INFINITE ALPHABET PASSWORDS
A Unified Model for a Class of Authentication Systems

Marcia Gibson, Marc Conrad, Carsten Maple
Institute for Research in Applicable Computing, University of Bedfordshire, Park Square, Luton, UK
marcia.gibson@beds.ac.uk, marc.conrad@beds.ac.uk, carsten.maple@beds.ac.uk

Keywords: User Authentication, Password, Infinite Alphabet, Formal Model.

Abstract: In the paper we propose a formal model for class of authentication systems termed, "Infinite Alphabet Password Systems" (IAPs). We define such systems as those that use a *character* set for the construction of the authentication token that is theoretically infinite, only bound by practical implementation restrictions. We find that the IAP architecture can feasibly be adapted for use in many real world situations, and may be implemented using a number of system architectures and cryptographic protocols. A security analysis is conducted on an implementation of the model that utilizes images for its underlying alphabet. As a result of the analysis we find that IAPs can offer security benefits over traditional alphanumeric password schemes. In particular some of the significant problems concerning phishing, pharming, replay, dictionary and offline brute force attacks are mitigated.

1 INTRODUCTION

It has been said that the user is often the weakest link in security (Sasse et al., 2001), and as a result when designing systems that we want to be effective in practice, we must consider the needs and tendencies of users.

A well documented example is the traditional password, which is a sequence of characters. The Universal Character Set defined by ISO/IEC standard 10646 (ISO, 2003) is used as the basis for many character encoding systems. For example, Unicode (The Unicode Consortium, 2009) corresponds with ISO/IEC 10646:2003 plus amendments 1-6. This contains "codepoints" comprising unique names and integer references for nearly 100,000 characters. When choosing passwords, individuals tend to select from a substantially smaller subset of this: those directly accessible via their input device. For instance, a standard United Kingdom keyboard can generate 103 printable characters by pressing a key or common key combinations.

In addition, users will often find recalling lengthy strings of high entropy (Shannon, 1948) difficult (Yan et al., 2004). When faced with the predicament of being unable to authenticate, they will often attempt to reduce the effort involved. Well known methods for this include, creating passwords based upon pre-existing semantic associations such as, pet's name or writing passwords down (Klein, 1990), sharing one or a handful of passwords between numerous accounts (Gaw and Felten, 2006), or selecting passwords that are shorter and less sporadically placed within the overall password space (Morris and Thompson, 1979).

Actions such as these increase ease of use, but also negate some of the intrinsic security benefits that may otherwise be offered. In the security research community there has been a concerted effort to develop novel authentication schemes that directly address this problem. One such category of schemes concerns the recognition or cued-recall of password symbols from a presented visual (Dhamija and Perrig, 2000), auditory (Gibson et al., 2009), or haptic (Kuber and Yu, 2006) alphabet.

In systems that support software alphabets, the potential alphabet size is larger than those which are feasibly available when generated using hardware peripherals. In particular, the use of optimally designed image and sound based alphabets, have been found to

aid memorability when compared to a traditional text-based counterpart (Dhamija and Perrig, 2000; Gibson et al., 2009). In theory, enhanced memorability should alleviate the need to devise workarounds that undermine security.

In this paper we devise a formal model for an authentication system architecture termed, "Infinite alphabet passwords" (IAP), which utilizes software based alphabets to enhance security. We envisage an IAP system as a system where there is no limitation in the number of available symbols that can be combined to form password sequences.

An example of a true infinite alphabet is the set of all images containing a blue n-gon (triangle, square, pentagon, ...). It should be clarified, that although the alphabet can be modeled as being infinite, in practice the alphabets are in fact *virtually* infinite due to the limitations of time and space. A feasible bound on alphabet size, is the *data width* (number of bits that can be used to represent a distinct symbol), as well as the capability of the system upon which the alphabet is generated to handle strings of a particular bit length. Hence a more realistic example of a virtually infinite alphabet might be the set of images indexed in the Google search engine, or the set of all top selling music singles, in a given location and time-frame. Although only virtually infinite, such an alphabet would be large enough for any practical purpose.

2 MODELING AN IAP

An infinite alphabet password system is a model where we allow the set of symbols that can be combined to fomulate passwords $A = \{a_1, a_2, \ldots\} \cong \mathbf{N}$ to be infinite.

In the following we assume that a user authenticates him or herself to a server S via a client node over an open network with the password data itself stored as a database entry on the server. However we note that in a more general setting depending on envisaged architecture the authentication may take place on a grid, client node, removable media device or as part of a cloud architecture.

As a consequence of the infinity of A, we let different servers carry different alphabets. That means if Σ denotes a system of servers with $S \in \Sigma$ and $A_S \subseteq A$ denotes an alphabet used for authentication for $S \in \Sigma$ then $A_S \cap A_T = \emptyset$ for any $T \in \Sigma$ with $S \neq T$. Note that because of $\mathbf{N} \times \mathbf{N} \cong \mathbf{N}$ this is possible also if we model $\Sigma \cong \mathbf{N}$ to be an infinite set of servers.

The enrolment process is accomplished by the user supplying to the system a unique identifier u. Example identifier formats include biometric data, hardware token, PIN or username. u is then stored via a one-way hash function as a record on $S \in \Sigma$.

The server S on receipt of the the user id u generates (randomly) a finite subset $A_{S,u} \subseteq A_S$, which becomes that user's password alphabet. This is sent to the client device for presentation, where the user selects from $A_{S,u}$ a predefined number of elements to form a password. This sequence is transmitted to S where it is also associated with u concluding the enrollment process.

At authentication to S, the user presents their unique identifier u. The server returns the finite set of symbols from the infinite alphabet namely $A_{S,u}$. This is presented to the user. No explicit confirmation is given as to whether the identifer data was correct. If an incorrect identifier $v \neq u$ is entered, a decoy set $A_{S,v}$ with $A_{S,v} \cap A_{S,u} = \emptyset$ is presented.

During key replacement, a new finite subset that does not contain any of the users previous password symbols is created and associated with the account (not problematic due to the infinity of A). From this, the user may select a new password sequence as described in enrollment.

In the remainder, we assume that the $A_{S,u}$ is presented to the user in authentication stages $A_{S,u}(1)$, $A_{S,u}(2)$, ... with $\bigcup_i A_{S,u}(i) = A_{S,u}$. In the stage m the user is presented with the set $A_{S,u}(m)$ and subsequently selects a number of elements from this before being presented with $A_{S,u}(m+1)$. We also assume that the placement of symbols $A_{S,u}(m)$ is randomized at each presentation.

2.1 Optional Enhancement: Injective Password Sequence

We want the user to be able to authenticate via recognition of their password symbols. If a symbol appears in more than one finite presentation set $A_{S,u}(i)$ the user also has to memorize the sequence in which they must select their password symbols. This needlessly increases cognitive load (as well as the probability that insecure coping strategies will be adopted). For this reason, we can optionally require that $A_{S,u}(i) \cap A_{S,u}(j) = \emptyset$ for any $i \neq j$. In a conventional finite alphabet this would reduce security, because the average search space for a successful brute force attack is considerably decreased. In an IAP system, the search space is infinite, so there is no loss of security.

2.2 Optional Enhancement: Error Feedback and Recovery

In traditional password systems, users are given minimal help in recovering from input errors. This is be-

cause in this case, an error is not simply an error, but may be an indication of an attempt to break into the system. In an IAP system, we have the option of providing feedback to the authentic user without inadvertently providing clues to an attacker.

We illustrate this with the following example. Assume that the client's password is $(a_{\iota(1)},\ldots,a_{\iota(n)})$ with n even and $a_{\iota(2m-1)},a_{\iota(2m)} \in A_{S,u}(m)$ for $1 \leq m \leq n/2$, i.e in every authentication stage the user selects two password letters. When the selection (b,c) in stage m is correct, i.e. $(b,c) = (a_{\iota(2m-1)},a_{\iota(2m)})$ the client is exposed to the set $A_{S,u}(m+1)$ unless $2m = n$ and access is granted. When the selection is incorrect, the user is exposed to a different, unrelated set $B_{S,u}(m,b,c) \subseteq A$. In the model we may assume that all sets $B_{S,u}(m,b,c)$ and $A_{S,u}(m)$ are disjunct to prevent an intersection attack. On presentation of $B_{S,u}(m,b,c)$, the authentic user may notice that the presented set $B_{S,u}(m,b,c)$ differs from the expected set $A_{S,u}(m)$. At this point, they may signal to the system that they cannot find their password element, for instance via a "reset" option. This action will return them to the first stage $A_{S,u}(1)$ hence allowing them to retry their authentication attempt from the beginning. The user who does not recognize the disjunctive nature of $A_{S,u}(m)$ and $B_{S,u}(m,b,c)$ may be identified as a potential intruder and be implicitly excluded from the system.

2.3 Optional Enhancement: Time Localization

We may also design the alphabet A such that it consists of an infinite number of pairwise disjunct equivalence classes $A(1)$, $A(2)$, $A(3)$, ... of infinite size such that $a \sim_{A(k)} a'$ when a and a' have a same, or obviously related semantical meaning. The user when requested to enter the password, must be able to identify elements of the same equivalence class and distinguish elements of different equivalence classes. Examples of such an equivalence class may be images where the colors have been distorted, for example in a similar way to the images described in (Hayashi et al., 2008), or those that relate to clear concepts such as "tree" or "house".

The alphabet is then the set of equivalence classes and in the authentication process the client is exposed to a member of each equivalence class. The member chosen from each equivalence class changes from session to session.

An illustrative example for such a mechanism using equivalence classes might be dogs shown in different positions (sitting, standing, jumping). Large numbers of different images of a particular dog can be generated easily by making a movie of a dog and using the stills. The password alphabet $A(S,u)$ would then be a set of different equivalence classes, each of which would feature a different dog. A possible intruder who wants to guess the password must then, for successful intrusion, know that the key is given in the dog itself, independent of the behavior that is shown for each dog. In addition, a nonce can be added, for example in the form of a watermark in order to enhance security further.

3 SECURITY ANALYSIS

In this section we envisage a feasibly implementable adaptation of the IAP model, where the alphabet A is no longer *infinite*, but instead only *very large*. A and all related subsets are hence also *finite*, as are the number of S elements contained within Σ.

In order to keep things as simple as possible, we use a conservative example and imagine that A is composed of bitmaps (i.e. no compression). Each image is 150×150 pixels with 24 bit color (RGB 0-255). The total number of unique images is hence: $150^2 \times 256^3 = 377,487,360,000$. Not all of these would be suitable for use as alphabet elements, as many would not be perceivably different for users. For this example, we will specify that 0.0001% of the elements are easy to distinguish, and hence suitable for inclusion in A. Therefore, A consists of 37,748,736 images.

We distribute A equally over 10,000 Web servers, providing on average around 3,770 images to each. Each user is given a personal finite subset of 50 images. In this conservative example, we therefore support around 75 users per server. Given the dimensions and bit depth, each image would require approximately 66 KB of storage space, and for each one we also use 4 additional semantic equivalence class images. This would require a total of around 18,850 unique images to be stored and 1.1 GB of storage space per server.

We assume that each user account is protected by a username, coupled with a sequence of 5 IAP password elements chosen from a distracter set of 50 and that presentation takes place over 5 screens. Finally, although we wish our model to be independent of any particular cryptographic scheme, this directly affects security against some attacks. For this reason we consider that passwords undergo a one-way hash function before being stored on the server, and that the set A is stored in unencrypted form, but that encrypted pointers are used to specify a mapping between a user account and a corresponding finite subset.

We consider the following typical scenarios of attacks upon this adapted IAP system, where all three optional enhancements namely, injective password sequences, error feedback and recovery, and time localization have been incorporated. Here, Attacker Mallory attempts to gain access to user Alice's IAP protected account.

3.1 Tricking the User into Verbally Communicating the Key

Mallory knows Alice's username and fools her into revealing her password sequence. The chance that this attack would be successful is a property of the communicability of Alice's individual alphabet elements. If we imagine that they are images with concrete semantic associations, such as, "child" or, "dog", then the system is neither stronger nor weaker than a traditional password in this sense.

3.2 Finding a Written Record of the Password

Attacker Mallory gains access to Alice's workstation where the username and a description of the password symbols has been sketched or written down.

Again, the chance that this will happen is directly related to the intrinsic memorability and communicability of the password elements. In our example, the user may be less likely to write their password down. If they do however, then the system is again neither stronger or weaker than a text-based counterpart.

3.3 Password Prediction

Attacker Mallory comes to know Alices interests and preferences along with her username. He proceeds to enter the username at the interface and the system responds with the first 10 images from her password alphabet. Mallory then attempts to predict Alices chosen images given his knowledge of her taste, in an attempt to gain access.

A system offering an alphabet with pre-existing semantic associations, or with some images being more visually appealing than others would be prone to entry using this method.

This is confirmed in (Davis et al., 2004), where a visual authentication system utilizing photographs of faces as the alphabet was implemented. Here, users tended to select attractive faces over less attractive ones, or faces of people belonging to the same racial group as themselves. As way of countermeasure, it might be possible to learn from the user at enrolment their preferences and interests, in order to minimize the predictability of choices. This should not be problematic given the large corpus of images available. However some may see this as an infringement of privacy.

3.4 Phishing and Pharming

Here Alice is either redirected to, or receives an email enticing her to view an intruder Web site masquerading as the genuine log in page for the IAP system. On loading the imposter site, Alice is prompted for her username and on entry is presented with a set of challenge images. Here, the challenge set presented would be incorrect, and Alice would be unable to proceed with authentication, thereby securing her password information.

3.5 Hard or Software Keylogger

In this attack Alice's inputs to the access device are recorded and sent to Mallory. Since Alice will select her password symbols using a pointing device, Mallory would receive information that Alice has clicked, and possibly the co-ordinates of where she has clicked, but not information about what she has clicked. Element placements are shuffled between log in sessions, therefore her password information is secured.

3.6 Malicious Screen Capture

In this attack a sequence of images detailing Alice's interactions with the IAP interface is viewed remotely by Mallory who later attempts to reenact the process. Image based alphabets are particularly vulnerable to this type of attack. It would therefore be necessary to incorporate a mechanism to delineate or disguise interactions with password elements during selection. An example of such a scheme is the random cursor matrix system described in, (Boit et al., 2009).

3.7 Timing Attack

In this attack, Mallory gains physical access to Alice's access device where he enters Alice's username and is presented with the first presentation screen of her challenge set. Mallory proceeds to guess at random the first element and measures the length of time it takes for the images in the subsequent challenge set to load in full. Mallory then uses the recovery option to return to the first screen where he selects a new symbol and repeats the process. On happening upon a second screen that loads the symbols more quickly than others, Mallory assumes that the symbol set has

been accessed by Alice before because it is cached locally in her browser. The symbol that was clicked in order to load the screen was hence a correct password symbol.

There are two options available as countermeasure. Firstly, at enrolment the system may download every image in the user's finite set, to the client machine, whereby it is cached. Or secondly, to ensure that no symbol is cached locally. The second option seems most sensible, as caching all symbols on the local machine would be ineffective for those users who happen to clear their browser cache regularly, and may create new vulnerabilities to attack via the client.

3.8 Brute Force Attack (Online)

Mallory knows Alice's username and enters it at the interface. The system responds by returning the first subset of her individual alphabet. From here Mallory selects symbols at random until the correct password sequence is obtained.

Permutations in Alice's alphabet is q^r, where q is the number of symbols contained in the alphabet and r is the password length.

In our example, $q = 10$ and $r = 5$ as users choose symbols over five screens, each offering a choice of 10 images. Giving a total of 100,000 permutations and means the average brute force attack would elicit the correct sequence in 100,000/2 attempts. For practical purposes we can mitigate this risk by blocking authentication after a given number of attempts is exceeded.

This approach does not rule out a "low and slow" attack, where Mallory circumvents the lock out policy by distributing his guesses over a number of user accounts. We must for this reason, ensure that the number of accounts the system supports, and upon which Mallory can make a password guess, is substantially lower than the average number of guesses required. In the example system, we support 75 accounts. We imagine that each account allows up to 3 authentication attempts before detection and that Mallory does not wish to reach this limit. In this scenario, he is able to make 150 guesses – far lower than the average 5,000 required.

3.9 Brute Force Attack (Offline)

Attacker Mallory gains access to Alice's password hash and the symbols stored in A (which are in the clear). Let us assume that in this instance, the key that is required to reveal the pointers to Alice's finite subset symbols is derived from the password symbols themselves (which Mallory does not yet know). As a result, in order to find the password, Mallory must compute all permutations of A and pass these to the same hash function used to encrypt the password.

A itself contains 18,850 unique images, although they each belong to an equivalence class. We hence compute permutations based upon the number of equivalence classes, of which we have 3,770. We prevent the user from selecting the same images more than once, therefore the maximum possible number of permutations Mallory can create based upon the number of sounds in the list is $q!/(q-r)! \simeq 7.595 \times 10^{17}$ where $q = 3,770$ and $r = 5$. This is roughly equivalent to the number of *combinations* possible from a 9 character traditional password over an alphabet of 103 letters (i.e $\simeq 1.304 \times 10^{18}$).

3.10 Dictionary Attack

Mallory uses a list of historically common password sequences and submits these to the system in an attempt to elicit access to Alice's account. The principal requisite for Mallory's success is that non-standard frequencies must exist in the passwords selected (i.e. there needs to exist common passwords). The strength of an IAP against an attack of this type is a result of the relative popularity of the images used as alphabet elements.

The risk may be mitigated by splitting up the challenge set into subsets. During sequential presentation the user must be shown at least one popular image in each presentation set in order to create a password made up of completely popular elements. In a traditional scheme, all of the popular passwords are available to the user for selection all of the time. Furthermore, an important aspect of our model is that each server can be populated with an individual symbol alphabet A. The result is that we could do away with the practice of attackers employing standard password cracking dictionaries to gain access, as any dictionary created would only be of use against the system upon which it was originally generated, because the letters used to create a password sequence therein, would not be available as password elements elsewhere.

3.11 Replay Attack

In this attack, Mallory eavesdrops the connection between Alice's access device and the server and gathers message digests containing the identity of Alice's password symbols. Here, a nonce value is used as a wrapper to the time-localized password, which is encoded using a one-way hash. If Mallory attempts to replay the session back to the server at a later time, the

nonce is incorrect and the request denied. If Mallory learns to predict the nonce and then creates a counterfeit, he would still be unable to replay the data until the server expects a message using the same equivalence class image.

4 DISADVANTAGES OF IAPS

The drawbacks are the storage and bandwidth requirement. As a solution, we could populate a system with compressed images or sounds in order to reduce storage consumption. Another option in the case of images might be to use vectors or fractals. This would allow for better upwards scalability in systems that support a large number of accounts.

A third option is that we could weaken the condition that the user's personalized alphabets are disjunct for any two users. It should be possible to implement the IAP architecture and only require a low probability that two users share a given symbol in their mutual alphabets – This might have ramifications elsewhere, and hence requires further research.

5 CONCLUSION

We modeled passwords as utilizing an infinite alphabet, allowing us to devise an optimized architecture upon which image and sound based authentication schemes can be based. We give an example of a feasible implementation of the IAP model, using images as a password alphabet, as a result we find that although modeled on infinity, the architecture can be feasibly adapted for use in many real world scenarios. The envisaged system underwent a security analysis, wherein it was found that depending upon the nature of the alphabet used, the system is at least as strong as a traditional alphanumeric counterpart against social engineering and online brute force attacks and more secure against replay, keylogging, phishing, pharming, offline brute force and dictionary attacks. However, when image based alphabets are implemented, the model is weaker than traditional passwords against the threat of remote screen capture. It is therefore *essential* that any image based IAP system also incorporate countermeasures to mitigate this risk.

The IAP model was developed with flexibility in mind. For this reason, it should be implementable over a number of preferred architectures and cryptographic protocols. It is hoped that the model may prove useful to those considering future implementations of alternative authentication schemes.

REFERENCES

Boit, A., Geimer, T., and Loviscach, J. (2009). A random cursor matrix to hide graphical password input. In *SIGGRAPH '09: SIGGRAPH '09: Posters*, pages 1–1, New York, NY, USA. ACM.

Davis, D., Monrose, F., and Reiter, M. K. (2004). On user choice in graphical password schemes. In *SSYM'04: Proceedings of the 13th conference on USENIX Security Symposium*, pages 11–11, Berkeley, CA, USA. USENIX Association.

Dhamija, R. and Perrig, A. (2000). Déjà vu: A user study using images for authentication. In *Proceedings of USENIX Security Symposium*, pages 45–58, Denver, Colorado.

Gaw, S. and Felten, E. W. (2006). Password management strategies for online accounts. In *SOUPS '06: Proceedings of the second symposium on Usable privacy and security*, pages 44–55, New York, NY, USA. ACM Press.

Gibson, M., Renaud, K., Conrad, M., and Maple, C. (2009). Musipass: authenticating me softly with "my" song. In *NSPW '09: Proceedings of the 2009 workshop on New security paradigms*, pages 85–100, New York, NY, USA. ACM.

Hayashi, E., Dhamija, R., Christin, N., and Perrig, A. (2008). Use your illusion: secure authentication usable anywhere. In *SOUPS '08: Proceedings of the 4th symposium on Usable privacy and security*, pages 35–45, New York, NY, USA. ACM.

ISO (2003). ISO/IEC 10646:2003 Information technology – Universal Multiple-Octet Coded Character Set (UCS).

Klein, D. V. (1990). "foiling the cracker" – A survey of, and improvements to, password security. In *Proceedings of the second USENIX Workshop on Security*, pages 5–14.

Kuber, R. and Yu, W. (2006). Authentication using tactile feedback. In *HCI Engage 2006, Interactive experiences*.

Morris, R. and Thompson, K. (1979). Password security: A case history. *Communications of the ACM*, 22(11):594–597.

Sasse, M. A., Brostoff, S., and Weirich, D. (2001). Transforming the 'weakest link' — a human/computer interaction approach to usable and effective security. *BT Technology Journal*, 19(3):122–131.

Shannon, C. (1948). A mathematical theory of communication. *The Bell System Technical Journal*, 27:379–423.

The Unicode Consortium (2009). The Unicode Standard, version 5.2.0. Mountain View, CA. ISBN 978-1-936213-00-9. http://www.unicode.org/versions/Unicode5.2.0/.

Yan, J., Blackwell, A., Anderson, R., and Grant, A. (2004). Password memorability and security: Empirical results. *IEEE Security and Privacy*, 2(5):25–31.