zkPAKE: A Simple Augmented PAKE Protocol

Karina Mochetti[1] , **Amanda C. Davi Resende**[1] , **Diego F. Aranha**[1]*

[1] Institute of Computing (UNICAMP)
Av. Albert Einstein, 1251, 13083-852, Campinas-SP, Brazil
{mochetti,dfaranha}@ic.unicamp.br,amanda@lasca.ic.unicamp.br

Abstract. *Human memory is notoriously unreliable in memorizing long secrets, such as large cryptographic keys. Password-based Authenticated Key Exchange (PAKE) protocols securely establish a cryptographic key based only on the knowledge of a much shorter password. In this work, an* augmented *PAKE protocol is designed and proposed for secure banking applications, requiring the server to store only the image of the password under a one-way function. The protocol is more efficient than alternatives because it requires fewer public key operations or a lower communication overhead.*

1. Introduction

Cryptographic keys for encryption and signature schemes must be generated randomly and can have from a few hundred bits to many thousand bits. Since human memory can hardly memorize such amount of unstructured data, keys are often stored in external devices. However, this is not always possible and a secure communication key must be established using a smaller and simpler password, that the user is able to remember.

A Password-Authenticated Key Exchange (PAKE) protocol is a method for establishing secure cryptographic keys based only on the knowledge of a simple password, short enough to be easily memorized by humans [Boyd and Mathuria 2003]. In most PAKE protocols, the server and the client share only the knowledge of the password in some form and use it to negotiate a shared key in an authenticated way.

The first PAKE protocol [Lomas et al. 1989] was developed under the additional assumption that the client has knowledge of the server public key, alongside the shared password. Other protocols have been developed over the years, but the main limitation in practice nowadays is that the more mature protocols are patented.

In this work, we reassure the importance of PAKE protocols in secure banking applications, with emphasis to augmented PAKE protocols, and propose an augmented PAKE protocol constructed from zero-knowledge proofs.

2. Background and Related Work

The main goal of a PAKE protocol is to establish a cryptographic key between a client and a server, based only on their knowledge of a password, without relying on a Public Key Infrastructure (PKI), which is complex and subject to man-in-the-middle attacks. The most efficient and commonly used PAKE protocols are EKE [Bellovin and Merritt 1992] and SPEKE [Jablon 1996], constructed from the basic Diffie-Hellman protocol. The main difference in their construction is that the SPEKE protocol uses the password as the group generator, while the EKE protocol uses it only as an auxiliary encryption key.

*Supported by Intel in the scope of the project "*Physical Unclonable Functions for SoC Devices*".

A secure PAKE protocol must fulfill four basic security requirements [Hao and Ryan 2010]: it must be resistant to both offline and online dictionary attacks, provide forward security and known-session security. Dictionary attacks consist of an exhaustive search of the password based on a list of words which are guessed as most likely to succeed. An online attack tries several inputs against a legitimate protocol, while an offline attack attempts to emulate the protocol using several known outputs. Therefore, a PAKE protocol implementation cannot leak any information that allows an attacker to learn the password through an exhaustive search.

A protocol is forward secure if it ensures that session keys remain secure even if the password is disclosed. This property implies that if an attacker knows the password but only passively observes the key exchange, he cannot derive the session key. Finally, in a known-session secure protocol, a compromised session should not harm the security of any other sessions, i.e., an attacker may have all information specific to the session, but this must not affect the security of other established sessions.

An extra security requirement can be resistance against server compromise. To accomplish this, the protocol must assure that an attacker cannot impersonate a user even if the credential files are stolen. PAKE protocols with this feature are called *augmented PAKEs* [Perlman and Kaufman 1999], as opposed to *balanced PAKEs* [Jablon 1996].

In an augmented PAKE the server does not know or store a plaintext password, but an image of the client's password under a one-way function. Augmented PAKE protocols are usually more complex and computationally expensive than balanced PAKEs. For some applications, this feature is not useful and the additional complexity and computational costs are not worthy. Such applications use secure balanced PAKE protocols, such as EKE and SPEKE, but without resistance against server compromise. For other applications, such as secure banking though, resisting server compromises can be critical, even with some performance penalty.

Secure banking typically employs cryptographic protocols to provide secure communication between two parties, such as Transport Layer Security (TLS) and Secure Sockets Layer (SSL). Although popular, these protocols are subject to man-in-the-middle attacks [Anderson 2001] and are sensible to user flaws; most users click through certificate warnings and ignore browser security indicators [Engler et al. 2009].

In this scenario, the client already knows a small and simple password to be able to perform transactions in the server maintained by the bank. If this password is stored as plaintext in the server, any malicious employee may successfully impersonate the client in a balanced PAKE protocol. Therefore, for this kind of application, resistance against server compromise is important, preventing an insider from impersonating the client. Note that not all bank employees may have control over clients accounts to perform transactions, specially the ones involved in maintaining the computer infrastructure.

To solve this problem we design an augmented PAKE. In this case, the user will have to register his/her password with the bank upon opening an account. This will be performed in the enrollment phase, in which the bank will receive and store an image of the password. Now, a malicious employee does not have knowledge of the plaintext password and cannot impersonate the user on the authentication phase of a PAKE protocol.

3. Our Protocol

In this Section we describe our contribution, the zkPAKE protocol, presented in Figure 1. zkPAKE is an augmented PAKE protocol, based on zero-knowledge proof of knowledge (ZKPK), a feature shared with some authentication protocols.

Server		**Client**
Shared Information: g: generator of group $\mathbb{G}$, g^r		
$n \leftarrow\$\, \mathbb{Z}_q, N = g^n$	$\xrightarrow{\quad N \quad}$	$r = H_1(pwd)$
		$v \leftarrow\$\, \mathbb{Z}_q$
		$t = N^v$
		$c = H_1(g, g^r, t, N)$
		$u = v - H_1(c)r \bmod q$
$t' = g^{un} g^{rnH_1(c)}$	$\xleftarrow{\quad u, H_1(c) \quad}$	$\mathsf{sk}_c = H_2(c)$
$c' = H_1(g, g^r, t', N)$		
$H_1(c') \stackrel{?}{=} H_1(c)$		
$\mathsf{sk}_s = H_2(c')$	$\xrightarrow{\quad H_1(\mathsf{sk}_s) \quad}$	$H_1(\mathsf{sk}_s) \stackrel{?}{=} H_1(\mathsf{sk}_c)$

Figure 1. zkPAKE Protocol.

An enrollment phase must be held before the main zkPAKE protocol execution. This phase is performed in a physically secure way between the client and the server, such as an user registering his/her password in person within the bank. A shared secret is then generated based on the password pwd and the generator g of group $\mathbb{G}$. The client computes the secret g^r, with r being a hash of the password pwd and sends it privately to the server. Note that, instead of storing and using the password directly, the server will use the image of the password in the authentication phase, satisfying the augmented PAKE definition. The enrollment phase needs to be executed only once for each client.

The next phase consists in the authentication steps of the basic PAKE protocol. The server begins the transmission sending a nonce N. The client is able to calculate g^r and generate a secret key $\mathsf{sk}_c = H_2(c)$ using a technique similar to a protocol for zero-knowledge proof of possession [Chaum et al. 1987]. After u and $H_1(c)$ are returned, the server can generate and prove knowledge of a secret key $\mathsf{sk}_s = H_2(c')$. Note that in our construction the authentication is done by both sides, thus the protocol inherently provides mutual authentication.

4. Results

Table 1 presents a performance analysis of our protocol, comparing it with the main PAKE protocols proposed in the literature. The number of exponentiations on client or server side are computed for each protocol, considering the usual optimizations for implementing exponentiations depending on the base. For simplicity, symmetric encryption, hash function and other cheap operations are not taken into account. Powering an unknown basis has a unitary cost (1.0), while fixed-base exponentiation costs half as much (0.5). Double exponentiation can be implemented by interleaving to save squarings, costing a unity and half (1.5). All protocols can be instantiated using elliptic curve groups, enjoying these optimizations [Hankerson et al. 2003]. The computation and communication savings of zkPAKE compared to alternatives become clear.

Protocol	Type	Exp (Client)	Exp (Server)	Exp (Total)	Messages
EKE	balanced	1.5	1.5	3	4
SPEKE	balanced	2	1.5	3.5	3
J-PAKE	balanced	4	4	8	4
A-EKE	augmented	1.5	1.5	3	5
B-SPEKE	augmented	3	3	6	3
SRP	augmented	2.5	2	4.5	4
zkPAKE	augmented	1.5	1.5	3	3

Table 1. Performance comparison among PAKE protocols, by number of messages and exponentiations computed by server/client. Powering an unknown base costs 1.0, a fixed base costs 0.5, and double exponentiation costs 1.5.

5. Conclusion

In this work we reviewed PAKE protocols, a method to establish secure cryptographic keys based only on the knowledge of a simpler password, focusing on augmented PAKEs. We proposed an augmented PAKE protocol that improves the performance of the protocols found in the literature, either in computation or communication costs. A formal security analysis is under way.

References

Anderson, R. J. (2001). *Security Engineering: A Guide to Building Dependable Distributed Systems*. John Wiley & Sons, Inc., New York, NY, USA, 1st edition.

Bellovin, S. M. and Merritt, M. (1992). Encrypted Key Exchange: Password-based Protocols Secure Against Dictionary Attacks. In *IEEE Computer Society Symposium on Research in Security and Privacy, Oakland, CA, USA*, pages 72–84.

Boyd, C. and Mathuria, A. (2003). *Protocols for Authentication and Key Establishment*. Information Security and Cryptography. Springer.

Chaum, D., Evertse, J., and van de Graaf, J. (1987). An Improved Protocol for Demonstrating Possession of Discrete Logarithms and Some Generalizations. In *Advances in Cryptology (EUROCRYPT), Amsterdam, The Netherlands*, pages 127–141.

Engler, J., Karlof, C., Shi, E., and Song, D. (2009). Is it too late for PAKE? In *Web 2.0 Security and Privacy Workshop (W2SP)*.

Hankerson, D., Menezes, A. J., and Vanstone, S. (2003). *Guide to Elliptic Curve Cryptography*. Springer-Verlag, Secaucus, NJ, USA.

Hao, F. and Ryan, P. (2010). J-PAKE: Authenticated Key Exchange without PKI. *Transactions on Computational Science*, 11:192–206.

Jablon, D. P. (1996). Strong Password-only Authenticated Key Exchange. *Computer Communication Review*, 26(5):5–26.

Lomas, T. M. A., Gong, L., Saltzer, J. H., and Needham, R. M. (1989). Reducing Risks from Poorly Chosen Keys. In *12th ACM SOSP*, pages 14–18.

Perlman, R. J. and Kaufman, C. (1999). Secure Password-Based Protocol for Downloading a Private Key. In *Network and Distributed System Security Symposium (NDSS)*.