Problems with PAKE protocols

Lars Mueller *Technical University Munich*
Munich, Germany
lars.mueller@tum.de

Abstract

In the last years data breaches in websites have become fairly common. To protect their users from potential access to their data the providers developed different methods to secure users. One solution which is to date not often used is called Password Authentication Key Exchange (PAKE) protocols. It enables a way for two parties to communicate with each other over an insecure channel. This connection can be established by knowing the same password or by proofing the other party that oneself knows the password. However like the other regularly used solutions PAKE protocol are attackable. In this paper we will introduce you how PAKE protocols work and then the basic methodology used to attack PAKE protocols. During that we will take a look at different PAKE protocols, there usage and how PAKE protocols developed over time. In today's world PAKE protocols provide an often unknown way to secure users from attackers which we would like to outline in this paper.

I. Introduction

To authenticate a user on a website the user must provide an identifier unique to himself, like a username or an email, as well as a password only known to himself. The requirements a password is needed to meet are often pretty difficult for the user to achieve. These requirements are needed to secure the user in case of a server-side data breach. The website saves the password of the user in their database. That database is leaked, and therefore it is possible to look up the password of the user.

There are ways to provide some form of security that the password can not directly look up. However, these measurements do not prevent any lookups with enough computation power. Because of high requirements there are so much different possibilities, so that it is physically impossible to look them up. Hence, passwords are difficult to remember, this is why users tend to user the same password on different services. Another threat for the user is that their password can be phised or guessed, especially if it is a pretty weak one. To protect a user in case of a

data breach and in some form of phishing service providers have introduced a second factor to login. These are often a pretty inconvenient way to login as the user is required to take extra steps to access the service.

It is possible to use an authentication protocol which provides enough security to be able to withstand such lookup attacks as well as phishing attacks. These protocols are called Password Authentication Key Agreement protocol (PAKE) and provide ways to withstand these attacks. They were introduced with the idea in mind to provide two parties with a secure way of communicating with each other over an insecure communication connection. The connection could be eavesdropped. This means they can authenticate and provide proof two each other that they are the person they tell the other to be, without ever sending a password to the other party.

While PAKE protocols exist and were developed other solutions performed better and became the standard. This means some services use PAKE protocols, but they are generally a niche. The majority of services uses other protocols and solutions to protect users from attackers. As we will explain later PAKE protocol have a unique approach to these problems while often providing more security in theory. This raises the question: ”Why PAKE protocols are not used as an industrial standard ?”. We will try to give a somewhat complete answer to this question on this paper.

To do that we will introduce you to the basics of cryptography and take a deep look into different PAKE protocols and how they can be classified. While we are at it, we will give you a brief digression onto the history of PAKE protocols. After that we will explain different attack vectors on different PAKE protocols. To do that we start at explaining some basic attacks on encrypted secrets as well as explaining attacks on the hardware the cryptographic computations are run on. These attacks are one of the hugest threats to cryptographic implementations. This part aims at finding out the secret of a user to access their encrypted communication. The part will be closed by a showing you an attack on a widely used PAKE protocol which combines the basic attack strategies with an attack on a deeper layer on the machine. In the next part we will take a look at vulnerabilities found in different PAKE protocols trying to impersonate another user without knowing the secret of said user. In the last part we will explain an attack on the new Wi-Fi security standard WPA3. WPA3 uses a PAKE protocol to authenticate a user unlike the previous iteration WPA2. To attack a PAKE protocol, or even to attack a service/protocol in general it is often needed to combine different attacks. The attack on WPA3 and another attack on another relatively popular PAKE protocol are a combination of different attacks. We will

show and explain both. To finish up we will evaluate the explained vulnerabilities and make a conclusion.

In summary, we will make the following contributions

- We provide an overview over the history of PAKE and their development over the last 20 years.
- As PAKE is crypto protocol we will introduce some basic concepts of crypto as well who are used by PAKE protocols.
- To give a reason why PAKE is not used in today's environments, we will take a close look at some attacks and vulnerabilities found at PAKE protocols.
- Then there will be a discussion as to why PAKE protocols are not used widely.
- In the last step we will combine the history and attacks and make a conclusion why PAKE protocols are not used until today and why that may change.

II. Related Work

PAKE protocols have a long history in computer science terms with a lot of research already being done. There are a lot of different approaches on the topic and different ideas to solve different problems. [1] The first appearance of Encrypted Key Exchange was 1992 in a paper which described a basic protocol secure against dictionary Attacks. [2] The first standardization of PAKE Protocols came with IEEE P1363.2. This project was formed because of huge interest in Industry and Science in the theme During this first so-called period of PAKE protocols these protocols where revised and reworked multiple times which extended the working period to 2008. However after the standardization was finished it did not lead to huge adoption in the industry as it was hoped. [1] In the second phase of PAKE development some services adopted the PAKE protocol such as Apple I-Cloud or Mozilla Firefox. [3] A lot of further developed PAKE protocols like the Juggeling-PAKE (J-PAKE) [4] or Secure Remote Password (SRP) [5] were developed. Advancements were made on old PAKE protocols. This lead to a new standardization process in 2018 [1] which is still going on which brought in new PAKE like OPAQUE [6]. In 2018 WPA3 the replacement for WPA2, the protocol to secure Wi-Fi networks, was announced by the Wi-Fi Alliance. [7] It includes an PAKE protocol called Dragonfly to authenticate with the Wi-Fi router. While a lot of devices suffer from such attacks, often it is not the theoretical approach than rather the practical implementation of such protocols.

III. Background

To explain the attacks and some of their corresponding protocols it is needed to define some variables which will be used throughout the paper. We have G which denotes subgroup of prime order Z_p^*. p is a high prime number which is considered secure against attacks. Let g be a generator of g. This means we can generate keys $g^{randomnumber}$. We will use Alice (A) and Bob (B) as two communicating parties

A. The basic security principles of PAKE

PAKE basically allow 2 parties to establish a secure channel in which they can communicate without the fear of a 3rd party to listen. Bellowing stated in 1992 the following requirements for Encrypted Key Exchange Protocols: [2]

- Resistance against dictionary Attacks:
  The communication between the two parties must not be decryptable. This means that there is no data obtainable which allows an attacker to find the private secret. Especially if the guesses on the secret are run offline by a dictionary or another password-decryption like attack.
- One Password guess per connection:
  When establishing a new connection between two parties only one guess on the secret is possible. It is not possible for an attacker to send for example 100 different, possible passwords in one connection attempt, the attacker needs 100 to try them out. Additionally, these attempts should be visible and blockable to prevent further guessing the secret.
- Forward secrecy:
  Alice and Bob have established a protected session with their pre-shared secret, they both a session-secret which allows them to communicate securely. Eve gets to know the pre-shared secret. She can establish a new session with either Alice or Bob and impersonate the other, however she does not know the session-secret and can therefore not listen to what Bob and Alice are doing in their already established session.
- Session-key security:
  Additionally to the session of Alice and Bob there is another session between Alice and Charlie. Eve is now able to obtain the session-secret from Alice-Bob, this means she can listen to their communication. The session between Alice and Charlie is still not compromised. This is the case for every other session.

A PAKE is a generally two stage protocol. [1] In these two stages both parties send 1 message each two the other party. Overall a PAKE flow usually consists of 4 messages. In the first phase a session-key is computed. The second phase is used to confirm the session-key with both parties. After that you have a key which can be used to encrypt your messages with a symmetric encryption using the session-key. The other member of the session can decrypt your message with the session-key.

B. Basics of encrypted Communication

Now we will take a look at basics of encryption. They are often used by PAKE protocols and other encryption protocols.

1) Safe Prime: A Safe Prime p is a prime number in the form $p = 2q + 1$ if q is a Sophie Germain prime. A prime s is a Sophie Prime if l is a prime as well $l = 2s + 1$. As every integer number is a factorization of primes, and it is costly to compute factorization of primes large primes are used to generate secret keys. These keys are then used in different encryption and decryption protocols.

2) Hashing: A hash function takes a key as input. The output is a fixed size hash code. These functions are used to map data to make it indexable That would be the case if the hash function was perfect which is physically not possible. Hash functions follow three principles to withstand different types of attacks [8]

- Pre-Image resistance

  It is difficult to find a corresponding message M to a given hash h, $h = hash(M)$. The function is a one-way function.
- Second Pre-Image resistance

  It is difficult to find another message $M2$ getting the same hash as the first message $M1$. $hash(M1) = hash(M2)$.
- Collision resistance

  Similar to second pre-image resistance, it should be difficult to find two messages $M1, M2$ that have the same hash. $hash(M1) = hash(M2)$.

3) Zero-Knowledge-Proof: The Zero-Knowledge-Proof describes a way to proof someone else that you know a secret without ever telling the person the secret. The verifying person knows the secret as well. The verifier can ask you different questions which are conducted from the secret,

which you can answer correct if you know the secret. This can be repeated until the verifier is convinced that you know the secret.

An Abstract example would be Alice is colorblind and Bob is not. Bob has a red and a green ball. They seem identical to Alice, so she is not sure if Bob is telling her the truth, and they are in fact different. She wants to proof Bob and holds one Ball in her left the other in her right hand. Bob knows in which hand they currently are. Alice decides, without Bob looking, if she wants to switch the balls or not after she has done that she asks bob if she switched or not. Bob answers, if both balls have the same color bob will eventually choose the wrong option. If they are differently colored Bob should be able to tell Alice if she switched or not.

Like a lot of concepts in cryptography the zero-knowledge-proof has some properties which define it. [9]

- Completeness
  if the proof is correct, the prover will convince the verifier that he is correct
- Soundness
  if the proof is wrong, the verifier will not be convinced by the prover, however there is a small probability for error.
- Zero-Knowledge
  There is no secret leaked by proofing.

A famous Zero-Knowledge-Proof would be the Schnoor-Signature. [10] Alice wants to proof Bob that she is Alice.

- Alice generates a private-key a and
  computes public key $A = g^a$ where a is a random number
- Alice computes $V = g^v$ where v is a random number
- Alice computes $r = v - x \cdot Hash(g, V, A, username)$
- Alice sends Bob $A, V, r, username$.
- Bob computes $h' = Hash(g, V, A, username)$.
- Bob computes $V' = g^r \cdot X^{h'}$

C. PAKE Handshake

1) Balanced PAKE: Diffie-Hellman Encrypted Key Exchange: A balanced PAKE protocol is a protocol which only can be used if both communicating parties know one pre-shared-key (PSK) or a password. The most famous Balanced PAKE is the Diffie-Hellman Encrypted Key

Exchange (DH-EKE). Even though it is not used most other balanced PAKEs are based on the Diffie-Hellman key Exchange. The Diffie-Hellman Key Exchange is an asymmetric key exchange. It consists of a public and private key per party. The public key is used to encrypt a message. [11] Then it can only be decrypted by the person who is in possession of the private key. In the following we will explain the DH-EKE. [12] It consists of a Diffie-Hellman Key exchange and a generation of a session-key which allows to encrypt the communication in a way that it is not readable if one private key is later leaked.

- Alice sends Bob $A_{encrypted} = Encrypt_{PSK}(g^a)$, where a is a random generated number.
- Bob decrypts $A_{encrypted}$ and computes session-key $K = A^b$, where b is a random generated number.
- Bob computes $B = g^b$ and a random challenge $challenge_{Bob}$.
- Bob sends Alice $B_{encrypted} = Encrypt_{PSK}(B, Enrypt_K(challenge_{Bob}))$.
- Alice decrypts $B_{encrypted}$ and computes session-key $K = B^a$
- Alice decrypts $challenge_{Bob}$ and computes $challenge_{Alice}$
- Alice sends Bob $Encrypt_K(challenge_{Alice}, challenge_{Bob})$
- Bob decrypts and checks $challenge_{Bob}$ against his previously generated one.
- Alice is authenticated to Bob if $challenge_{Bob}$ are the same.
- Bob sends Alice $Encrypt_K = (challenge_{Alice})$
- Alice decrypts and checks $challenge_{Alice}$ against her previously generated one.
- Bob is authenticated to Alice if $challenge_{Alice}$ are the same

There are other balanced PAKE protocols like SPEKE or J-PAKE. Both protocols are building on the DH-EKE. SPEKE is not using a public known generator he is using a generator derived from the password [13]. J-PAKE uses a Zero-Knowledge-Proof in the second phase of the protocol, usually the Schnoor-Signature. This requires much more computational power.

2) Augmented PAKE: SRP: The Secure Remote Desktop Protocol is an augmented PAKE protocol which means one side does not know the secret but only a way to verify somebody that he knows the secret. [14] It is used to perform login and registration processes. To perform a login process/proof of the password we assume Alice as client and Bob as the authentication

server.

- Alice sends Bob $A = g^a$ where a is a random generated value and g a generator of a multiplicative group
- Bob sends Alice $B = kv+g^b$ where k is a parameter both sides know, $k = Hash(saveprime, g)$
- Both parties compute $u = Hash(A, B)$
- Alice can compute the session-key: $K_{Alice} = Hash(B - kg^password)^(a + u \cdot password)$.
- Bob can compute the session-key as well $K_{Bob} = Hash((A \cdot verifier^u)^b)$
- Both parties share the same session-key, this needs proof.
- Alice sends Bob who verifies,
  $M_1 = Hash(Hash(N)XORHash(g)$
  $|Hash(username)|salt|A|B|K_{Alice})$
- Bob sends Alice who verifies,
  $M_2 = Hash(A|M_1|K_{Bob})$
- The calculation of K and its verification can be skipped.
- Alice sends Bob who verifies, $M_1 = Hash(A|B|B - kg^password)^(a + u \cdot password)$
- Alice sends Bob who verifies, $M_2 = Hash(A|M_1|(A \cdot verifier^u)^b)$

SRP is probably the most used protocol of the PAKE protocols. It is used by the mail provider ProtonMail and in the smart home solution of Apple, Apple HomeKit [1]. OPAQUE is another newly developed augmented PAKE, it uses an additional salt to secure the verification hash [6]. Another protocol which will probably used in the future a lot is the Dragonfly protocol used in the new Wi-Fi security standard WPA3 [7]. [14]

IV. Attacks on PAKE

Now we will list and explain different attacks on PAKE protocols.

A. Dictionary attack

A pretty basic attack which is pretty common in Public-Key-Infrastructure as well. [15] To perform a dictionary attack you need a dictionary with common passwords, it is possible to create dictionary based on information from the user. A lot of times dictionary are also generated from leaked passwords on the internet. Next you need an encrypted message or hash which you want to decrypt or find the corresponding message to. The last thing you need is the used encryption/hash function to cipher the message.

Now it is possible to try out as many passwords as you have in your dictionary, only limited by time and available resources.

Even if one security principle of PAKE is that it is protected against offline dictionary attacks some protocols are vulnerable to these attacks. An augmented PAKE like SRP, is not storing the password on the server. Only a verifier is saved, which is a one-way-function of the password hash. This means if the database is breached it would allow an attacker to start an offline dictionary attack on the verifier. This is pretty similar to today's mostly used Public Key Infrastructure, where only a password hash is stored on the server [16]. Nonetheless, this does not mean that the SRP protocol is completely secure against dictionary attacks, it is only secure against server data leaks with following bruteforce attacks [3].

B. Pre-computation attacks

Pre-Communication attacks are a form of dictionary attack, which is like the dictionary attack not limited to PAKE protocols. It works by pre-computing password hashes with a salt. After a data breach on the server-side or another way that leaks the password hash it is possible to look up the corresponding password instantly. Even though an augmented PAKE is not directly vulnerable to such attacks, it is sometimes still possible. An Example would be the Secure Remote Password (SRP) protocol. It is a rather famous protocol because it is used by some applications, therefore there are a lot of different implementations for it. This protocol does not store the password on the server only a verifier to verify the password [16]. But it is possible to pre-compute verifier for passwords if it is known that the target server is using the SRP protocol. [3]

C. Side-channel attack

This is a group of attacks, which try to find secrets by gathering information from the target. They do not aim at the design of the protocol, they target the implementation. This information can be of physical nature like power consumption, electromagnetic radiation. Measuring the time it needs to compute a cryptographic function and then relaying information on the secret is also a side-channel attack. Another one would be, watching the access of the cache used by the victim and getting the memory pattern with the ability to make a well-educated guess on the secret [17].

D. PARASITE

While we explained some theory based attacks, we will now focus on an attack on the OpenSSL implementation of SRP. This attack could lead to a password leak and is called PARASITE (PAssword Recovery Attack against Srp implementations in ThE wild) and is a cache based side channel attack. In this attack the target is the password of a victim which is saved on the memory. Especially the Flush+Reload and Performance Degradation Attack (PDA) is used. [3] It monitors the memory addresses and finds the used ones by the victim. This is achieved by flushing and reloading the cache and measuring the time needed for the victim to receive the data. If the memory is flushed at the point of access the data needs to be looked up from the memory directly which subsequently takes longer. In the OpenSSL implementation of the modular exponentiation, they use a special variable to speed up the square and multiplication process. However, the variable overflows, after that a new bigger variable with the Montgomery representation is used. The Montgomery representation is an elliptic curve used to speed up the process of modular exponentiation. Hence, this part oft the implementation allows an attacker to distinguish every iteration on it, as the first part of the implementation was too fast. This makes the amount of iterations guessable, and therefore you can guess some bit patterns [17].

The chosen generator is 5, because it is the most convenient one to used. With knowing the generator we can identify on which bits the accumulated variable is overflowing and how many iterations are needed until the overflow occurs. Through FLUSH+RELOAD attacks it is possible to deduct some bits of the password while others remain unknown however due to known position of said bits it is possible to reduce the remaining possibilities. If it is not possible to find the password with one run of the Protocol, it can be run more times, with another salt, to get different bit patterns of the same password. Because we know some parts of the password it is possible to start a dictionary attack on the password even a pre-computation attack as we know the used salt. With knowing the password we can impersonate both parties. We know the password to tell the server we are the user he expects as well as telling the user we are the server because we could calculate the verifier stored on the server. These type of attacks happen and are partly possible on different cryptographic protocols. Also, it is important to mention that the rather unique threat model of the attack: It needs access to the memory and the cache of the processor. This is possible through a spy program on the pc or a JavaScript injection in the browser. This threat model is often needed for side-channel attacks, and with such kind of

threat it is often easier to just log the keyboard and the password. The OpenSSL library is one of the most used crypto libraries and therefore used as reference in other implementations. This is also the case on other implementations of the SRP protocol which made them vulnerable as well [3].

E. Impersonation attack

The first EKE protocols as well as the newer SPEKE protocol is suffering from this attack. EKE and SPEKE are balanced PAKEs. Like the name suggests the attacker poses as a user in the impersonation attack to obtain private information. The impersonation attack is applicable if two users have multiple sessions in parallel with each other. Alice and Bob share a common password. Now Alice starts a session with Bob (Session 1) by sending him random selected number $g^x modp$ (p is a safe prime number). This randomly selected number generated by a generator provided a function which takes the shared password as input. After the stage Alice and Bob get their session keys k which is generated from $g^{ab} modp$ For the key variation Alice sends her first key confirmation challenge H(H(k)), where h is a hash function. The prime number g^x and the first key confirmation challenge are intercepted by Eve. Eve raises the prime number by the power of z, a random selected number. Eve now initiates another session (Session 2) with Alice using g^{xz}. Alice replies with another random prime number g^y. Eve does the same as before and raises this number by the power of z to g^{yz} and sends it back to Alice. In the next step Eve sends the intercepted key confirmation challenge to Alice which will be answered by Alice with $H(k)$. Now Eve has intercepted Session 1 while owning Session 2 completely which opens up from different scenarios [13].

F. Replay-Attack

This is an attack on the J-PAKE protocol, which was used in the Mozilla Firefox synchronization feature. It is related to the impersonation attack, because it replays eavesdropped messages and impersonate the other party. This attack is limited by the fact that the session key cannot be computed from the eavesdropped messages without the password of a session member. It means, that attackers are only authenticated, but they cannot encrypt or decrypt messages from the other participant. Basically this is a variation of the impersonation attack on augmented PAKEs J-PAKE consists of 4 messages which can be denoted into two independent parts. The Authentication Part is independent of the key exchange part. [4] The second part of J-PAKE

which allows to run a replay-attack on it is that the function to generate a session-key does not include some session-specific information. [5] We assume an Attacker Eve intercepted an earlier run of the protocol, and he wants to impersonate Alice. E sends a derived message to B. B proceeds to start his part of the J-PAKE by selecting random numbers and computing Zero-Knowledge-Proofs to them. E answers with the next eavesdropped and derived message. B now verifies the numbers of E with his own and generates a session key k, using values from the first message and his own random generations. Bob thinks Eve is Alice, but Eve cannot compute the session key because she does not have the password of Alice and x_2. x_2 is the second random generated value of Alice which is never sent to Bob because it was only used as exponent of a save prime number. [5]

G. Dragonblood

Is a series of vulnerabilities on the balanced PAKE Dragonfly. This PAKE is used in WPA3 as Handshake [18]. To secure a wireless network from eavesdropper it is necessary to encrypt the connection between the two communicating devices. To achieve that, it is needed to complete multiple steps, which basically ensure that both devices have the same encryption key and a potential eavesdropper does not. In the new WPA3 standard, there was new protocol introduced called Dragonfly Handshake. Dragonfly is a two phase balanced PAKE protocol and provides the on PAKE proposed security. The first phase is called Password Derivation, and it is used to generate a hash from a pre shared password/key (PSK). This hash function uses an elliptic curve hash to curve function. The hash is calculated when a user tries to connect to a WPA3 secured network. This method usually allows for timing attacks like the PARASITE attack, however in this case there is an additional loop called which makes measuring the time needed for one iteration impossible. This defense measure is not used in the authentication standard EAP-pwd. WPA2-Enterprise was the solution in the old standard. It is used to authenticate a user and map their device to a specific user. The second phase is called Confirm and Commit Phase. This phase is a two phase PAKE in itself similar to the already explained DH-EKE. There are several possible attacks on WPA3:

- Downgrade attack: it is possible to connect to a WPA3 network using the old WPA2 Handshake. While the standard provides security against the old attacks and this attack gets detected it is possible to catch enough data to start a dictionary attack on the password.

- Costly Defense Measurements: Dragonfly has implemented a lot of different defensive measurements against side-channel attacks, however they are pretty resource intensive. This could lead to small devices not implementing these measurements and therefore be Vulnerable against side-channel attacks
- Timing Attacks: The hash-to-curve methods have timing leaks, if the algorithm uses the Brainpool Curves. These attacks are performed by sending a clear signal to the WPA3 Access Point which results in deleting all data corresponding to that connection. The point of time measurement is the response times which get seemingly longer the more iterations are needed for the hash-to-curve function. The included defensive measurements against timing attacks have proven to be not sufficient.
- Cache Based Attacks: These attacks also aim at the hash-to-curve functions. This attack is similar to the PARASITE attack, as it uses the Flush+Reload Pattern and the resulting bit pattern to get a secret.
- Brute-force Attack: The previous attacks only got us parts of the password needed to connect with the Wi-Fi. To get a password all possible solutions are computed and run them against the Wi-Fi.

[7]

V. Evaluation & Discussion

Despite their long history PAKE protocols have not yet made the jump to be a well known solution to the common problem of authenticating a user to a server. The current standard is involving a TLS connection which prevents man-in-the-middle attacks, as well as saving the password of the user on the server. In the best case password is saved on the server as a salted hash. Worst case would be that it is saved or even logged somewhere in clear text. This flaw can also be found in a lot of balanced PAKEs, because they share the same password with each other. With Balanced PAKEs having the problem to rely on saved passwords on server side, augmented PAKEs entered the display. Another important issue with PAKE not used widely, is that most PAKE protocols used to have a patent and that stopped them from being used at all. Nowadays, most patents are expired.

J-PAKE was the first protocol that was publicly available and therefore used in some applications [4]. Especially Mozilla tried to integrate it into their browsers sync feature, but it got removed shortly after. The same happened with an OpenSSL implementation of J-PAKE it got

removed. J-PAKE was a balanced PAKE, it did not provide any advantage over the established methods. These advantages are While the PAKE protocols where under the protection of patents other solutions where developed. The most famous and used other solutions are SSL and their, now standard, successor TLS. Implementing these protocols in today's services would not change much for the user as he is still required to choose a password. One important factor however is the strength of the used password as the most actual versions (OPAQUE) [6] of PAKE protocols provide enough protection from bruteforce attacks in case of a server breach. This is a good thing as users tend to use simple passwords for their logins which provides a huge security risk for them. So implementing these protocols would not change anything for the user, but it will help strengthen the security of the user of a potential password leak. With developing these protocols the next problem occurs: There could be attacks found on the implementation, like the PARASITE attack on the SRP protocol. On the other hand today's standard protocols are not immune to attacks either, like the Heatbleed attack on the TLS protocol showed in 2014 [19]. Implementing a PAKE in your login flow has the side effect of being resource intensive on the user device. It is easier to compute a salted hash of a password on the device than to compute several hashes combined with multiple additional exponential modular calculations.

Some PAKE protocols already found their way in the most used crypto library called OpenSSL. When optimized it should be possible that today's used devices are able to handle the additional workload provided by using a PAKE protocol. These protocols make a pretty strong case for them nowadays especially combined with the already established TLS protocol they are a good and more secure alternative to established methods of authentication a user. That PAKE protocols can be used in wireless networks is also showed in by Apple by using the SRP Protocol in their smart home solution [3].

VI. Conclusion

With WPA3 there is a PAKE underway to come to every Wi-Fi network, while it still has some flaws it is nowadays a better solution than WPA2. With a new generation of PAKE there is a possibility that more services and devices will adapt these protocols and be therefore more secure. The past showed that there are a lot of possible threats on PAKE protocols. These threats and vulnerabilities can be exploited as it was shown by the PARASITE and Dragonblood attacks. Our contribution was to get a short introduction to PAKE protocols and show their weakness and strengths. We also provided some history on why PAKE protocols are not adapted

by industry despite being a superior solution. We showed that PAKE protocols are not immune to attacks they say they should be, like the SPEKE or EKE protocol against dictionary attacks. As implementations of PAKE protocols becoming standardized in other protocols it is important for them to be secure. It was important for them to be performant which opened them up for side-channel attacks. While still important with today's availability of computing power it should be possible to implement PAKEs performant and secure. As a final thought, while still not standard PAKEs are on a good way to become more famous as the topic gets more traction. We wanted to contribute to that possible traction by giving the people some insight on why PAKEs are not widely used.

Being a neat idea but still having some child troubles PAKEs still have a long way to go until they are a standard. Today it is standard to have and encrypted connection with a website without a possible eavesdropper. However, there still is the possibility that a password could be leaked due to bad configuration of the website owner. Hence, it is important to strengthen the user even allow them to choose easier passwords as they can not guess easily with a PAKE protocol in use. A PAKE protocol also authenticates the website against the user meaning the user is safe that he is browsing the right website and not a phising site trying to grab his information.

REFERENCES

[1] F. Hao and P. C. van Oorschot, "Sok: Password-authenticated key exchange–theory, practice, standardization and real-world lessons," in *Proceedings of the 2022 ACM on Asia Conference on Computer and Communications Security*, 2022, pp. 697–711.

[2] S. M. Bellovin and M. Merritt, "Encrypted key exchange: Password-based protocols secure against dictionary attacks," *Proceedings: 1992 IEEE Computer Society Symposium on Research in Security and Privacy*, 1992.

[3] D. de Almeida Braga, P.-A. Fouque, and M. Sabt, "Parasite: Password recovery attack against srp implementations in the wild," in *Proceedings of the 2021 ACM SIGSAC Conference on Computer and Communications Security*, 2021, pp. 2497–2512.

[4] F. Hao and P. Y. Ryan, "Password authenticated key exchange by juggling," in *International Workshop on Security Protocols*. Springer, 2008, pp. 159–171.

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

[6] S. Jarecki, H. Krawczyk, and J. Xu, "Opaque: an asymmetric pake protocol secure against pre-computation attacks," in *Annual International Conference on the Theory and Applications of Cryptographic Techniques*. Springer, 2018, pp. 456–486.

[7] M. Vanhoef and E. Ronen, "Dragonblood: Analyzing the dragonfly handshake of wpa3 and eap-pwd," in *2020 IEEE Symposium on Security and Privacy (SP)*. IEEE, 2020, pp. 517–533.

[8] R. Sobti and G. Geetha, "Cryptographic hash functions: a review," *International Journal of Computer Science Issues (IJCSI)*, vol. 9, no. 2, p. 461, 2012.

[9] U. Feige, A. Fiat, and A. Shamir, "Zero-knowledge proofs of identity," *Journal of cryptology*, vol. 1, no. 2, pp. 77–94, 1988.

[10] G. Neven, N. P. Smart, and B. Warinschi, "Hash function requirements for schnorr signatures," *Journal of Mathematical Cryptology*, vol. 3, no. 1, pp. 69–87, 2009.

[11] S. Mitra, S. Das, and M. Kule, "Prevention of the man-in-the-middle attack on diffie–hellman key exchange algorithm: A review," in *Proceedings of International Conference on Frontiers in Computing and Systems*. Springer, 2021, pp. 625–635.

[12] C.-M. Chang, "Dh-eke: Diffie-hellman encrypted key exchange," access: 6.7.2022. [Online]. Available: https://chunminchang.gitbooks.io/j-pake-over-tls/content/pake/balanced/dh-eke.html

[13] F. Hao and S. F. Shahandashti, "The speke protocol revisited," in *International Conference on Research in Security Standardisation*. Springer, 2014, pp. 26–38.

[14] T. D. Wu *et al.*, "The secure remote password protocol." in *NDSS*, vol. 98. Citeseer, 1998, pp. 97–111.

[15] L. Bošnjak, J. Sreš, and B. Brumen, "Brute-force and dictionary attack on hashed real-world passwords," in *2018 41st international convention on information and communication technology, electronics and microelectronics (mipro)*. IEEE, 2018, pp. 1161–1166.

[16] A. T. Sherman, E. Lanus, M. Liskov, E. Zieglar, R. Chang, E. Golaszewski, R. Wnuk-Fink, C. J. Bonyadi, M. Yaksetig, and I. Blumenfeld, "Formal methods analysis of the secure remote password protocol," in *Logic, Language, and Security*. Springer, 2020, pp. 103–126.

[17] F.-X. Standaert, "Introduction to side-channel attacks," in *Secure integrated circuits and systems*. Springer, 2010, pp. 27–42.

[18] "Wpa3 specification - wi-fi alliance," Dec 2020, access: 6.7.2022. [Online]. Available: https://www.wi-fi.org/downloads-public/WPA3_Specification_v3.0.pdf

[19] M. Carvalho, J. DeMott, R. Ford, and D. A. Wheeler, "Heartbleed 101," *IEEE security & privacy*, vol. 12, no. 4, pp. 63–67, 2014.