, shares of each party may be exposed.
(3) Third, as the main contribution of this work, we propose a zero-knowledge (2,3)-threshold PAKE scheme that we call PATS, which involves parties in group key sharing. This protocol takes advantage of zero-knowledge proofs compared to well-known threshold key exchange schemes and will tolerate the existence of semi-trusted parties.
(4) Fourth, we present centralized and distributed PATS variants and then generalize PATS to (t, n)-threshold scheme. PATS supports the distributed operation and facilitates verification of the key by a TTP that may also partake in group key sharing.

1.3. Outline. In Section 2, we review several PAKE protocols from which PATS is inspired by threshold key sharing and threshold key exchange protocols. Section 3 presents (2,3)-threshold PATS mechanism. Two variants employing finite fields and elliptic curves are presented for the TTP-verified and distributed (2,3)-PATS. Section 4 generalizes the scheme to (t, n)-PATS. Section 5 is devoted to the security analysis of PATS based on the predefined security requirements. In Section 6, the PATS scheme is evaluated numerically. Finally, we will conclude the paper.

2. Related Work

In this section, we first review the PGKE protocols and then present threshold secret-sharing approaches by comparing the related work to PATS.

2.1. PGKE Protocols. A group key can be generated by employing PAKE protocols [1, 2, 5, 6, 7, 8, 9]. Two parties can use an authenticated key exchange (AKE) protocol to communicate securely. If the number of parties in a communication increases, a Group AKE protocol can be established. Instead of any public key infrastructure (PKI), a Group AKE protocol can be based on password authentication and is called Group PAKE (or PGKE). PGKE protocols enable parties to use low-entropy passwords to resist attacks coming from insiders. In the literature, PAKE protocols have essential contributions. However, PGKE has not received the same attention.

Asokan and Ginzboorg [10] proposed a PGKE protocol in ad hoc networks that do not provide formal security proof of the protocol. Bresson et al. [11] provide a provable security framework using Diffie–Hellman key exchange to assess the security of PGKE protocols. Yao et al. [1] proposed a PGKE protocol in which each party shares a different password with a trusted server. Using a similar approach to Bresson et al. [11], they provide formal proof of the protocol's security. Similarly, Liu et al. [2] proposed a password-based authentication key exchange protocol for groups, called nPAKE, in which each party shares a password with a trusted server and a temporary encryption key with the adjacent parties with the help of the server. The protocol's security is based on the chosen-based computation Diffie–Hellman assumption problem. It is also shown that the protocol is secure against dictionary attacks and man-in-the-middle attacks. In 2015, Hao et al. [5] presented a "fairy-ring dance" construction that transforms any PAKE protocol into a PGKE protocol. The paper provides two protocols, SPEKE+ and J-PAKE+, requiring 2 and 3 rounds consecutively. They showed that both protocols are secure against offline dictionary attacks while providing forward secrecy and known-session security. In 2018, Wei et al. [9] proposed a compiler that can convert any Group KE protocol into a secure PGKE protocol with two extra rounds of communication overhead. The paper includes the security proof of the compiler in the standard model.

2.2. Threshold Secret Sharing. Threshold secret sharing was invented separately by Blakley [12] and Shamir [13]. In (t,n)-threshold secret sharing, there are n parties that each hold a part of the secret, called shares, and t of the parties' part will reconstruct the secret. It is called threshold since less than t parties will not gain any information about the secret.

There are several well-known secret-sharing schemes applying different approaches, such as Shamir's Secret Sharing [13], Blakley's Secret Sharing [12], and Mignotte's and Asmuth-Bloom's Schemes [14]. Shamir's secret sharing is based on Lagrange interpolation over finite fields using the fact that to determine a polynomial of degree less than or equal to $t-1$ uniquely, t points are enough. This scheme is perfect since no information is leaked by the shared. Also, it is ideal since the length of each share is equal to the resulting secret. Blakley's secret sharing is based on hyperplane geometry. The dealer distributes a t-dimensional hyperplane equation over a field to each party. The intersection of these hyperplanes will give the secret. Asmut-Bloom secret sharing is based on the Chinese Remainder Theorem. Krawczyk's secret sharing is based on the existence of a one-way function. Having equal-length shares in a secret sharing scheme provides information-theoretic security. Instead of information-theoretic security, this work depends on computational security against resource-bounded adversaries.

A secret sharing scheme can be verifiable or publicly verifiable. In a verifiable secret sharing scheme, parties and the dealer verify the consistency of their shares in a 2-phase protocol. The notion of verifiability was first introduced by Chor et al. [15]. In the sharing phase, the dealer distributes the secret to n parties such that less than t parties will not deduce any information, similar to threshold secret sharing. In the reconstruction phase, each party brings their information to reconstruct the secret.

There are several well-known verifiable secret-sharing schemes, such as Chor–Goldwasser–Micali–Awerbuch's scheme [15], Feldman's scheme [16], and Benaloh's scheme [17].

Threshold secret sharing is also used in password-based single-sign-on authentication schemes. Zhang et al. [18] proposed a secure and efficient single-sign-on authentication scheme for mobile users. The protocol is shown to provide stronger security compared to existing schemes under the defined adversary model. Jiang et al. [19] proposed a password-based single-sing-on authentication protocol based on lattices. Using lattices, the proposed protocol is quantum-resistant and resistant to offline password-guessing attacks.

There are threshold secret-sharing protocols that take advantage of authentication servers. In this case, when t of the n authentication servers communicates, the authentication is successful. Abdalla et al. [3] proposed a threshold password-based authenticated key exchange (GTPAKE) protocol in which authentication servers act as two different parties, consisting of a gateway that handles the communication with the clients and a back-end-server used to check the identity of a client. Camenisch et al. [4] proposed a protocol in which the verification process is distributed to multiple servers, and the password will remain secure as long as the number of breached servers is less than the threshold. This type of threshold password-authenticated secret sharing, called TPASS, was previously proposed by Bagherzandi et al. [20] and Camenisch et al. [21]; however, they are vulnerable to phishing attacks in which users interact with malicious servers.

2.3. Group Key Agreement (GKA) Protocols. In group communications, GKA protocols are widely used to provide secure communication. There are two types of GKA protocols, namely centralized GKA and distributed GKA. In their paper, Zhang et al. [22] proposed a threshold authenticated group key protocol based on Shamir's secret sharing for unmanned aerial vehicle (UAV) ad hoc network (UANET). Since the environment for UAVs requires dynamic interaction and wireless communication, which would cause temporary disconnection from a group, having a stable GKA is important to address these problems. Their solution considers the disconnection of group members, as well as a group key recovery protocol in case a disconnected tries to reconnect and obtain the latest group key.

3. PATS: Password-Authenticated (2,3)-Threshold Key Exchange

In this section, we present the details of the zero-knowledge (2,3)-threshold password-authenticated key exchange (PATS) schemes where shares are constructed mutually over public channels. PATS uses the general approach of the PAKE protocol that we will briefly present in the sequel. After presenting the (2,3)-threshold PATS, we will generalize this approach to n parties and present the (t,n)-threshold PATS. First, we will define the distributed approach called distributed (2,3)-PATS, where there is no TTP in the protocol, and each party creates the shares of a group key. We present two variants of this scheme. The first is implemented in finite fields, and the second is implemented using elliptic curves. Then, we present the centralized approach that requires a TTP.

3.1. PATS Inspired by PAKE. A PAKE protocol provides two or more parties a secure authenticated communication over an insecure channel and establishes a high-entropy cryptographic key using a shared low-entropy secret such as a humanly-memorable password. While many password-based protocols risk offline attacks [23], the AKE protocols prevent such vulnerabilities. Among many other PAKE protocols, treating PAKE as a secure two-party computation problem, password-authenticated key exchange by juggling (J-PAKE) stands out from the rest.

J-PAKE depends on zero-knowledge proofs [24, 25, 26]. J-PAKE, standardized in ISO/IEC 11770-4 and RFC 8236 [27, 28], is a balanced PAKE protocol, and it does not require any PKI deployments, Hash-to-Group function, or a trusted setup. Such independence makes J-PAKE's implementation simple. The protocol is used over finite fields and elliptic curves. We give a brief overview of J-PAKE over a finite field.

J-PAKE relies on the hardness assumption of the decision Diffie–Hellman (DDH) problem. Two parties, Alice and Bob, have a shared password s, and they agree on an element g of $\mathbb{Z}_p^*$ of prime order q. Both parties randomly choose their private key pairs where the first component is in the range $[0, q-1]$, and the second component is in the range $[1, q-1]$. Say Alice chooses (x_1, x_2) and Bob chooses (x_3, x_4). They compute and interchange all g^{x_i} values over an unsecured network with the zero-knowledge proofs of the exponents. In the second round, Alice computes $A = g^{(x_1+x_3+x_4)x_2 s}$ and Bob computes $B = g^{(x_1+x_2+x_3)x_4 s}$. They send these values to each other with a ZKP of their secret exponents $x_2 s$ and $x_4 s$. Now both can compute the keying material $K = (A/g^{x_2 x_4 s})^{x_4} = g^{(x_1+x_3)x_2 x_4 s} = (B/g^{x_2 x_4 s})^{x_2}$. Then the session key is derived by a hash (or key generation) function $k = H(K)$.

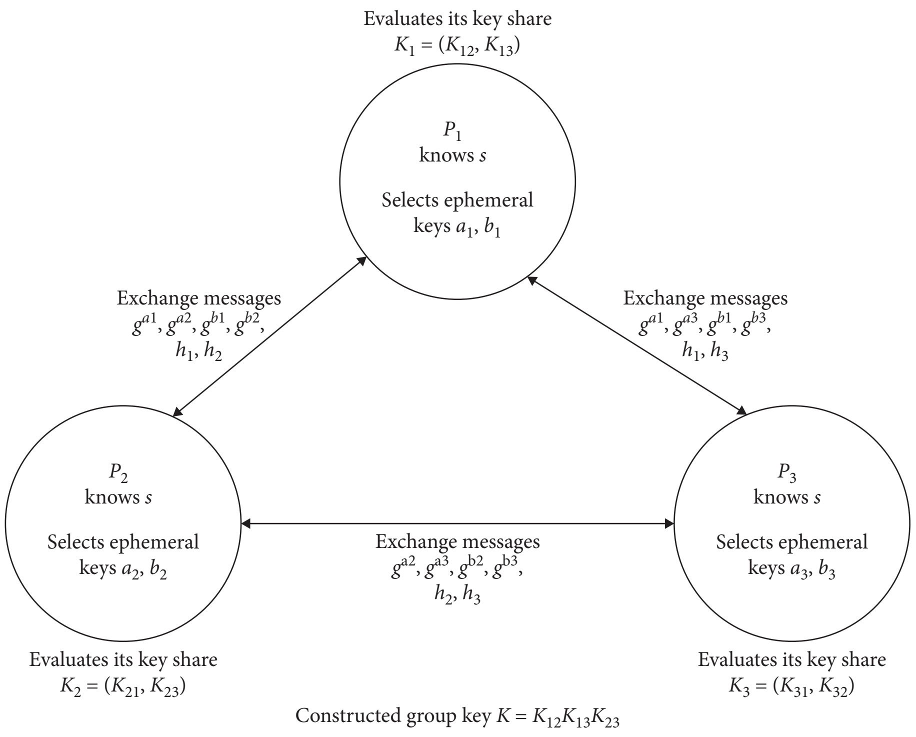

FIGURE 2: Architectural diagram of distributed (2,3)-PATS in finite fields.

PATS is inspired by PAKE. It relies on the DDH problem's hardness assumption. As the shares are constructed, it uses the juggling method of J-PAKE and provides mutual authentication among three parties via a shared password.

3.2. Design Challenges. To develop PATS, security and efficiency are the two important motivations that we focused on:

(1) Security: Using a PAKE protocol in the protocol's core allows us a way to establish a high-entropy cryptographic key from a low-entropy secret. While password-based protocols are prone to offline attacks, the AKE protocols overcome this vulnerability. The protocol also takes advantage of zero-knowledge proofs compared to well-known threshold key exchange schemes.
(2) Efficiency: One variant of the PATS depends on the elliptic curves. The computation over elliptic curves is known to use smaller coefficients but provides similar security strength compared to finite field calculation.

According to Zipf's law [29, 30], the space of user-chosen passwords is constrained compared to the space of all possible passwords. It is stated that the advantage for guessing will be over 0.5 in some cases. In PATS, the final secret key has different components. For the TTP-verified version, the b_i values are provided by the TTP, and a_i values are generated by the parties. For the distributed version, the a_i and b_i values are generated by the parties. However, for any party P_i, accessing the values a_j and b_j for $j \neq i$ is equal to having a solution to the DDH problem. The secret key K also requires the value of s that is shared between parties. Therefore, Zipf's law does not have an impact on the value of K.

3.3. Distributed (2,3)-PATS. In some scenarios, such as the one we mentioned earlier in Section 1, about Alice and her friends' time capsule, the parties do not require any centralized entity (e.g., a TTP). They may have to run a distributed protocol for sharing the group key. Both ephemeral key integers, a_i and b_i, are selected by the party P_i. There is no TTP verification. Whenever a group key is necessary, any two among three parties use their shares to construct the group key. This section presents two variants that use finite fields or elliptic curves.

3.3.1. Distributed (2,3)-PATS in Finite Fields. Assume there are three parties, P_1, P_2, and P_3, who share a common low-entropy password s in the range $[1, q-1]$. The parties will establish a common (2,3)-threshold group key. Any two out of three parties can construct the group key collaboratively. Let p and q be large primes such that q is a divisor of $p-1$. Let g be an element of $\mathbb{Z}_{\mathbb{p}}^*$ of order q. The communicating parties, P_1, P_2, and P_3 agree on the (p, q, g) value string, which can be hard-wired in software. The architectural diagram of this protocol is given in Figure 2. Distributed (2,3)-PATS involves three rounds, as shown in Figure 3.

(1) Round 1: Each party P_i for $i = 1, 2, 3$ selects ephemeral private key integers a_i uniform randomly from $[0, q-1]$, b_i uniform randomly from $[1, q-1]$ and then broadcasts $g^{a_i} \bmod p$ and $g^{b_i} \bmod p$ together with the zero-knowledge proofs of the exponents a_i, b_i. For example, Schnorr's non-interactive zero-knowledge (NIZK) proof, as used in J-PAKE, can be employed here [31]. When the round finishes, each party checks that the received ZKPs are valid. Also, each party P_i checks

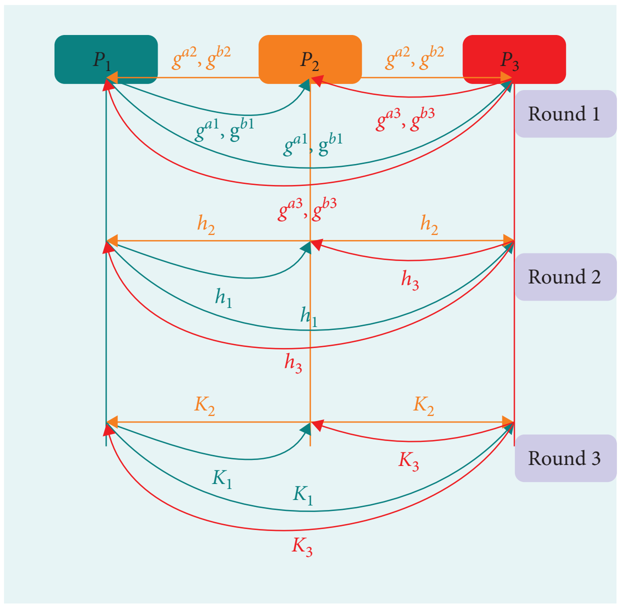

FIGURE 3: Distributed (2,3)-PATS in finite fields.

that $g^{b_j} \not\equiv 1 \bmod p$ for $j \neq i$. If any of these checks fail, this session should be canceled.

(2) Round 2: Each party P_i computes and broadcasts $h_i = (g^{\sum_{j=1}^{3} a_j + \sum_{j \neq i} b_j})^{b_i s}$ together with the zero-knowledge proofs of the exponent $b_i s$. Again, at the end of the round, each party checks that the received ZKP is valid. Also, party P_i should ensure that the elements h_j and h_t are non-identity.

(3) Round 3: Each party P_i computes its secret shadow K_i where $K_i = (K_{ij}, K_{il}) = ((h_j(g^{b_j b_i s})^{-1})^{b_i}, (h_l(g^{b_l b_i s})^{-1})^{b_i})$. Notice that the indices commute $K_{ij} = K_{ji}$.

Whenever the necessity of a group key emerges, any two among three parties use their shares and construct the group key $K = K_{ij}K_{il}K_{jl} = K_{ji}K_{jl}K_{li} = K_{ij}K_{il}K_{lj}$. Figure 3 shows the flow of the protocol. Note that instead of directly using K, it can be used as input to a key generation function.

Claim 1. $K_{ij} = K_{ji}$

Proof.

$$\begin{aligned}
K_{ij} &= \left(h_j\left(g^{b_j b_i s}\right)^{-1}\right)^{b_i} \\
&= \left(\left(g^{\sum_{k=1}^{3} a_k + \sum_{k \neq j} b_k}\right)^{b_j s}\left(g^{b_j b_i s}\right)^{-1}\right)^{b_i} \\
&= \left(g^{\sum_{k=1}^{3} a_k b_j s + \sum_{\substack{k \neq i \\ k \neq j}} b_k b_j s}\right)^{b_i} \\
&= \left(g^{\sum_{k=1}^{3} a_k b_i s + \sum_{\substack{k \neq i \\ k \neq j}} b_k b_i s}\right)^{b_j} \\
&= \left(\left(g^{\sum_{k=1}^{3} a_k + \sum_{k \neq i} b_k}\right)^{b_i s}\left(g^{b_i b_j s}\right)^{-1}\right)^{b_j} \\
&= \left(h_i\left(g^{b_i b_j s}\right)^{-1}\right)^{b_j} \\
&= K_{ji}.
\end{aligned} \quad (1)$$

□

In practice, the final step will be combining the shares from at least two parties and generating the group key without revealing each share independently to other parties. Let us now revisit the time capsule example presented in Section 1. The time capsule is a trusted combiner that does not copy the shared values given as input. Each party uses its shares (in this case, the physical key components that are used to construct the key of the padlock) to open the padlock on the time capsule. To open the time capsule, at least two parties must come together. Neither the combiner nor the remaining parties have any idea about the value of each share of other parties.

3.3.2. Distributed (2,3)-PATS Using Elliptic Curves. Initially, all parties agree on the domain parameters. Namely, they agree on the sequence (p, a, b, G, q, h) where p and q are large primes, the elliptic curve E is the curve satisfying the equation $y^2 = x^3 + ax + b$ defined over the prime field F_p, the base point G has order q, and h is the cofactor, a small integer. One may wish to use recommended elliptic curves by NIST for key establishment schemes in SP 800-56A [32]. As in the finite field case, we assume there are three parties P_1, P_2, and P_3 share a common low-entropy password s in the range $[1, q-1]$, and they execute the following three rounds:

(1) Round 1: Each party P_i selects an ephemeral private key integers a_i uniformly random from $[0, q-1]$, b_i uniformly random from $[1, q-1]$ and then broadcasts a_iG and b_iG, together with the zero-knowledge proofs of the multiples a_i, b_i. At the end of the round, each party P_i should check that the received ZKPs are valid and b_jG a non-identity element for $j \neq i$. If any of these checks fail, this session should be canceled.

(2) Round 2: Each party P_i computes and broadcasts $H_i = sb_i(\sum_{j=1}^{3} a_j + \sum_{j \neq i} b_j)G$ together with the zero-knowledge proofs of the multiple sb_i. At the end of the round, each party checks that the received ZKPs are valid and that the elements H_j and H_k are non-identity. If any of these checks fail, the session will be canceled.

(3) Round 3: Each party P_i computes their secret shares $K_i = (K_{ij}, K_{il})$ where $K_i = (K_{ij}, K_{il}) = (b_i(H_j - b_j b_i sG), b_i(H_l - b_l b_i sG))$. Notice that the indices commute $K_{ij} = K_{ji}$.

3.4. TTP-Verified (2,3)-PATS. In some scenarios, such as the banking use-case mentioned above, the secret will eventually be employed by a centralized entity (e.g., a TTP), for example, for encrypting a database or authenticating a transaction. In such scenarios, the TTP has to use the group key for conducting the operation where the TTP-verified (2,3)-PATS can be employed. In Figure 4, we illustrate the architectural diagram of the proposed protocol. The protocol runs as in the previous case. However, the only difference is that ephemeral key integer b_is are selected by the TTP and sent to the parties over a secure channel. Figure 5 shows the flow of the protocol.

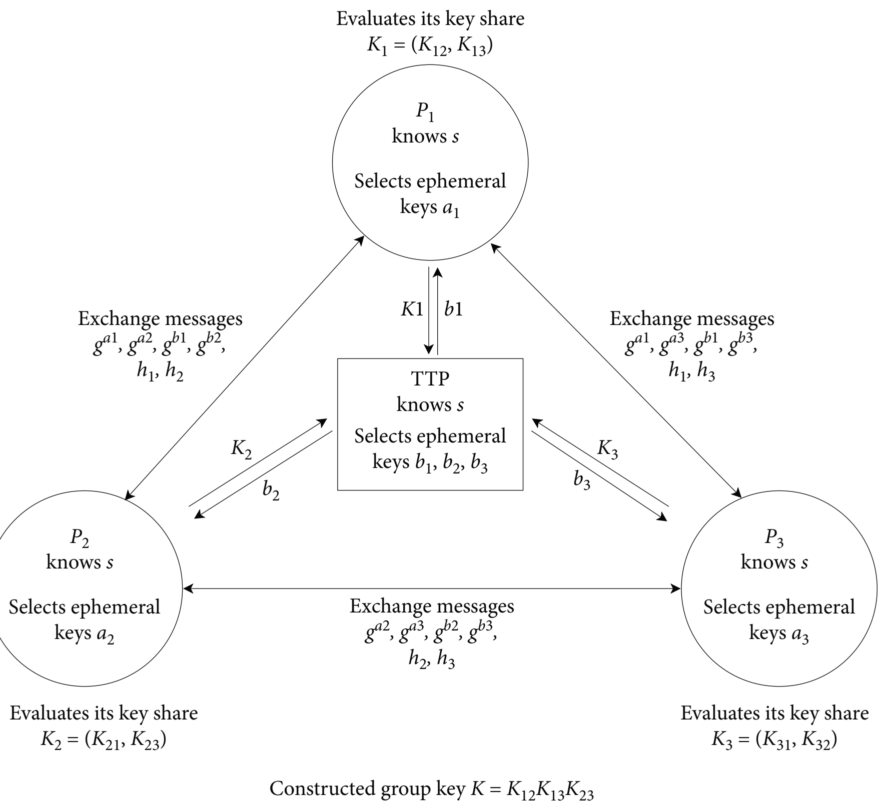
Evaluates its key share
$K_1 = (K_{12}, K_{13})$
P_1
knows s
Selects ephemeral keys a_1
$K1$
$b1$
TTP
knows s
Selects ephemeral keys b_1, b_2, b_3
Exchange messages
$g^{a1}, g^{a2}, g^{b1}, g^{b2}, h_1, h_2$
Exchange messages
$g^{a1}, g^{a3}, g^{b1}, g^{b3}, h_1, h_3$
K_2
b_2
K_3
b_3
P_2
knows s
Selects ephemeral keys a_2
P_3
knows s
Selects ephemeral keys a_3
Exchange messages
$g^{a2}, g^{a3}, g^{b2}, g^{b3}, h_2, h_3$
Evaluates its key share
$K_2 = (K_{21}, K_{23})$
Evaluates its key share
$K_3 = (K_{31}, K_{32})$
Constructed group key $K = K_{12}K_{13}K_{23}$

FIGURE 4: Architectural diagram of TTP-verified (2,3)-PATS in finite fields.

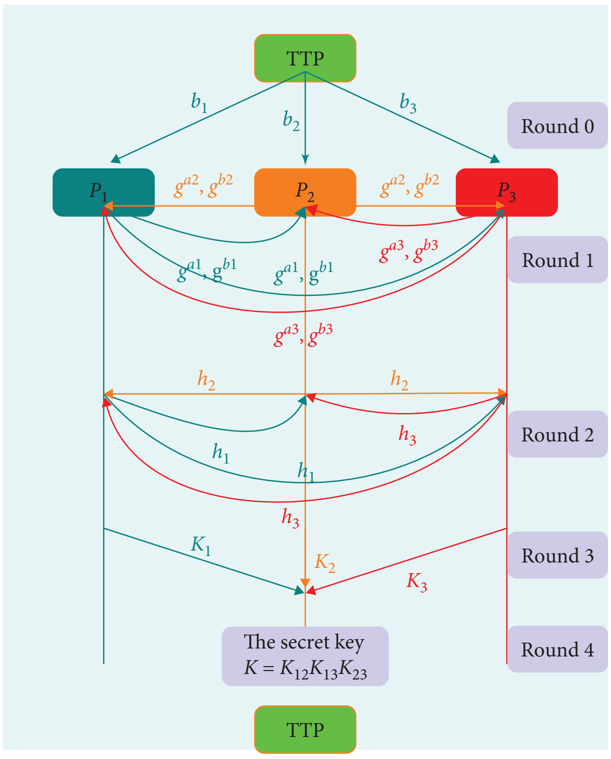

FIGURE 5: TTP-verified (2,3)-PATS in finite fields.

Note that in the TTP-verified (2,3)-PATS, not only is the TTP involved in the process and has a say in the constructed shares, but also all the parties involved in the process as well. The TTP verification has two steps. First, it calculates the shares: TTP has the values s, b_1, b_2, and b_3. Therefore, by gathering h_1, h_2, and h_3 values from the public channel, it can recover all shares by imitating the operations done by the parties. Second, it checks whether these values match the shares sent by the parties. Here the equality comes from the fact that indices of the shared shares commute.

Notice that any two parties among three can recover K. Using K as input, the parties may apply a key derivation function to derive a common session key k.

In this section, we have presented two variants of the PATS protocol, employing finite fields and elliptic curves, respectively. In the next section, we will generalize the scheme to a (t, n)-threshold scheme.

4. (t,n)-PATS

In this section, we generalize the (2,3)-PATS to (t,n)-threshold scheme. Considering the efficiency requirement, in cases where t is less than $n-1$, the authors think that the path followed in PATS will be cumbersome. We picture the scheme for distributed variants over finite fields. The generalized version can be phrased straightforwardly on elliptic curves, mutatis mutandis as above, for centralized (t,n)-PATS over finite fields and elliptic curves. Assume there are n parties who share a common low-entropy password s in the range $[1, q-1]$, and will establish a common (t, n)-threshold group key. That is any t among n will have the group key. Here, s can be a low-entropy group key or a session nonce. A TTP (dealer) and the communicating parties, $P_1, \ldots, P_n$ agree on (p, q, g) value string, which can be hard-wired in the software.

For $t = n - 1$, (t,n)-PATS procedure is as follows:

(1) Round 1: Each party P_i selects ephemeral private key integers a_i and b_i uniformly at random from $[0, q-1]$ and $[1, q-1]$, respectively. Then broadcasts $g^{a_i} \bmod p$ and $g^{b_i} \bmod p$ together with the zero-knowledge proofs of the exponents a_i, b_i. When the round finishes, each party checks that the received ZKPs are valid. Also, each party P_i checks that $g^{b_j} \not\equiv 1 \bmod p$ for $j \neq i$. If any of these checks fail, this session should be canceled.

(2) Round 2: Each party P_i computes and broadcasts $h_i = (g^{\sum_{j=1}^{n} a_j + \sum_{j \neq i} b_j})^{b_i s}$ together with the zero-knowledge proofs of the exponent $b_i s$. Again, at the end of the round, each party P_i checks that the received ZKPs and the received elements h_j for $j \neq i$ are nonidentities.

(3) Round 3: Each party P_i computes their secret shares sequence $K_i = (K_{ij})_{j \neq i}$ where $K_{ij} = (h_j \times (g^{b_j b_i s})^{-1})^{b_i}$.

After the first three rounds end, $(n-1, n)$-PATS is completed. When $t = n - r$, taking $(n-1, n)$-PATS as the initial step, (t, n)-PATS will be calculated iteratively. Each iteration step consists of two rounds. Suppose in the $(k-1)^{th}$ iteration step each participant P_i computed its key shares $K_i = \{K_{ij_1 \ldots j_{k-1}}\}$ where j_i's are distinct entries from $I = \{1, \ldots, n\} \setminus \{i\}$. At kth iteration step, in Round 1, each P_i calculates the exponent set $\{(g^{b_{j_1} \cdots b_{j_{k-1}}})^{b_i}\}$ where indices j_r ranges over I and distinct. After the first round of the kth iteration step, the set $\{(g^{b_{j_1} \cdots b_{j_{k-1}}})^{b_i}\}_{j_r \in I}$ will be broadcasted with the keyshare sequence K_i. In the second round of the kth iteration step, P_i computes the new key share sequence $K_i = \{K_{ij_1 \ldots j_k}\}_{j_r \in I}$ where $K_{ij_1 \ldots j_k} = [K_{j_1 \ldots j_k} \times [(g^{b_{j_1} \cdots b_{j_k}})^{b_i s}]^{-1}]^{b_i}$.

In this section, we have how the (t, n)-threshold generalization of PATS works. In the next section, we will first give the adversary model and define security requirements that PATS shall satisfy. The section will end with a detailed analysis of PATS, showing how the mentioned requirements are fulfilled.

5. Security Analysis of PATS

In this section, we will discuss an adversary model stating the capabilities of an attacker, common security requirements of PAKE protocols, and how PATS satisfies the security requirements.

5.1. Adversary Model. To define security requirements and show that PATS satisfies them, it is important to understand the capabilities of an attacker. An adversary A is assumed to have the following capabilities:

(1) Similar to conventional PAKE protocols, A is assumed to have full control over the communication channel between parties, eavesdropping, tampering, deleting, and changing messages gathered over the public channel.

(2) A can generate all possible values of s since PATS assumes that low-entropy password s is in the range $[1, q-1]$.

(3) A is able to learn the previous session keys a_i's and b_i's.

5.2. Security Requirements. PATS shall satisfy the following security requirements:

(1) Offline dictionary attack resistance: An active/passive attacker has no advantage in verifying the correctness of password guesses.

(2) Forward secrecy [33]: If the password is disclosed later, the current session keys will remain secure.

(3) Known-session security: If any session is disclosed, other established sessions will not be affected and will remain secure.

(4) Online dictionary attack resistance: An active attacker is limited to testing only one password per execution of the protocol.

(5) Man-in-the-middle attack resistance [2]: Man-in-the-middle attack happens when an active attacker stays in the middle of the communication to relay and alter messages. At the same time, parties believe that they directly communicate with each other.

(6) Replay attack resistance [34, 35]: Replay attack is a man-in-the-middle attack in which the communication is delayed or replayed by a passive attacker.

5.3. Analysis. Previously, we have defined an adversary model. According to this model, first, an attacker A is assumed to have full control over the communication channel between parties, being able to eavesdrop, tamper, delete, and change messages. Since the protocols include zero-knowledge proofs of the exponents a_i's and b_i's, and they depend on the DDH problem, accessing each value to generate K_i's is not feasible. Second, A can generate all possible values of s. However, s is not the only component of the secret key K. A should also know a_i and b_i values, as well as the value of g. Finally, A can learn the previous session keys a_i's and b_i's. Since every time PATS runs, parties are expected to select new a_i and b_i values. Knowing any previous session keys would not provide any advantage.

PATS provides several security features, some of them coming from the existence of zero-knowledge proofs and the use of threshold group key sharing in the protocol. Being a threshold key exchange scheme, failure or malfunction of some entities will not affect the scheme. Owing to the following lemma, we can ensure the resistance of PATS to offline dictionary attacks.

Lemma 1. *For any party P_k, if $k \neq i$ the exponent $X_i = \sum_{j=1}^{n} a_j + \sum_{j \neq i} b_j$ is a secret random value over $\mathbb{Z}_q$.*

Proof. a_i and b_i values are uniform randomly selected over the intervals $[0, q-1]$ and $[1, q-1]$, respectively. P_k only knows the value of a_k; the other summands of X_i are unknown to P_k. The value $X_i - a_k$ ranges over the interval $[0, q-1]$. Although b_i values can not be zero, for the sum $\sum_{j \neq i} b_j$, the probability of non-zero values are equal and the probability difference $P(0) - P(x \neq 0) = \frac{1}{(q-1)^2}$ is negligible. Therefore, the value $X_i - a_k$ can be regarded as uniformly distributed over $\mathbb{Z}_q$. □

Theorem 1. *PATS is secure against offline dictionary attacks under the DDH assumption.*

Proof. We examine this security in two cases: active and passive attacks. First, we will show that a malicious attacker pretending as a participant P_j but does not have the password s, cannot distinguish the ciphertext h_i from a random non-identity element of G. This case corresponds to active attacks. At the end of the second round of the protocol, the information that the attacker gathers from P_i is the values g^{a_i}, g^{b_i}, $h_i = (g^{X_i})^{b_i s}$ and the corresponding ZKP's of exponents. In the second round, P_i sends $h_i = (g^{X_i})^{b_i s}$ where X_i as given in Lemma 1. Here, P_i allocates the element g^{X_i} as a random generator. The randomness follows from Lemma 1. It is well-known that any nontrivial element of a prime order group is a generator [36]. The element g^{X_i} is nontrivial with high probability, and this can be guaranteed by P_i by simply checking it. Therefore, g^{X_i} is a generator. The password s and the random value b_i are in the interval $[1, q-1]$ and are unknown random values for the attacker. Therefore, by the DDH assumption, the malicious party cannot distinguish the value h_i from a random non-identity element in the group.

In the second case, suppose a passive attacker gathers the information floating on the air while two authentic participants P_i and P_j are communicating. The attacker has no better knowledge than the first case. So, it can neither correlate the values h_i and h_j nor distinguish them from a random non-identity element in the group. □

We know that the original Diffie–Hellman key exchange method is vulnerable to a man-in-the-middle attack. Since Round 1 and Round 2 of the suggested protocols include zero-knowledge proof of the values b_i and $b_i s$, an active attacker will not be successful without knowing the value of s; this provides resistance to the man-in-the-middle attacks.

We previously mentioned that our protocol relies on the hardness assumption of the DDH problem. Moreover, we noted that p and q are large primes such that q is a divisor of $p-1$. DDH is believed to be intractable in all the groups when $p = kq + 1$ for prime numbers p and q and $q > p^{1/10}$ [37].

In the implementation, we considered p to be a safe prime such that $p = 2q + 1$, where q is also a prime [38]. This assumption makes PATS rely on the DDH problem's hardness assumption since $q = (p-1)/2 > p^{1/10}$ for sufficiently big values of p.

Suppose a malicious entity was able to disclose the value of K. In that case, the security of other sessions will not be disrupted. This comes from the necessity that each party in the protocol is expected to select a private integer a_i uniformly at random from $[0, q-1]$ at each run of the scheme. This feature is known as known-session secrecy [24].

Using a zero-knowledge protocol for the exponents a_i and b_i is accepted to be secure since a party does not have to reveal these values to other parties and parties who receive g^{a_i} and g^{b_i} will verify that these values belong to the sender.

It is not surprising that PATS, in the spirit of JPAKE, provides similar security requirements. The footprints in JPAKE give the idea, while below, we show that PATS provides forward secrecy. Theorem 2 shows that PATS provides forward secrecy under the square computational Diffie–Hellman assumption (SCDH).

Theorem 2. *In PATS protocol, under the SCDH assumption, the current session key components will remain secure if the password is disclosed later.*

Proof. Suppose the password s is captured by a malicious attacker. We will observe that the previously derived K_{ij} components can not be disclosed if the SCDH assumption holds. Note that the PATS protocol is designed in such a way that no key component is the identity element. We consider the key component K_{12}. Mutatis mutandis one may derive the same conclusion for the other key components. To ease the formulation set $A = a_1 + a_2 + a_3$. The idea is if an attacker can obtain $K_{12} = g^{(A+b_3)b_1 b_2}$ from the data set $\{g^{a_1}, g^{a_2}, g^{a_3}, g^{b_1}, g^{b_2}, g^{b_3}, g^{(A+b_1+b_2)b_3}, g^{(A+b_1+b_3)b_2}, g^{(A+b_2+b_3)b_1}\}$ then it can compute g^{x^2} from g and g^x for any uniform randomly chosen x value in range $[1, q-1]$, which contradicts the SCDH assumption.

Let attacker have an algorithm, which can return $K_{12} = g^{(A+b_3)b_1 b_2}$ for the ordered input set $\{g^A, g^{b_1}, g^{b_2}, g^{b_3}, g^{(A+b_1+b_2)b_3}, g^{(A+b_1+b_3)b_2}, g^{(A+b_2+b_3)b_1}\}$. So for g^x with random exponent x in $[1, q-1]$ the algorithm gives the evaluation $f(g^x) = g^{[(-2x+a)(x+d)](x+b)(x+c)} = g^{-x^3+(-c-b+a+d)x^2+(-bc+ac+ab+dc+db)x+abc+bcd}$ for the given data set $\{g^{-2x+a}, g^{x+b}, g^{x+c}, g^{x+d}, g^{(x+a+b+c)(x+d)}, g^{(x+a+b+d)(x+c)}, g^{(x+a+c+d)(x+b)}\}$ where a, b, c, d are arbitrary chosen in $\mathbb{Z}_q$. Similarly, we determine $f(g^{x+1})$. It follows that using the given information one can evaluate the element $g^{x^2} = [f(g^x) f(g^{x+1})^{-1} g^{(2a-2b-2c+2d)x+(-1+a-b-c+d-bc+ac+ab+dc+db+abc+bcd)}]^{\frac{1}{3}}$. □

PATS is information-theoretically secure [39] since less than the required number of shares will provide no information about the secret. Moreover, the share of each party is as long as the final secret K.

An active attacker is limited to testing/checking only one s value per execution of the protocol since parties will refresh a_i and b_i values at each run of the protocol. PATS prevents online dictionary attacks.

In this section, we have given an adversary model and analyzed the security requirements that PATS satisfies. In the

Table 1: Analysis of computational cost of PATS in finite fields.

	Cost breakdown	No. of operations per party	No. of broadcast messages
0	Round 0	—	3*
1	Round 1	2 E, 2 (ZKP)	6 + 2 [3]
2	Round 2	1 E, 5 M, (ZKP)	6 + [3]
3	Round 3	4 E, 2 M, 2 A	6 − 3*
4	Total	7 E, 7 M, 2 A, 3 (ZKP)	18 + 3 [3]

[1]E, exponentiation in $\mathbb{Z}_q$. [2]M, multiplication in $\mathbb{Z}_q$. [3]A, addition in $\mathbb{Z}_q$. [4](ZKP) = (3 E, 2 M, 1 A), # of operations required for a single ZKP. [5][x], # of messages required for a single ZKP. *TTP-verified (2,3)-PATS requires three broadcast messages from TTP in Round 0, and three fewer messages in Round 3.

next section, the performance analysis of PATS, including computational load and run-time analysis, will be provided for each variant of the protocol defined previously. The section will include a comparison of two variants in terms of run-time.

6. Performance Analysis of PATS

In this section, we present the performance analysis of PATS. First, we discuss the computation overhead of PATS and then present the running time results of PATS.

Both protocols are implemented in Python on a PC (8 CPU Cores, 16 GB RAM) running macOS Monterey. ZKP proof is not included in the implementation, and parties are not separated to cause messaging costs; i.e., messaging delays are not incorporated.

Prime numbers of different lengths are selected using OpenSSL prime generation with the "safe" parameter. If the number generated is p, this feature helps us check that $(p-1)/2$ is also a prime number.

6.1. PATS in Finite Fields

6.1.1. Computational Load. A party P_i needs to do the following computations in a round of PATS protocol in finite fields:

(1) g^{a_i}
(2) g^{b_i}
(3) $b_i s$
(4) $g^{\sum_{j=1}^{3} a_j + \sum_{j \neq i} b_j} = g^{a_1} \times g^{a_2} \times g^{a_3} \times g^{b_j} \times g^{b_k}$
(5) $h_i = \left(g^{\sum_{j=1}^{3} a_j + \sum_{j \neq i} b_j}\right)^{(b_i s)}$
(6) $K_i = (K_{ij}, K_{it})$ where $K_{ij} = \left(h_j \times (g^{b_j})^{-b_i s})\right)^{b_i}$

In Round 1, it takes two exponentiations to compute $g^{a_i} \bmod p$ and $g^{b_i} \bmod p$ and two additional exponentiations will be necessary to compute commitments for the zero-knowledge proofs of the multiples a_i and b_i if Schnorr non-interactive ZKP will be applied. To prove that $g^x = y$, the prover must perform one exponentiation as a commitment. After Round 1, every party should verify two Schnorr ZKPs. For each ZKP, it takes one exponentiation and one multiplication for the verifier to verify and one multiplication and one addition for the prover to generate a response. In Round 2, there will be four multiplications of previous calculations and broadcast messages, one multiplication to compute the power $b_i s$ and one exponentiation to compute h_i value. To compute the zero-knowledge proof of the exponent $b_i s$, additional exponentiation will be necessary to compute a commitment for the zero-knowledge proofs of the multiple $b_i s$. To verify this Schnorr ZKP, each verifier computes one exponentiation and one multiplication, and the prover computes one multiplication to generate a response. In Round 3, to reduce the cost of inverse/exponents, we suppose that the participant P_i evaluates $q - b_i s$ then gets the value $K_{ij} = [h_j \times [(g^{b_j})^{b_i s}]^{-1}]^{b_i} = (h_j \times (g^{b_j})^{q - b_i s})\,)^{b_i}$. So each key share component costs two exponentiations, one multiplication, and one addition. Regarding communication, TTP-verified (2,3)-PATS starts with $n = 3$ broadcast messages coming from TTP to parties. In Round 1, each party sends a broadcast message to transfer the values $g^{a_i} \bmod p$ and $g^{b_i} \bmod p$. After Round 1, every party should verify two Schnorr ZKPs, which requires one commitment, one response from the prover, and one challenge from the verifier. In Round 2, two broadcast messages will be sent per party to transfer the values h_i. To compute the zero-knowledge proof of the exponent $b_i s$, the prover must send two messages, and the verifier must communicate with one message. In Round 3, two broadcast messages will be per party to transfer the values K_i. For the distributed (2,3)-PATS, the number of messages will be equal to the total number of messages in TTP-verified (2,3)-PATS since there are three broadcast messages in Round 0 of the TTP-verified version. However, in Round 3, it does not require broadcast. Instead, all parties send their share only to TTP. A breakdown of the cost and number of broadcast messages of each round is summarized in Table 1.

6.1.2. Run-Time Analysis. Implementing the proposed protocol in finite fields uses p values of different lengths and 256-bit q values. As seen in Figure 6, the length of p affects the run-time of the protocol significantly. The run-time is close to 0.02 s when p is of the length 512 bits, while it reaches close to 0.33 s for a random p value of length 3,072 bits.

6.2. PATS Using Elliptic Curves

6.2.1. Computational Load. A party P_i needs to do the following computations in a round of PATS protocol using elliptic curves:

(1) $a_i G$
(2) $b_i G$
(3) $s b_i$
(4) $(\sum_{j=1}^{3} a_j + \sum_{j \neq i} b_j) G = a_1 G + a_2 G + a_3 G + b_j G + b_k G$
(5) $H_i = s b_i (\sum_{j=1}^{3} a_j + \sum_{j \neq i} b_j) G$

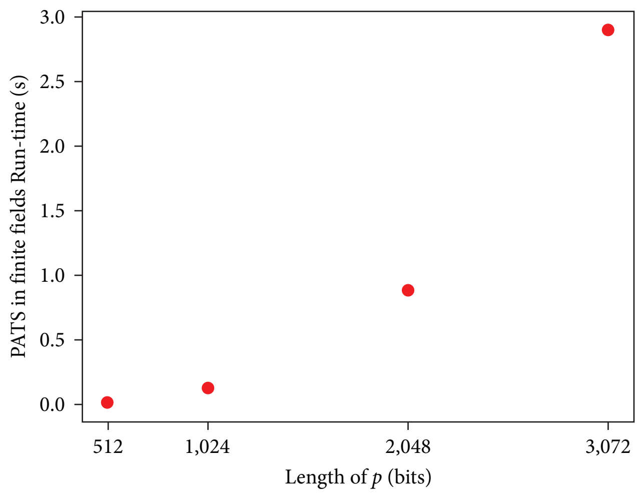

Figure 6: Run-time of (2,3)-PATS in finite fields. The results of 100 experiments within the 95% confidence interval are shown in the figure.

(6) $K_i = (K_{ij}, K_{it})$ where $K_{ij} = b_i(H_j - b_j b_i sG)$

In Round 1, it takes two elliptic curve scalar multiplications to compute a_iG and b_iG, and two additional elliptic curve scalar multiplications will be necessary to compute commitments for the zero-knowledge proofs of the multiples a_i and b_i if Schnorr non-interactive ZKP is applied. To prove that $Q = [a]P$ where Q and P are two points and a is a scalar, the prover must perform one scalar multiplication as a commitment. After Round 1, every party should verify two Schnorr ZKPS. Each ZKP takes two scalar multiplications for the verifier to verify and one multiplication and one addition for the prover to generate a response. In Round 2, there will be four point additions of previous calculations and broadcast messages, one multiplication to compute the power sb_i and one scalar multiplication to compute H_i value. To compute the zero-knowledge proof of the exponent b_is, an additional scalar multiplication will be necessary to compute a commitment for the zero-knowledge proof. To verify this Schnorr ZKP, each verifier computes one scalar multiplication and one point addition, and the prover computes one multiplication to generate a response. Round 3 requires three scalar multiplications (one of them is multiplying by -1) and one point addition for each K_{ij} value.

Regarding communication, TTP-verified (2,3)-PATS starts with $n = 3$ broadcast messages from TTP to parties. In Round 1, each party sends a broadcast message to transfer the values a_i and b_i. After Round 1, every party should verify two Schnorr ZKPS, which requires one commitment and one response from the prover and one challenge from the verifier. In Round 2, two broadcast messages will be sent per party to transfer the values h_i. To compute the zero-knowledge proof of the exponent b_is, the prover must send two messages, and the verifier must communicate with one message. In Round 3, two broadcast messages will be per party to transfer the values K_i. For the distributed (2,3)-PATS, the number of messages will be equal to the total number of messages in the TTP-verified (2,3)-PATS since there are three broadcast messages in Round 0 of the TTP-verified version, but in Round 3, it does not require broadcast. Instead, all parties send their share only to TTP. A breakdown of each round's cost and the number of broadcast messages is summarized in Table 2.

6.2.2. Run-Time Analysis. The implementation of the proposed protocol uses different curves, including P-192, P-224, P-256, P-384, and P-521, which are recommended in the FIPS PUB 186-4 digital signature standard report [40]. The length of the prime field p changes with each selected curve, and the run-time of the protocol is affected; however, the run-time for the P-521 curve is still less than half of a second, as seen in Figure 7.

6.3. (t,n)-PATS

6.3.1. Computational Load. A party P_i needs to do the following computations in $(n-1, n)$-PATS protocol in finite fields:

(1) g^{a_i}
(2) g^{b_i}
(3) b_is
(4) $g^{\sum_{j=1}^{n} a_j + \sum_{j\neq i} b_j}$
(5) $h_i = \left(g^{\sum_{j=1}^{n} a_j + \sum_{j\neq i} b_j}\right)^{b_is}$
(6) $K_i = (K_{ij})_{j\neq i}$ where $K_{ij} = \left(h_j \times \left(\left(g^{b_j}\right)^{b_is}\right)^{-1}\right)^{b_i}$

In Round 1, it takes two exponentiation to compute $g^{a_i} \bmod p$ and $g^{b_i} \bmod p$ and two additional exponentiation will be necessary to compute commitments for the zero-knowledge proofs of the multiples a_i and b_i if Schnorr non-interactive ZKP will be applied. To prove that $g^x = y$, the prover must perform one exponentiation as a commitment. After Round 1, every party should verify two Schnorr ZKPs. For each ZKP, it takes one exponentiation and one multiplication for the verifier to verify and one multiplication and one addition for the prover to generate a response.

In Round 2, the party multiplies $2n-1$, and many group elements are obtained and gathered from previous calculations and broadcasted messages. This requires $2n-2$ group multiplication. A scalar multiplication to compute the power b_is and an exponentiation to compute h_i value evaluated. To calculate the zero-knowledge proof of the exponent b_is, additional exponentiation will be necessary to compute a commitment for the zero-knowledge proofs of the multiple b_is. To verify this Schnorr ZKP, each verifier computes one exponentiation and one multiplication, and the prover computes one multiplication to generate a response.

In Round 3, the key share sequence of length $n-1$ is evaluated. As in the $(2,3)$-PATS the exponent $-b_is$ value is obtained by one addition. To be explicit, P_i evaluates $q - b_is$. This value will be used for each component in this step and any further iterative steps. After obtaining the $q - b_is$ value, each component K_{ij} of the key share takes two exponentiation and one multiplication operation.

When the first three rounds end, $(n-1, n)$-PATS is completed. For $t = n - r$, the corresponding (t, n)-PATS will be calculated iteratively in r-steps. Here, $(n-1, n)$-PATS is the initial step. Each iteration step consists of two rounds. Let us discuss the computational cost of the kth step. In Round 1,

Table 2: Analysis of computational cost of PATS using elliptic curves.

	Cost breakdown	No. of operations per party	No. of broadcast messages
0	Round 0	—	3*
1	Round 1	2 SM, 2 (ZKP)	6 + 2 [3]
2	Round 2	1 SM, 4 PA, 1 M, (ZKP)	6 + [3]
3	Round 3	3 SM, 1 PA	6 − 3*
4	Total	6 SM, 5 PA, 1 M, 3 (ZKP)	18 + 3 [3]

[1]SM, elliptic curve scalar multiplication. [2]PA, elliptic curve point addition. [3]M, multiplication [4]A, addition. [5](ZKP) = (3 SM, 1 PA, 1 M, 1 A), # of operations required for a single ZKP. [6][x], # of messages required for a single ZKP. *TTP-verified (2,3)-PATS requires three broadcast messages from TTP in Round 0, but three fewer messages in Round 3.

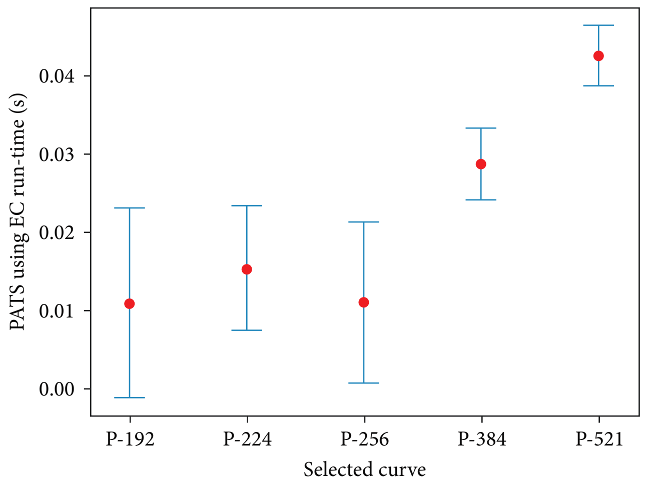

Figure 7: Run-time of PATS using ECC. The results of 100 experiments within the 95% confidence interval are shown in the figure.

each P_i calculates the exponent set $\{(g^{b_{j_1}\cdots b_{j_{k-1}}})^{b_i}\}$ where indices j_r ranges over I and distinct. This requires $\binom{n-1}{k-1}$ exponentiation since the base of the exponentiation was already calculated in the previous step. After the first round of the kth iteration step, the set $\{(g^{b_{j_1}\cdots b_{j_{k-1}}})^{b_i}\}_{j_r\in I}$ will be broadcasted with the keyshare sequence K_i obtained in $(k-1)$th step. In the second round of the kth iteration step, P_i computes the new $K_i = \{K_{ij_1\ldots j_k}\}$ sequence for distinct $j_r \in I$, where $K_{ij_1\ldots j_k} = [K_{j_1\ldots j_k} \times [(g^{b_{j_1}\cdots b_{j_k}})^{b_i s}]^{-1}]^{b_i}$. Recall that the value $q - b_i s$ was already obtained in the initial step. Therefore, each component requires one multiplication and two exponentiation. As the length of new key share sequence is $\binom{n-1}{k}$, computational cost of the second round is $\binom{n-1}{k}(1M, 2E)$.

Regarding communication, TTP-verified (t,n)-PATS starts with n broadcast messages coming from TTP to parties. In Round 1, each party sends a broadcast message to transfer the values $g^{a_i} \bmod p$ and $g^{b_i} \bmod p$. After Round 1, every party should verify two Schnorr ZKPs, which requires one commitment, one response from the prover, and one challenge from the verifier. In Round 2, two broadcast messages will be sent per party to transfer the values h_i. To compute the zero-knowledge proof of the exponent $b_i s$, the prover must send two messages, and the verifier must communicate with one message. In Round 3, two broadcast messages will be per party to transfer the values K_i. For the repeated steps of (t,n)-PATS, $2(n-t-1)$ broadcast messages will be sent from each party to deliver the sequence $K_i = (K_{ij})_{j\neq i}$ and $g^{b_j b_i} \bmod p$. For the Distributed (t,n)-PATS, the number of messages will be equal to the total number of messages in TTP-verified (t,n)-PATS since there are n broadcast messages in Round 0 of the TTP-verified version. However, in Round 3, it does not require broadcast. Instead, all parties send their share only to TTP. A breakdown of the cost and number of broadcast messages of each round is summarized in Table 3.

6.4. Limitations. Considering the efficiency requirement for (t,n)-threshold scheme, in cases where t is less than $n-1$, we think the path followed in PATS will be cumbersome. We decided to picture the scheme for distributed variants over finite fields. The generalized version can be phrased straightforwardly on elliptic curves, mutatis mutandis.

6.5. Comparison. Table 4, compiled from the report NIST 800-57 Recommendation for Key Management [41], provides a security strength level for the approved algorithms of finite-field cryptography and elliptic-curve cryptography. Column 2 states the average run-time of PATS in finite fields/PATS using elliptic curves, respectively. Column 3 states the minimum size of parameters in standardized finite-field cryptography. Column 4 gives the range of f for algorithms based on elliptic-curve cryptography. Using curve P-521 provides 256-bit security strength and a very efficient protocol for PATS. Note that protocols with a security strength of less than 112 bits are no longer approved, and 192-bit and 256-bit key strengths for FFC algorithms are not considered by NIST due to interoperability and efficiency problems [41].

7. Future Work

PATS has to be retrofitted with classical security tools to be applicable in practice. This includes the integration of nonces or timestamps to mitigate replay attacks and using message authentication codes to safeguard against integrity attacks. We left them as future work since they are classical approaches. As another future work, we plan to develop techniques for reducing the messaging load in (t,n)-PATS.

Table 3: Analysis of computational cost of (t,n)-PATS in finite fields.

	Cost breakdown	No. of operations per party	No. of broadcast messages
0	Initial step round 0	—	n^*
1	Initial step round 1	2 E, 2 (ZKP)	$n(n-1)+2[3]$
2	Initial step round 2	1 E, $(2n-1)$ M, (ZKP)	$n(n-1)+[3]$
3	Initial step round 3	1 A, $(n-1)$ (2E, M)	$n(n-1)-n^*$
4	Initial step total	1 A, $(2n+1)$ E, $(3n–2)$ M, 3 (ZKP)	$3n(n-1)$
5	kth iteration step round 1	$\binom{n-1}{k-1}$E	—
6	kth iteration step round 2	$\binom{n-1}{k}$(2E, M)	$2(n-t-1)$
7	Total	1 A, $(1+2n+\sum_{k=2}^{(n-t)}\left[\binom{n}{k}+\binom{n-1}{k}\right]$)E, $(3n-2+\sum_{k=2}^{(n-t)}\left(\binom{n-1}{k}\right))$M	$3n(n-1)+2(n-t-1)+3[3]$

[1]E, exponentiation in $\mathbb{Z}_q$. [2]M, multiplication in $\mathbb{Z}_q$. [3]A, addition in $\mathbb{Z}_q$. [4](ZKP) = (3 E, 2 M, 1A), # of operations required for a single ZKP. [5]$[x]$, # of messages required for a single ZKP. *TTP-verified (t,n)-PATS requires n broadcast messages from TTP in Round 0, and n fewer messages in Round 3.

Table 4: Comparable security strengths and run-time of finite-field cryptography and curves.

Security strength	Run-time	FFC (DSA, DH, MQV)	ECC (ECDSA, EdDSA, DH, MQV)
≤ 80	0.134/0.011	$L=1{,}024$ $N=160$	$f=160$–223 (P-192)
112	0.888/0.015	$L=2{,}048$ $N=224$	$f=224$–255 (P-224)
128	2.899/0.011	$L=3{,}072$ $N=256$	$f=256$–383 (P-256)
192	-/0.029	$L=7{,}680$ $N=384$	$f=384$–511 (P-384)
256	-/0.043	$L=15{,}360$ $N=512$	$f=512+$ (P-521)

8. Conclusion

In this paper, we proposed a password-authenticated (2, 3)-threshold group key-sharing mechanism called PATS. We first mentioned how it differs from threshold secret key-sharing schemes with its structure. We then showed two approaches to PATS: distributed (2,3)-PATS and TTP-verified (2,3)-PATS. We also noted that both approaches could have two different variants implemented using finite fields or elliptic curves. We generalized PATS to (t,n)-threshold scheme. We gave security requirements and presented how PATS satisfies them. The performance analysis of PATS is explained in detail. Finally, we reviewed related work in this area by elaborating on PGKE protocols and threshold secret sharing.

Data Availability

The source code for the finite field and elliptic curve implementations can be found at https://wins.ceng.metu.edu.tr:8085/gitlab/adnan/pats.

Disclosure

This work is a part of the PhD thesis of Adnan Kılıç at Middle East Technical University.

Conflicts of Interest

The authors declare that they have no conflicts of interest.

Acknowledgments

Dr. Ertan Onur and Dr. Betin Onur are supported by TÜBİTAK BİDEB 2219 grant. Adnan Kılıç is partially supported by the ASELSAN Tohum grant.

References

[1] G. Yao, H. Wang, and D. Feng, "A group pake protocol using different passwords," in *2009 International Conference on Networks Security, Wireless Communications and Trusted Computing*, vol. 1, pp. 270–273, IEEE, Wuhan, China, 2009.

[2] X. Liu, J. Liu, and G. Chang, "nPAKE: an improved group PAKE protocol," in *2012 Ninth Web Information Systems and Applications Conference*, pp. 115–118, IEEE, Haikou, China, 2012.

[3] M. Abdalla, O. Chevassut, P.-A. Fouque, and D. Pointcheval, "A simple threshold authenticated key exchange from short secrets," in *Advances in Cryptology-ASIACRYPT 2005*, vol. 3788 of *Lecture Notes in Computer Science*, pp. 566–584, Springer, Berlin, Heidelberg, 2005.

[4] J. Camenisch, A. Lehmann, A. Lysyanskaya, and G. Neven, "Memento: how to reconstruct your secrets from a single password in a hostile environment," in *Advances in Cryptology–CRYPTO 2014*, vol. 8617 of *Lecture Notes in Computer Science*, pp. 256–275, Springer, Santa Barbara, CA, USA, 2014.

[5] F. Hao, X. Yi, L. Chen, and S. F. Shahandashti, "The fairy-ring dance: password authenticated key exchange in a group," in *Proceedings of the 1st ACM Workshop on IoT Privacy, Trust, and Security*, pp. 27–34, Association for Computing Machinery, New York, NY, USA, 2015.

[6] Q. Dai, X. Zhao, Q. Xu, and H. Jiang, "A new cross-realm group password-based authenticated key exchange protocol," in *2011*

Seventh International Conference on Computational Intelligence and Security, pp. 856–860, IEEE, Sanya, China, 2011.

[7] L. Zhu, C. Guo, Z. Zhang, W. Fu, and R. Xu, "A novel contributory cross-domain group password-based authenticated key exchange protocol with adaptive security," in *2017 IEEE Second International Conference on Data Science in Cyberspace (DSC)*, pp. 213–222, IEEE, Shenzhen, China, 2017.

[8] Y. Zhang, Y. Xiang, and X. Huang, "Password-authenticated group key exchange," *ACM Transactions on Internet Technology*, vol. 16, no. 4, pp. 1–20, 2016.

[9] F. Wei, N. Kumar, D. He, and S.-S. Yeo, "A general compiler for password-authenticated group key exchange protocol in the standard model," *Discrete Applied Mathematics*, vol. 241, pp. 78–86, 2018.

[10] N. Asokan and P. Ginzboorg, "Key agreement in ad hoc networks," *Computer Communications*, vol. 23, no. 17, pp. 1627–1637, 2000.

[11] E. Bresson, O. Chevassut, and D. Pointcheval, "Provably secure authenticated group Diffie-Hellman key exchange," *ACM Transactions on Information and System Security*, vol. 10, no. 3, Article ID 10, 2007.

[12] G. R. Blakley, "Safeguarding cryptographic keys," in *1979 International Workshop on Managing Requirements Knowledge (MARK)*, pp. 313–318, IEEE Computer Society, New York, NY, USA, 1979.

[13] A. Shamir, "How to share a secret," *Communications of the ACM*, vol. 22, no. 11, pp. 612-613, 1979.

[14] C. Asmuth and J. Bloom, "A modular approach to key safeguarding," *IEEE Transactions on Information Theory*, vol. 29, no. 2, pp. 208–210, 1983.

[15] B. Chor, S. Goldwasser, S. Micali, and B. Awerbuch, "Verifiable secret sharing and achieving simultaneity in the presence of faults," in *26th Annual Symposium on Foundations of Computer Science (sfcs 1985)*, pp. 383–395, IEEE, Portland, OR, USA, 1985.

[16] P. Feldman, "A practical scheme for non-interactive verifiable secret sharing," in *28th Annual Symposium on Foundations of Computer Science (sfcs 1987)*, pp. 427–438, IEEE, Los Angeles, CA, USA, 1987.

[17] J. C. Benaloh, "Secret sharing homomorphisms: keeping shares of a secret secret," in *Advances in Cryptology—CRYPTO' 86*, vol. 263 of *Lecture Notes in Computer Science*, pp. 251–260, Springer, Berlin, Heidelberg, 1986.

[18] Y. Zhang, C. Xu, H. Li, K. Yang, N. Cheng, and X. Shen, "PROTECT: efficient password-based threshold single-sign-on authentication for mobile users against perpetual leakage," *IEEE Transactions on Mobile Computing*, vol. 20, no. 6, pp. 2297–2312, 2021.

[19] J. Jiang, D. Wang, G. Zhang, and Z. Chen, "Quantum-resistant password-based threshold single-sign-on authentication with updatable server private key," in *Computer Security – ESORICS 2022*, vol. 13555 of *Lecture Notes in Computer Science*, pp. 295–316, Springer, Cham, 2022.

[20] A. Bagherzandi, S. Jarecki, N. Saxena, and Y. Lu, "Password-protected secret sharing," in *Proceedings of the 18th ACM conference on Computer and Communications Security*, pp. 433–444, ACM, 2011.

[21] J. Camenisch, A. Lysyanskaya, and G. Neven, "Practical yet universally composable two-server password-authenticated secret sharing," in *Proceedings of the 2012 ACM Conference on Computer and Communications Security*, pp. 525–536, ACM, 2012.

[22] Z. Zhang, X. Li, Y. Wang et al., "TAGKA: threshold authenticated group key agreement protocol against member disconnect for UANET," *IEEE Transactions on Vehicular Technology*, vol. 72, no. 11, pp. 14987–15001, 2023.

[23] M. Weir, S. Aggarwal, M. Collins, and H. Stern, "Testing metrics for password creation policies by attacking large sets of revealed passwords," in *Proceedings of the 17th ACM conference on Computer and communications security*, pp. 162–175, ACM, 2010.

[24] F. Hao and P. Ryan, "J-pake: authenticated key exchange without pki," in *Transactions on computational science XI*, pp. 192–206, Springer, 2010.

[25] F. Hao and P. C. van Oorschot, "Sok: password-authenticated key exchange–theory, practice, standardization and real-world lessons," Cryptology ePrint Archive, 2021.

[26] M. Abdalla, M. Barbosa, P. B. Rønne, P. Y. Ryan, and P. Šala, "Security characterization of j-pake and its variants," 2021, Cryptology ePrint Archive, Report 2021/824, https://ia.cr/2021/824.

[27] F. Hao, *J-PAKE: Password-Authenticated Key Exchange by Juggling*, RFC Editor, 2017.

[28] F. Hao, "Prudent practices in security standardization," *IEEE Communications Standards Magazine*, vol. 5, no. 3, pp. 40–47, 2021.

[29] D. Wang, H. Cheng, P. Wang, X. Huang, and G. Jian, "Zipf's law in passwords," *IEEE Transactions on Information Forensics and Security*, vol. 12, no. 11, pp. 2776–2791, 2017.

[30] D. Wang, Z. Zhang, P. Wang, J. Yan, and X. Huang, "Targeted online password guessing: an underestimated threat," in *Proceedings of the 2016 ACM SIGSAC Conference on Computer and Communications Security*, pp. 1242–1254, ACM, 2016.

[31] F. Hao, *Schnorr Non-Interactive Zero-Knowledge Proof*, RFC Editor, 2017.

[32] E. B. Barker, D. Johnson, and M. E. Smid, "Sp 800-56a. recommendation for pair-wise key establishment schemes using discrete logarithm cryptography (revised)," 2007.

[33] M. Toorani, "Cryptanalysis of two pake protocols for body area networks and smart environments," *International Journal of Network Security*, vol. 17, no. 5, pp. 629–636, 2015.

[34] M. Toorani, "Security analysis of j-pake," in *2014 IEEE Symposium on Computers and Communications (ISCC)*, pp. 1–6, IEEE, 2014.

[35] R. Lu and Z. Cao, "Simple three-party key exchange protocol," *Computers & Security*, vol. 26, no. 1, pp. 94–97, 2007.

[36] D. R. Stinson, *Cryptography: Theory and Practice*, Chapman and Hall/CRC, 2005.

[37] D. Boneh, "The decision diffie-hellman problem," in *International Algorithmic Number Theory Symposium*, pp. 48–63, Springer, 1998.

[38] A. Stiglic, "Safe prime," in *Encyclopedia of Cryptography and Security*, pp. 541–541, Springer US, Boston, MA, 2005.

[39] S. Iftene, "Secret sharing schemes with applications in security protocols," *Scientific Annals of Cuza University*, vol. 16, pp. 63–96, 2006.

[40] C. F. Kerry and P. D. Gallagher, *Digital Signature Standard (dss)*, FIPS PUB, 2013.

[41] E. Barker and Q. Dang, "Nist special publication 800-57 part 1, revision 4," NIST, Tech. Rep, vol. 16, 2016.