Secure and Efficient Key Derivation in Portfolio Authentication Schemes Using Blakley Secret Sharing

Peter Mayer
SECUSO - Security, Usability and Society
Technische Universität Darmstadt, Germany
peter.mayer@secuso.org

Melanie Volkamer
SECUSO - Security, Usability and Society
Technische Universität Darmstadt, Germany
Privacy and Security Research Group
Karlstad University, Sweden
melanie.volkamer@secuso.org

ABSTRACT

The ubiquitous usage of mobile devices in public spaces increases the risk of falling victim to shoulder surfing attacks, i.e. being observed by others during authentication. A promising approach to mitigating such shoulder surfing risks is portfolio authentication. It requires only an authorized subset of the password as input during each authentication attempt. One open challenge regarding portfolio authentication is how to securely and efficiently verify that a user input is actually an authorized subset of the password. In this paper we propose the (t, n)-threshold verification scheme, a novel scheme using Blakley secret sharing to provide secure verification of all authorized subsets of the password. Due to the lack of a viable alternative, we evaluate the efficiency of the (t, n)-threshold verification scheme in comparison to a naive approach. In terms of storage, the (t, n)-threshold verification scheme outperforms the naive approach in all settings and it offers lower computation times in most settings.

Categories and Subject Descriptors

K.6.5 [**Management of Computing and Information Systems**]: Security and Protection—*Authentication*

Keywords

authentication, secret sharing, verification, key derivation

1. INTRODUCTION

In today's world authentication often occurs in public places or in the presence of friends or co-workers. In such situations shoulder surfing attacks can occur easily unless the required password is stored on the device in question.

One solution to counter shoulder surfing attacks are portfolio authentication schemes as studied by e.g. Dunphy et al. [8]. In portfolio authentication schemes the password is regarded as a set of elements and only a random subset of it has to be provided in each single authentication attempt. That way an attacker has a lower probability of authenticating successfully unless the user is observed multiple times.

The usual implementation of portfolio authentication is as challenge-response schemes. The verifier (e.g. service provider or smartphone) chooses at random which elements of the password need to be provided. The prover (e.g. client or user) has to respond to these challenges accordingly. Possibly the simplest example of such a scheme would be to use traditional text passwords as a basis and to ask the user to enter specific characters of her/his password in a challenge-response fashion. Due to usability considerations, it has been proposed to use portfolio authentication in conjunction with recognition-based authentication schemes. In such schemes the user does not have to actually recall the password, but only has to recognize it among distractor choices.

While the results in usability studies of portfolio authentication schemes are promising [8], one challenge currently faced in the implementation of such schemes is the lack of a verification scheme which allows secure and efficient verification of the input provided by the user. Efficiency in the scope of this paper has two dimensions: *storage* and *computation time*.

Usually the secure verification of user input is implemented using secure key derivation functions (KDF), e.g. keyed hash functions. Consequently, a naive approach could be to generate all authorized subsets of the password portfolio and store keyed hashes of all the subsets for later comparison to the information entered by the user. However, the required storage of this approach grows factorially in the difference of the sizes of the portfolio and the authorized subsets.

We propose the (t, n)-threshold verification scheme to tackle the challenge of secure and efficient verification of user inputs in portfolio authentication schemes. It represents a novel verification scheme providing a secure way to derive a common secret for all authorized subsets of the password. To achieve this, we use Blakley secret sharing [2] and key derivation functions. Blakley's scheme allows the preselection of shares easily (unlike other (t, n)-threshold secret sharing schemes such as Shamir's scheme [21]).

As a result, the required storage of the (t, n)-threshold verification scheme grows only polynomially. In the PIN-level security setting it requires six times less storage than the naive approach and in the password-level security setting ten times less storage respectively. Analogously to the storage efficiency, the (t, n)-threshold verification scheme offers a computational speedup in comparison to the naive approach

ACSAC '15, December 07-11, 2015, Los Angeles, CA, USA
© 2016 ACM. ISBN 978-1-4503-3682-6/15/12...$15.00
DOI: http://dx.doi.org/10.1145/2818000.2818043

when the number of authorized subsets grows. For example in a password-level security setting the (t, n)-threshold verification scheme offers a speedup of 13%.

The remainder of this paper is structured as follows. Section 2 presents background information on required terminology in authentication, recognition-based authentication, graphical authentication as well as portfolio authentication and explains the basic working principles of Blakley secret sharing. In section 3 we identify general operations and properties required of verification schemes in the domain of portfolio authentication. Section 4 describes the proposed (t, n)-threshold verification scheme along the identified operations and properties. Section 5 provides a security evaluation of our proposed scheme. Section 6 presents a comparison to a naive approach in terms of storage and computation time requirements. Section 7 summarizes and discusses the findings. Section 8 describes further applications of the proposed (t, n)-threshold verification scheme. Section 9 concludes and points out areas of future work.

2. BACKGROUND

In this section we present the necessary background regarding authentication terminology. In particular we will describe the phases involved in a general authentication procedure, present the basic principles of recognition-based authentication as well as graphical authentication and introduce relevant work in the area of portfolio authentication. Furthermore, we describe the basic working principles of Blakley's secret sharing scheme.

2.1 The Authentication Procedure

What is often referred to as authentication is actually a procedure with multiple phases, one of which is confusingly enough also called authentication. Since we refer to some of the phases throughout the paper, we clarify the scopes of the relevant phases in this section.

2.1.1 Enrollment

Before a user can authenticate using a certain authentication scheme, s/he has to be enrolled. The operation of the verification scheme associated with this phase is the *creation of the verification information.* In the traditional text password setting, this phase refers to the selection of a password by the user (in case the password is not assigned by the system), hashing the password and storing the hash together with the user name for later verification.

2.1.2 Identification

In the identification phase the user claims to have a certain identity. Traditionally the action required from the user in this phase is to supply the system with the user name. Note that in this phase the password is not entered yet.

2.1.3 Authentication

This phase is the most prominent one and therefore eponymous for the whole procedure: the authentication phase. In this phase the user enters the password using the authentication scheme. Note that here the term password is used in a very abstract sense and refers to anything from text or graphical passwords to biometrics or smart cards. In the traditional text password setting the respective user action is entering the password in a text field and submitting it to the system. This is the phase in which authentication schemes operate.

2.1.4 Verification

In the verification phase the password entered by the user during the authentication phase is verified against the information stored in the enrollment phase. This operation is called *verification.* In the traditional text password setting this refers to hashing the password supplied by the user during the authentication phase and comparing the calculated hash to the hash stored during the enrollment phase. Our proposed (t, n)-threshold verification scheme operates in this phase.

2.2 Recognition-based Authentication

Recognition-based authentication relies on different mechanisms than traditional recall-based authentication: instead of freely recalling the password, users have to decide whether presented information is familiar or not (i.e. is the information part of the password). Usually the familiar information is randomly assigned by the system during the enrollment phase and presented alongside distractor information in a challenge-response fashion during the authentication phase. The primary motivation behind recognition-based schemes is their increased memorability. It is a widely accepted fact that recognition is a cognitively much easier task than free recall (as necessary for traditional text passwords) [14, 24].

A simple example of a recognition-based scheme would be to assign a random string as traditional text password to the user during the enrollment phase. During the authentication phase this password could be displayed alongside other random textual strings. The task for the user would then be to point out her/his password among the choices. This scheme is obviously insecure beyond practicality and serves only for illustration purposes. However, multiple different recognition-based schemes have been proposed and studied with promising results using both, graphical passwords (e.g. [11, 7]) and (with slightly worse results in terms of memorability) text passwords (e.g. [26]).

2.3 Graphical Authentication

The motivation behind graphical authentication is the possibility to exploit the vast human capacity to store and process visual information, the so-called pictorial superiority effect [15]. Paivio's dual-coding theory [16] explains that visual information is stored differently in the human brain compared to abstract information such as text. This difference in encoding leads to a higher probability of memory imprinting and thus to a higher memorability. Many schemes aiming to exploit this potential exist, displaying different usability and security properties (see e.g. [23, 19, 25]).

Combining the memorability advantages of recognition-based authentication and graphical authentication schemes results in highly memorable passwords. There are numerous studies which found that graphical recognition-based schemes offer higher success rates and lower reset rates when compared to other knowledge-based authentication schemes (e.g. [13, 7, 5]).

One example of graphical recognition-based authentication is the Passfaces scheme presented in [17] and its derivatives found in the literature. Hlywa et al. evaluated three different Passfaces-style schemes in [11]. The password in these schemes is a set of images (the familiar information)

Figure 1: The interface of a recognition-based graphical authentication scheme resembling the interface[2] as used in [11]. The scheme shows multiple such grids one after the other. The task for the user is to point out the one image that is part of her/his password.

and usually randomly assigned to the user during enrollment. Each of the images in the password is part of a larger group. During authentication, multiple grids of images are presented to the user, one after another. Each grid corresponds to one such group. The user has to point out the one image that is part of her/his password (the familiar information) among the other images (the distractors). Figure 1 shows a grid resembling one of their schemes.

2.4 Portfolio Authentication

The primary motivation behind portfolio authentication is decreasing the probability of a successful shoulder-surfing attack. To achieve this, portfolio authentication schemes regard the password as a set of elements. These elements can have any form (e.g. textual characters, textual strings, images, electronic certificates, etc.). A password P of length n is thus represented as

$$P = \{e_1, e_2, \dots, e_n\}.$$

During each authentication attempt only a random subset $P' \subseteq P$ of these elements has to be entered by the user. We borrow from the nomenclature of secret sharing and denote such a subset P' as authorized if it has at least t elements, where the parameter t is set by each authentication scheme depending on the desired security properties. In the remainder of this paper we refer to the magnitude by which the password is larger than the authorized subsets as portfolio overhead (denoted as o). It is represented as a fraction of the form

$$o = \frac{|P|}{|P'|} = \frac{n}{t}.$$

Note that usually $|P'|$ serves as starting point when determining o and the full password is then chosen accordingly (either by the system or by the user while respective policies guide the user's choice), since the strength of the authentication scheme against guessing attacks is dependent on $|P'|$ rather than $|P|$.

Portfolio authentication is most useful to prevent shoulder-surfing in public spaces or while in the presence of friends or co-workers. It thwarts naive shoulder-surfers that do not observe the user for a longer period of time. An attacker who can capture multiple (possibly any arbitrary number of) authentication attempts or who uses a recording device might still be able to break the portfolio authentication scheme, depending on the actual authentication scheme's design.

The most common application of portfolio authentication are challenge-response schemes. Thus the elements e_i are usually challenge-response pairs. In particular graphical recognition-based authentication schemes have been used. Dhamija and Perrig first introduced the portfolio term for graphical recognition-based authentication schemes [7]. DeAngeli et al. proposed the portfolio approach for graphical recognition-based passwords as high security setting [6]. In their scheme P' is a random subset of the password images and is chosen as challenge set for each authentication attempt.

Dunphy et al. first proposed portfolio authentication as a measure to mitigate shoulder-surfing risks on mobile phones in public places [8]. For their shoulder-surfing study they report a temporary resistance against shoulder-surfing, especially in the context of lunchtime attacks.

2.5 Blakley Secret Sharing

Cryptographic (t, n)-threshold secret sharing denotes cryptographic protocols to distribute sensitive information (the secret) among n parties such that only by collaboration of t such parties the secret can be reconstructed. These protocols usually consist of two phases: (1) the dealing phase in which each party is assigned a secret share and (2) the combination phase in which t or more parties can collaborate to reconstruct the secret using their shares.

Blakley proposed to use hyperplane geometry to solve this problem [2]. The shared secret is defined as the first coordinate of a randomly chosen point x in a t-dimensional vector space over a Galois field $GF(p)$, where p is a prime. Throughout the remainder of this paper $i \in \{1, \dots, n\}$ denotes the index of the party and $j \in \{1, \dots, t\}$ denotes the coordinate in the t-dimensional vector space.

2.5.1 Dealing Phase

To distribute the shares, the dealer chooses a sufficiently large prime p and a t-dimensional point

$$x = \begin{pmatrix} x_1 \\ \vdots \\ x_t \end{pmatrix}$$

at random. Its first coordinate x_1 serves as the secret. Then the shares for each of the n parties are generated. The dealer chooses t values m_{ij} at random for each party and calculates the shares y_i using the equation:

$$m_{i1}x_1 + m_{i2}x_2 + \cdots + m_{it}x_t = y_i \quad (1)$$

The resulting $n \times t$ matrix (the entirety of all coefficients m_{ij}) is denoted by M. M is public information and does

[2]The interface was recreated using images from the same source (www.freeimages.com). All images © Getty Images.

not need to be kept secret [4]. It can be stored in the clear. The dealer distributes only the values y_i to the n parties.

2.5.2 Combination Phase

To reconstruct the secret, t parties need to combine their shares and the respective coefficients from M to form the system of equations

$$M'x = y', \tag{2}$$

where y' is the vector of shares provided by the parties and M' is the $t \times t$ matrix of the respective coefficients. This linear system of equations is then solved for x. The reconstructed secret is x_1.

3. REQUIREMENTS

In this section we identify required operations and properties that any verification scheme (i.e. any scheme that operates in the verification phase) used for portfolio authentication should provide. The requirements presented here are mainly derived from findings and properties described by Pieprzyk et al. [18].

3.1 Necessary Operations

There are two primary operations every password verification scheme has to offer, independently of its use in a portfolio setting. Firstly, the scheme must be able to *create the verification information* by translating the password into verifiable information during enrollment. Secondly, the scheme must offer an operation for the actual *verification*.

3.2 Optional Operations

While not strictly necessary, additional operations might be desirable when a verification scheme is used with portfolio authentication to allow for a higher usability in some re-enrollment scenarios.

Adjusting the Portfolio Overhead: A change in the security policy of the authentication scheme might require the user to enter a smaller or larger subset of her/his password for each authentication attempt. This corresponds to adjusting the portfolio overhead. The operation especially offers benefits when used in conjunction with recognition-based authentication schemes. If the portfolio overhead can be adjusted without changing the password, this can prevent the user from having to learn completely new passwords in such cases. Instead, only additional challenge-response pairs need to be learned.

Adding and Removing Challenges: In case a challenge-response pair is known to be compromised, it is desirable to allow removal of the respective pair and addition of a replacement pair. Ideally this operation involves only changing the respective pairs, while leaving the remaining portfolio untouched. That way, the user does not have to learn a completely new password but only the respective challenges.

3.3 Security Requirements

The primary goal of any authentication scheme is to protect access-restricted resources. Therefore it is imperative that a verification scheme never impairs an authentication scheme's security properties. In terms of security the focus of verification schemes lies on two aspects, namely the *guessing resistance* and *secure storage*.

Guessing Resistance: When considering possible attacks, the resistance against guessing attacks seems to be the most relevant in terms of verification [18]. The strength of every authentication scheme against guessing attacks depends on its password space. Therefore, a verification scheme should not decrease the space of possible passwords. It needs to be adaptable to the desired strength of the authentication scheme.

Secure Storage: It has long been best practice to not store passwords in the clear [3]. Consequently no verification scheme should rely on the availability of the password in the clear i.e. prevent its storage in a cryptographically hashed form.

3.4 Efficiency Requirements

Any proposed verification scheme should ideally be more efficient than naive approaches such as the one outlined in section 1. Efficiency in the scope of this paper has two dimensions: *storage* and *computation time*.

Storage: In terms of storage, the influence of the portfolio overhead should be minimized in order to keep the cost of the shoulder-surfing resistance as low as possible.

Computation Time: In terms of computation time, the necessary operations as stated above are of concern (i.e. *creation of the verification information* and *verification*). However, it is widely considered best practice to artificially prolong verification of passwords by repeated application of key derivation functions (often several thousand times) in order to increase the cost of guessing attacks. Therefore, slower operations can simply be compensated by decreasing the iterations of the KDF. This of course only applies as long as the verification operation does not take longer than the repeated application in the non-portfolio setting.

4. (T,N)-THRESHOLD VERIFICATION

The verification scheme we propose in the following uses Blakley secret sharing and key derivation functions. It derives the same secret from all authorized subsets of the password. We refer to our approach as (t, n)-threshold verification scheme, where n is the number of elements in the password and t the threshold (analogously to the definition of (t, n)-threshold secret sharing). Before we describe each of the operations identified in section 3 along the phases described in section 2.1, we present preconsiderations to motivate the choice of Blakley's secret sharing scheme over other alternatives. An implementation of the enrollment and verification operations is freely available at our GitHub repository[3].

4.1 Preconsiderations

In usual secret sharing scenarios the shares are chosen at random. This is not possible in our verification scenario. The password is created by the authentication scheme and does not necessarily comply with the creation procedure of any secret sharing scheme. For example in Shamir's scheme [21] our scenario would require to derive a polynomial of degree $t-1$ which fits all n preselected points. This is usually not possible since the Lagrange interpolation polynomials are unique for each set of $t < n$ points.

We chose Blakley's scheme instead of other secret sharing schemes, because it offers the required property: it allows to predetermine the individual shares with only slight modifications. As outlined in section 2.5, the hyperplanes are

[3]https://github.com/SecUSo/t-n-threshold-verification

normally chosen at random during the dealing phase. However, by using cryptographic hash functions it is possible to derive pseudo-random values from the password and choose the remaining coefficients of M accordingly (see the following sections for details).

4.2 Enrollment

The basic working principles of Blakley's secret sharing scheme are unchanged for the (t, n)-threshold verification scheme. Each challenge-response pair in the authentication scheme corresponds to one party in the secret sharing scheme. The first step remains choosing a suitable p and a t-dimensional point

$$x = \begin{pmatrix} x_1 \\ \vdots \\ x_t \end{pmatrix}.$$

Its first coordinate (i.e. x_1) is the common secret and corresponds to the actual password in a traditional text password scheme. Consequently, it should only be stored after the application of an appropriate KDF. In the following, the value derived from x_1 is denoted $s = KDF(x_1)$. Care should be taken when choosing the value p for $GF(p)$. The larger p is, the more resilient the scheme is to guessing attacks (see section 5.1 for details).

The rest of the creation procedure deviates slightly from Blakley's secret sharing scheme. Only $t-1$ coefficients m_{ij} are chosen at random. Additionally, the shares y_i are derived from the user-chosen or assigned password elements e_i using a cryptographic hash function. This operation can also be delegated to the authentication scheme, depending on the actual implementation. The remaining coefficient m_{it} is calculated using equation (1) and the values m_{ij} and y_i chosen before. This is a notable deviation from Blakley's original procedure insofar as the distributed share is predetermined by the system and not chosen at random. However, the use of cryptographic hash functions ensures that all values are indistinguishable from randomly chosen values. Thus the security properties of Blakley's scheme remain unchanged (assuming the usage of a cryptographically strong hash function; see section 5 for details).

All coefficients of M and the value s are retained and stored for the verification procedure. Together they represent the verification information and correspond to the password hash in a traditional text password setting.

Note that when using Blakley secret sharing in affine geometry as we do here, it is important to ensure during the creation of the verification information that none of the hyperplanes are parallel (i.e. the determinants of all $t \times t$ submatrices M' of M are unequal to zero). Otherwise, there exist M' which are not uniquely solvable, i.e. the secret point x is not unique.

4.3 Verification

Whenever a user wants to authenticate to the system, first the user's inputs to the authentication scheme e'_i need to be collected during the authentication phase. The collected e'_i are used in the verification phase to derive the values y'_i in the same fashion as during the creation procedure. Then, the y'_i and the stored coefficients m_{ij} are used to form the linear system of equations (2) which is solved for x. The value s' obtained by application of the respective KDF to x_1 is then compared to the stored common secret s. During verification there is no deviation from Blakley's original procedure. Thus all authorized subsets can recover the common secret.

4.4 Re-Enrollment

The (t, n)-threshold verification scheme provides the optional operations identified in section 3, namely *adjusting the portfolio overhead* and *adding and removing challenges*. The procedures involved in these two operations are described in the following.

Adjusting the Portfolio Overhead: Adjusting the portfolio overhead corresponds in our scheme to changing the threshold t. In order to adjust the threshold (i.e. transforming the (t, n)-threshold verification scheme into a $(t \pm k, n)$-threshold verification scheme), the matrix M needs to be recreated. This requires the point x or all shares y_i. While an authorized subset is not sufficient for the actual adjustment, any authorized subset of the password can recover x. Using the recovered x, the procedure is then similar to the original creation procedure outlined in 4.2. Only the number of columns in M needs to be adjusted according to the desired change (i.e. the index in each row of M is adjusted to $j \in \{1, \ldots, t \pm k\}$).

Adding and Removing Challenges: To add further challenge-response pairs (i.e. transforming the (t, n)-threshold verification scheme into a $(t, n+1)$-threshold verification scheme) another row has to be added to M. To perform the necessary calculations x has to be reconstructed (which is possible, given any authorized subset of the password). With x the creation procedure using equation (1) as outlined in section 4.2 can be used to add another row to M.

Removing a challenge from the scheme (i.e. transforming the (t, n)-threshold scheme into a $(t, n-1)$-threshold scheme) is trivial. The respective share and coefficients are simply removed. Of course this is only viable if $n - 1 \geq t$.

5. SECURITY EVALUATION

In this section we evaluate the proposed (t, n)-threshold verification scheme regarding the security aspects identified in section 3.3.

5.1 Guessing Resistance

Blakley secret sharing is a perfect secret sharing scheme [18]. The guessing resistance of the (t, n)-threshold verification scheme is directly related to the size of $GF(p)$. As there are only p distinct values any variable in the system can assume, any attacker needs on average $\frac{p}{2}$ attempts to guess the correct value. This holds obviously if the attacker tries to guess x_1 directly. Additionally, guessing the secret x_1 does not get easier if a share is known to the attacker. If the attacker tries to guess the correct shares y_i and was (in the worst case) able to obtain all but one share s/he has again to try on average $\frac{p}{2}$ values for the remaining share: By definition the linear system of equations (2) has one solution, since the determinant of the $t \times t$ matrix M is unequal to zero for all possible vectors of shares y. Consequently, all shares $y_i \in \{1, 2, \ldots, p\}$ are equiprobable. Even when the number of available shares is constrained by the authentication scheme, guessing the share is not easier than a standard brute force attack.

To ensure the security properties outlined before, it is important to choose p as explained below, so that the space of actually chosen passwords is not shrunk. Note that in the following we assume that the passwords (i.e. the sets $P = \{e_1, \ldots, e_n\}$) are randomly chosen (i.e. user choice is not modelled). Following the classical information theoretic argumentation in [12] it is of the essence to ensure that

$$H(s) \leq -\sum_{i=1}^{p} \frac{1}{p} \log_2 \left(\frac{1}{p} \right).$$

As stated before, the attacker has to test on average $\frac{p}{2}$ values to to find x_1. Consequently, p should ideally be chosen such that

$$p \geq 2^{H+1}, \tag{3}$$

where H is the expected strength of the authentication scheme in bit. Otherwise, guessing one share is easier than guessing an authorized subset of the password.

5.2 Secure Storage

Our (t, n)-threshold verification scheme provides secure storage of the verification information. Analogously to the procedure common in traditional text password settings, the common secret x_1 is only stored for later verification of the user input after the application of an appropriate KDF. As long as the KDF is secure, the secret is secure.

6. EFFICIENCY EVALUATION

In this section we present an efficiency evaluation of the (t, n)-threshold verification scheme in comparison to the naive approach outlined in section 1 along the two efficiency dimensions outlined in section 3.4: *storage* and *computation time*. First however, we describe the naive approach and the comparison setup in greater detail.

6.1 Naive Approach

To our knowledge there is no other proposal for key derivation and verification of authentication information in portfolio authentication schemes described in published literature. Therefore, in the absence of viable alternatives, we use the following naive verification scheme as a baseline for our efficiency comparison.

The naive scheme creates hashes for all authorized subsets during the enrollment phase and stores all these hashes. During the verification phase the user's input is hashed and compared to all possible hashes. The verification is successful if the hashed user input is equal to one of the stored hashes.

The required storage in bytes of this naive approach grows factorially in the portfolio overhead and can be determined using the equation

$$b_{\text{naive}} = \binom{n}{t} \cdot b_{\text{hash}}, \tag{4}$$

where n denotes the total number of elements in the password, t denotes the size of the authorized subsets (i.e. the number of elements required during authentication) and b_{hash} denotes the size of one hash in bytes.

6.2 Comparison Setup

From a guessing-resistance point of view, knowledge-based authentication exists mostly at two security levels: the *PIN-level* and the *password-level*. Therefore, the main part of the evaluation focuses on these two levels. The portfolio overhead of $o = \frac{3}{2}$ as used in [8] is applied in all configurations.

The PIN-level spans a password space of 10^4 entries. It is used widely, from unlocking smartphones to banking applications. To achieve this security level in a portfolio setting, we consider random PINs of length 6, where 4 elements of the PIN have to be entered during the authentication phase over the usual 10 digit alphabet. For the (t, n)-threshold verification scheme we choose $p = 2^{15} - 12757$, so that guessing x_1 is harder than guessing the actual user secret (i.e. an authorized subset of the PIN).

The password-level is more ambiguously defined, but can generally be regarded as the level achieved by passwords that occur in the wild. Florêncio and Herley found that password policies used by large Internet companies result in minimum strengths of about 20 to 27 bits [10]. To approximate realistic values, but still provide a clear contrast to the PIN-level, we choose the value of 27 bits. We decided to increase the password length as well as the size of the alphabet to clearly contrast the configuration of the PIN-level and the password-level. Therefore, we assume random passwords of length 9, where 6 elements of the password have to be entered during each authentication attempt and an alphabet of of size 23 ($23^6 > 2^{27}$). For the (t, n)-threshold verification scheme we choose $p = 2^{32} - 5$, so that guessing x_1 is harder than guessing any authorized subset of the password.

To explore the properties of the (t, n)-threshold verification scheme beyond these two security levels, we provide results for six additional configurations. These additional configurations are based on random strings over the broadly considered alphabet of the 95 characters on a standard US keyboard (see e.g. [11]). The recommended key lengths for high security cryptographic keys given by Eastlake et al. [9] serve as upper bound in terms of password strength. At the time of this writing appropriate key lengths are in the 89- to 104-bit range. The parameter p for these configurations is chosen as the largest prime representable with the same number of bytes as the minimum value for p as determined by equation (3).

Barker et al. provide recommendations in terms of key length and respective hash functions [1]. At the time of writing SHA-256 is an adequate choice. Thus SHA-256 (hash size 32 bytes) is considered as hashing algorithm throughout the whole evaluation. We decided to use salted SHA-256 hashing as KDF for both, the naive and the (t, n)-threshold verification scheme.

6.3 Storage

The first aspect of this comparison is the required storage. The proposed (t, n)-threshold verification scheme needs to store the common secret s and the coefficients m_{ij}. Both are in $GF(p)$. As outlined above in section 5.1, using a sufficiently large prime number p is essential in order to not decrease the security of the used scheme against guessing attacks. The number of coefficients needed depends on the overall password size (number of elements) and the size t of the authorized subsets. The storage requirement b in bytes is given by: (a) the $n \cdot t$ coefficients of M, whose size is determined by the number of bytes necessary to store one integer smaller or equal to p (rounded up), plus (b) the stored secret s, whose size is determined by the number of bytes necessary to store the value (rounded up). Formally

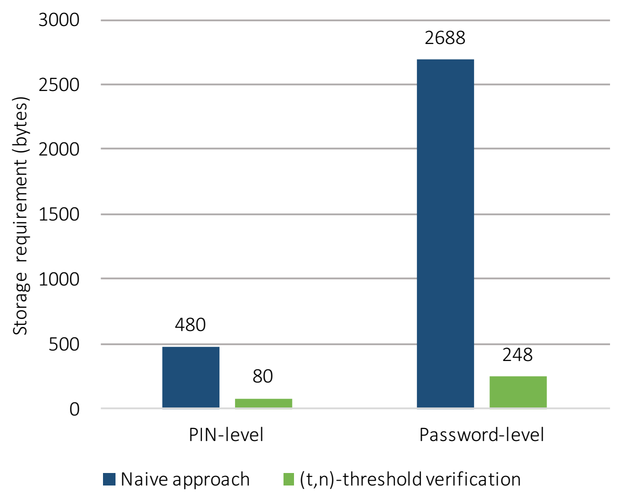


Figure 2: The storage requirements in bytes. Lower values are better.

this can be expressed using the equation

$$b_{(t,n)-\text{threshold}} = n \cdot t \cdot \left\lceil \frac{\log_2(p)}{8} \right\rceil + b_{\text{hash}}, \tag{5}$$

where b_{hash} denotes the size of the hash s of x_1 in bytes (rounded up). From this equation it becomes apparent that the required storage grows polynomially in n and t. Figure 2 shows the storage requirement for both schemes on the PIN-level and on the password-level. Calculations for the values of both levels and the additional configurations is given below, a summary of the results can be found at the end of this section.

6.3.1 *PIN-level*

Using equation (4) with settings for the PIN-level, the overall storage requirement for the naive approach is

$$\binom{6}{4} \cdot 32 \text{ bytes} = 480 \text{ bytes}.$$

For the proposed (t, n)-threshold verification scheme, the storage requirement in the PIN-level setting is

$$6 \cdot 4 \cdot 2 \text{ bytes} + 32 \text{ bytes} = 80 \text{ bytes}.$$

6.3.2 *Password-level*

Using the naive approach on the password-level, the overall storage needed is

$$\binom{9}{6} \cdot 32 \text{ bytes} = 2688 \text{ bytes}.$$

The storage requirement of the (t, n)-threshold verification scheme in this setting is

$$9 \cdot 6 \cdot 4 \text{ bytes} + 32 \text{ bytes} = 248 \text{ bytes}.$$

6.3.3 *Properties beyond password-level security*

Table 1 shows the storage requirements of the six additional configurations based on the settings outlined in section 6.2. From the data it becomes apparent that the more authorized subsets there exist for one password, the more storage is required by both, the naive approach and the (t, n)-threshold verification scheme. However, due to the different growths of the required storage in both approaches (polynomial vs. factorial) the difference between the naive approach and the (t, n)-threshold verification scheme steadily increases. The (t, n)-threshold verification scheme is significantly more efficient in terms of storage than the naive approach for large numbers of authorized subsets.

Table 1: The storage requirements in bytes b of the proposed (t, n)-threshold verification scheme for the additional configurations. The values for the naive approach are also given for reference. H is the desired strength of the authentication scheme. n and t are the portfolio parameters.

H	n	t	$b_{(t,n)-\text{threshold}}$	b_{naive}
39,42	9	6	302	2688
52,56	12	8	704	15840
65,70	15	10	1382	96096
78,84	18	12	2192	594048
91,98	21	14	3560	3720960
105,12	24	16	5408	23535072

6.3.4 *Results*

The storage requirement of the (t, n)-threshold verification scheme is six times smaller in the PIN-level security setting and ten times smaller in the password-level security setting than for the naive approach. The additional configurations let this difference become even more apparent: the longer the password is (assuming the same portfolio overhead), the larger the difference becomes. Yet, it is important to acknowledge that using portfolio authentication with either approach comes at a price. The storage requirement is much larger than in the non-portfolio scenario, where only one hash has to be stored for each password.

6.4 Computation Time

The second part of our efficiency evaluation is a comparison of the computation timings. We employ an empirical Monte Carlo evaluation. The two necessary operations in this regard are *creation of the verification information* and *verification*. Since the naive approach does not support the additional operations identified in section 3, they are left out of the comparison. The reported values are means of 10000 operation runs for the PIN-level and password-level and means of 1000 operation runs for each of the additional configurations. All calculations were run using an implementation in Mathematica 9 on a Core 2 Duo 2.4 GHz machine. Note that due to the test setting, the focus is on the magnitude of the difference, not the actual measured timings. A summary of the results can be found at the end of this section.

We chose to apply the KDF only once to the derived secrets for both, the naive and the (t, n)-threshold verification scheme. Consequently, in a real world setting the timings would increase according to the number of iterations I chosen for the KDF. In the naive scheme the hash of every authorized subset has to be calculated by applying I iterations (i.e. $\binom{a}{b} \cdot I$ iterations in total). In the (t, n)-threshold verification scheme the KDF has to be applied only to the value s (i.e. I iterations in total). Therefore, the values provided here can also be regarded as a lower bound for scenarios where the KDF is applied only a few times (e.g. due to computation timing constraints).

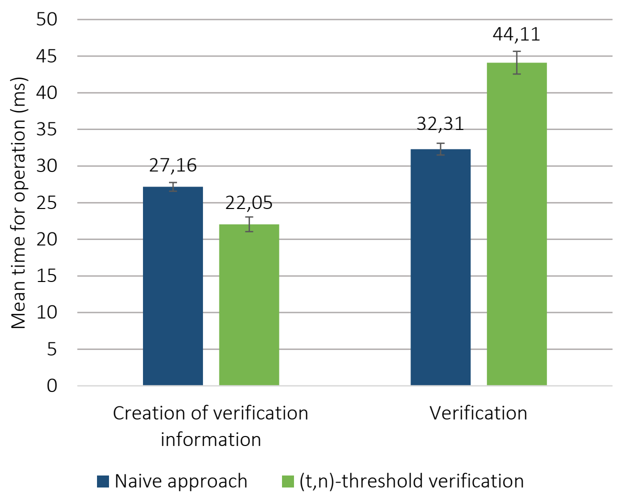


Figure 3: The computation time requirements for the PIN-level setting. The depicted values are the means and standard deviations calculated from 10000 timings captured in a Monte Carlo evaluation. Lower values are better.

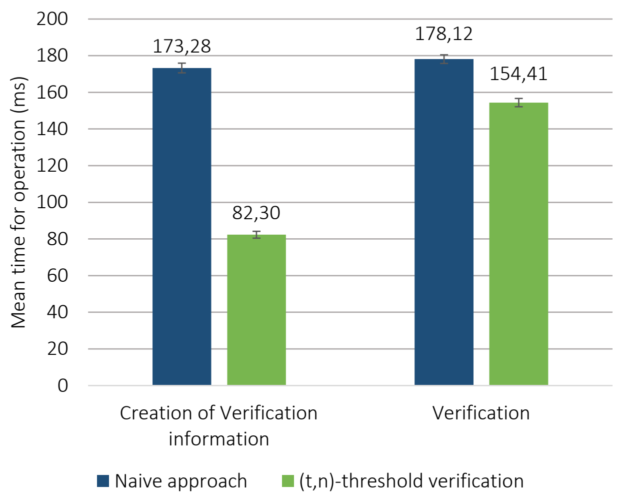


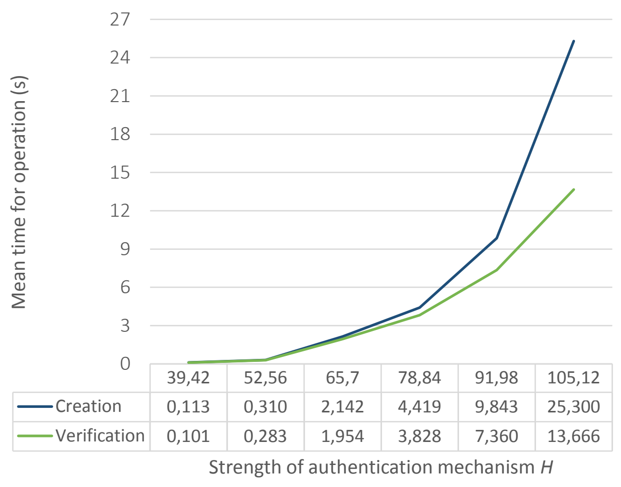
Figure 4: The computation time requirements for the password-level setting. The depicted values are the means and standard deviations calculated from 10000 timings captured in a Monte Carlo evaluation. Lower values are better.

6.4.1 *PIN-level*

Figure 3 shows the mean times obtained for both operations in the PIN-level setting. The timings for the creation of the verification information show an advantage of our (t, n)-threshold approach. The mean time is about 23% higher for the naive approach. Regarding the verification timings the naive approach is faster. The simple comparison of hashes is less complex than solving the linear system of equations in our approach. The advantage of the naive approach is about 36%.

6.4.2 *Password-level*

Figure 4 shows the mean times obtained for both operations in the password-level setting. The values for the creation of the verification information draw a clearer picture than those of the PIN-level. The time needed using the (t, n)-threshold verification scheme is about 53% lower. In contrast to the PIN-level, on the password-level verification is faster when using the (t, n)-threshold verification scheme. With about 13% the difference is, however, rather small.



	39,42	52,56	65,7	78,84	91,98	105,12
Creation	0,113	0,310	2,142	4,419	9,843	25,300
Verification	0,101	0,283	1,954	3,828	7,360	13,666

Figure 5: The results of the computation timing evaluation of the additional configurations.

6.4.3 *Properties beyond password-level security*

Figure 5 shows the timings of the additional configurations as function of the respective password strength H. For details of the configurations refer to Table 1. Analogously to the storage evaluation, the computation time increases with longer passwords and the number of authorized subsets that exist for the password. In contrast to the results for the PIN-level and password-level, the creation of the verification information takes longer than the verification for all additional configurations. A thorough evaluation is needed to further investigate this issue.

6.4.4 *Results*

The (t, n)-threshold verification scheme offers faster creation of the verification information on both security levels under consideration (PIN-level and password-level). The verification has more mixed results. However, it becomes clear that in all cases, the repeated application of the key derivation function would dominate the computation timings. 10000 rounds (a value used by large Internet companies and also recommended by the OWASP [22]) of simple keyed SHA-256 hashing take about 17.5 seconds in the same computation setup, which is more than one hundred times the value for the verification operation at the password-level. Even the value for the verification operation at the highest security level we evaluated is lower. Some of the results require further investigation. A thorough evaluation investigating the effects of all parameters of the (t, n)-threshold verification scheme is required.

7. DISCUSSION

This paper proposes the (t, n)-threshold verification scheme, a novel verification scheme using Blakley secret sharing to facilitate secure and efficient verification in portfolio authentication schemes.

In terms of efficiency, two aspects were evaluated in comparison to a naive approach, namely the *required storage* and *computation timings*. The comparison of the required storage revealed that the proposed scheme requires significantly less storage space for the verification information. The required storage of the (t, n)-threshold verification scheme grows only polynomially, while the storage of the naive approach grows factorially in the length of the password and the size of the authorized subsets. The efficiency in terms of storage space can be regarded as the most important trait of the (t, n)-threshold verification scheme. In relevant usage scenarios it requires six to ten times less storage than a naive approach. This advantage increases further, the larger the number of authorized subsets for each password is.

The evaluation of the computation timings draws a similar, but slightly more diverse picture. On the one hand, the (t, n)-threshold verification scheme is faster than the naive approach in both operations for the password-level security setting. On the other hand, the verification operation of the naive approach is faster than the (t, n)-threshold verification scheme for the PIN-level security setting. However, this holds only when the key derivation function is applied few times to the secret as it was done in our evaluation. The longest computation time found in the evaluation of the PIN-level and password-level configurations (verification for the naive approach in the password-level security setting) is equivalent to about 102 SHA-256 iterations in the same computation setting. This number fades in comparison to the multiple thousand iterations usually applied in order to render guessing attacks more costly. Thus we argue that the increase in computation time of the (t, n)-threshold verification scheme (in comparison to non-portfolio authentication schemes) can easily be compensated by slight adjustments of the iterations of the key derivation function used for the common secret.

Consequently, the (t, n)-threshold verification scheme is most suitable for scenarios in which the number of authorized subsets is large. When storage is of no concern, but computation time is important (and the KDF is applied only few times) the naive approach might be more suited for scenarios with smaller numbers of authorized subsets. However, it must be acknowledged that in all cases the required storage and computation timings are significantly higher than in non-portfolio scenarios. Thus the increased resilience to shoulder-surfing attacks comes at the price of additional storage and computation time.

However, with respect to the verification operation, the increased computation time can also be seen as an advantage. The scheme's verification timings scale with the complexity of the password: the higher the security in terms of resistance to shoulder-surfing, the longer an attacker needs to calculate one guess. Further investigations are required to explore the effects of this trait on the security and computation requirements of the (t, n)-threshold verification scheme.

In terms of security, the properties regarding *secure storage* and the *guessing resistance* of the (t, n)-threshold verification scheme were investigated. Regarding the secure storage of the authentication information it could be shown that the choice of a secure KDF allows secure storage for the (t, n)-threshold verification scheme. Regarding the guessing resistance, the verification scheme can be adapted to the desired strength of the authentication scheme by choosing p large enough according to equation (3). However, it becomes apparent from equation (5) that balancing the storage requirements and the resistance to guessing attacks by carefully choosing the parameter p can be of the essence. For PIN-level or password-level secrets and in scenarios where storage efficiency is not of utmost importance, this is not critical. However, when longer secrets are stored and storage efficiency is of the essence, the advantage in terms of storage can decrease.

In the course of our investigations, required operations and properties for verification schemes in the context of portfolio authentication were identified. The (t, n)-threshold verification scheme offers operations which can render reenrollment more user-friendly. Both, *adjusting the portfolio overhead* as well as *adding or removing challenges*, require reissuing a new password to the user in the naive approach. In contrast, the (t, n)-threshold verification scheme allows these operations without the need to issue new passwords.

8. FURTHER AREAS OF APPLICATION

While portfolio authentication was originally designed with graphical recognition-based schemes in mind, it can be applied to a more general context. It is a natural extension of the (t, n)-threshold verification scheme to support passwords that comprise elements of different types (i.e. originate from different authentication schemes). Such an extension requires only an adequate hash function to derive the respective share from the user input. Therefore, it is conceivable to allow multi-factor passwords in systems in which not all factors have to be provided for each authentication attempt (e.g. one element of the password could be a text password, a second element a certificate on a smart card, a third element a biometric such as a fingerprint, another element a USB-token and so on, but only two of these are required at login). This can be beneficial in scenarios where not all factors are available at all times or where factors are prone to error.

All devices and authentication schemes come with design spaces that are specific to the respective device and authentication scheme (e.g. [20]). Using the proposed (t, n)-threshold verification scheme, each device can offer multi-factor authentication tailored to its specific requirements and features (screen size, presence of keyboard or touch-screen, biometric sensors etc.) while still being able to verify all the information securely and efficiently. As such, our scheme facilitates novel authentication procedures spanning multiple devices with improved security and usability due to device-specific considerations regarding available authentication methods. The authentication scheme can then meet each device's requirements while respecting user preferences instead of providing a one size fits (not) all solution for all users on all devices.

9. CONCLUSION

In this paper we propose the (t, n)-threshold verification scheme. It serves as an important enabler for other technologies and allows the implementation of portfolio authentication schemes including secure and efficient verification for the first time. Especially in terms of storage efficiency the (t, n)-threshold verification scheme shows its strengths.

Recognition-based authentication schemes that are more resilient to shoulder-surfing attacks than their non-portfolio variants are one important use case that benefits greatly from the proposed verification scheme. However, we also discussed how the possible applications of the proposed (t, n)-threshold verification scheme go far beyond this one scenario. It enables multi-factor setups that stretch multiple devices to deliver the most user friendly experience possible.

Future work is required in three areas. Firstly, a thorough evaluation in terms of computation timings in a more realistic computation setting is required. Such an evaluation allows (1) to investigate the influence of all parameters involved in the (t, n)-threshold verification scheme and (2) to obtain representative timings additionally to the overview of differences in magnitude provided in this paper. Secondly a prototype application in the graphical recognition-based authentication domain should be implemented and evaluated. Thirdly, a suitable multi-device/multi-factor scenario should be explored. Such a scenario would also enable a comparison of the impact of different factors (e.g. biometrics, smart cards or knowledge based authentication) on the performance of the scheme.

10. ACKNOWLEDGMENTS

We would like to thank Stephan Neumann, Johannes Braun and Matthias Schempp for their valuable input.

This work has been developed within the project 'KMU AWARE' which is funded by the German Federal Ministry for Economic Affairs and Energy under grant no. BMWi-VIA5-090168623-01-1/2015. The authors assume responsibility for the content.

11. REFERENCES

[1] E. Barker, W. Barker, W. Burr, W. Polk, and M. Smid. NIST Special Publication 800-57 - Recommendation for Key Management Part 1: General (Revision 3). U.S. Department of Commerce - National Institute of Standards and Technology, 2012.

[2] G. R. Blakley. Safeguarding cryptographic keys. In *Proceedings of the national computer conference. Vol. 48*, 1979.

[3] J. Bonneau and S. Preibusch. The password thicket: technical and market failures in human authentication on the web. *WEIS '10: The 9th Workshop on the Economics of Information Security*, 2010.

[4] l. N. Bozkurt, K. Kaya, A. A. Selçuk, and A. M. Güloglu. Threshold Cryptography Based on Blakley Secret Sharing. *Information Sciences*, 2008.

[5] S. Brostoff and M. A. Sasse. Are Passfaces more usable than passwords? A field trial investigation. In *People and Computers XIV - Usability or Else!*, pages 405–424. Springer, 2000.

[6] A. De Angeli, L. Coventry, G. Johnson, and K. Renaud. Is a picture really worth a thousand words? Exploring the feasibility of graphical authentication systems. *International Journal of Human-Computer Studies*, 63(1):128–152, 2005.

[7] R. Dhamija and A. Perrig. Deja vu: A user study using images for authentication. In *USENIX Security Symposium*, pages 45–58, 2000.

[8] P. Dunphy, A. P. Heiner, and N. Asokan. A closer look at recognition-based graphical passwords on mobile devices. In *SOUPS '10: Proceedings of the Sixth Symposium on Usable Privacy and Security*. ACM, 2010.

[9] D. Eastlake, J. Schiller, and S. Crocker. *Request for Comments: 4086 - Randomness Requirements for Security*. The Internet Society, 2005.

[10] D. Florêncio and C. Herley. Where do security policies come from? In *SOUPS '10: Proceedings of the Sixth Symposium on Usable Privacy and Security*. ACM, 2010.

[11] M. Hlywa, R. Biddle, and A. S. Patrick. Facing the facts about image type in recognition-based graphical passwords. In *ACSAC '11: Proceedings of the 27th Annual Computer Security Applications Conference*, pages 149–158. ACM, 2011.

[12] E. Karnin, J. Greene, and M. Hellman. On secret sharing systems. *IEEE Transactions on Information Theory*, 29(1):35–41, 1983.

[13] P. Mayer, M. Volkamer, and M. Kauer. Authentication Schemes - Comparison and Effective Password Spaces. In *10th International Conference on Information Systems Security*, pages 204–225, 2012.

[14] E. F. Mulhall. Experimental Studies in Recall and Recognition. *The American Journal of Psychology*, 26(2):217–228, 1915.

[15] D. L. Nelson, V. S. Reed, and W. John R. Pictorial Superiority Effect. *Journal of Experimental Psychology: Human Learning and Memory*, 2:523–528, 1976.

[16] A. Paivio, T. B. Rogers, and P. C. Smythe. Why are pictures easier to recall than words? *Psychonomic Science*, 11:137–138, 1968.

[17] Passfaces Corporation. *The Science Behind Passfaces*. Passfaces Corporation, 2006.

[18] J. Pieprzyk, T. Hardjono, and J. Seberry. *Fundamentals of Computer Security*. Springer, 2003.

[19] A. Salehi-Abari, J. Thorpe, and P. C. van Oorschot. On purely automated attacks and click-based graphical passwords. In *Computer Security Applications Conference, 2008. ACSAC 2008. Annual*, pages 111–120. IEEE, 2008.

[20] F. Schaub, M. Walch, B. Könings, and M. Weber. Exploring The Design Space of Graphical Passwords on Smartphones. In *SOUPS '13: Proceedings of the Ninth Symposium on Usable Privacy and Security*. ACM, 2013.

[21] A. Shamir. How to share a secret. *Communications of the ACM*, 22(11):612–613, 1979.

[22] J. Steven and J. Manico. Password Storage Cheat Sheet (visited May 2015). https://www.owasp.org/index.php/Password_Storage_Cheat_Sheet.

[23] X. Suo, Y. Zhu, and G. S. Owen. Graphical passwords: A survey. In *ACSAC '05: Proceedings of the 21st Annual Computer security applications conference*, pages 10–pp. IEEE, 2005.

[24] B. Tversky. Encoding processes in recognition and recall. *Cognitive Psychology*, 5(3):275–287, 1973.

[25] P. C. van Oorschot, S. Chiasson, and R. Biddle. Graphical passwords: Learning from the first twelve years. *ACM Computing Surveys (CSUR)*, 44(4), 2012.

[26] N. Wright, A. S. Patrick, and R. Biddle. Do You See Your Password? Applying Recognition to Textual Passwords. In *SOUPS '12: Proceedings of the Eighth Symposium on Usable Privacy and Security*. ACM, 2012.