ISSN 1991-1696

SAIEE

Africa Research Journal

Research Journal of the South African Institute of Electrical Engineers

Incorporating the SAIEE Transactions

www.saiee.org.za

June 2014
Volume 105 No. 2

SAIEE AFRICA RESEARCH JOURNAL

(SAIEE FOUNDED JUNE 1909 INCORPORATED DECEMBER 1909)
AN OFFICIAL JOURNAL OF THE INSTITUTE

ISSN 1991-1696

Secretary and Head Office
Mrs Gerda Geyer
South African Institute for Electrical Engineers (SAIEE)
PO Box 751253, Gardenview, 2047, South Africa
Tel: (27-11) 487-3003
Fax: (27-11) 487-3002
E-mail: researchjournal@saiee.org.za

EDITORS AND REVIEWERS

Published by
SAIEE Publications (Pty) Ltd, PO Box 751253, Gardenview, 2047,
Tel. (27-11) 487-3003, Fax. (27-11) 487-3002,
E-mail: SAIEEPublications@saiee.org.za

Additional reviewers are approached as necessary

ARTICLES SUBMITTED TO THE SAIEE AFRICA RESEARCH JOURNAL ARE FULLY PEER REVIEWED PRIOR TO ACCEPTANCE FOR PUBLICATION

The following organisations have listed SAIEE Africa Research Journal for abstraction purposes:

INSPEC (The Institution of Electrical Engineers, London); 'The Engineering Index' (Engineering Information Inc.)

VOL 105 No 2
June 2014

SAIEE Africa Research Journal

GUEST EDITORIAL
INFORMATION SECURITY SOUTH AFRICA (ISSA) 2013

This special issue of the SAIEE Africa Research Journal is devoted to selected papers from the Information Security South Africa (ISSA) 2013 Conference which was held in Johannesburg, South Africa from 14 to 16 August 2013. The aim of the annual ISSA conference is to provide information security practitioners and researchers, from all over the globe, an opportunity to share their knowledge and research results with their peers. The 2013 conference focused on a wide spectrum of aspects in the information security domain. The fact that issues covering the functional, business, managerial, human, theoretical and technological aspects were addressed emphasizes the wide multi-disciplinary nature of modern-day information security.

With the assistance of the original reviewers, twelve conference papers that received good overall reviews were identified. At the conference, I attended the presentation of each of these twelve papers and based on the reviewer reports and the presentations I selected six of these papers for possible publication in this Special Edition. The authors of these six selected papers were asked to rework their papers by expanding and/or further formalizing the research conducted. Each of these papers was subsequently reviewed again by a minimum of four international subject specialists. In some cases, where conflicting reviews were received, further reviews were requested. In some cases five reviews were requested to enable objective and quality decisions to be made. In all cases the reviews were conducted by members of the Technical Committee (TC) 11 of the International Federation of Information Processing (IFIP) or some subject experts suggested by them. Thus, in all cases the reviews were conducted by reputable subject experts and enough reviews were received to make a confident decision as well as to improve the relevant papers.

In the end three papers were selected to get published in this Special Edition after the reviewer comments were attended to satisfactorily. The three papers cover various aspects of information security. The one paper identifies some new ideas in the user authentication arena; one paper focuses on the very important area of IT forensics and the third paper on the ever-present cloud and associated issues. This Special Edition includes three very diverse papers in the discipline of information security, giving a true reflection of the multi-disciplinary nature field of study.

Lastly, I would like to express my appreciation to IEEE Xplore, who originally published the ISSA conference papers, for granting permission that these reworked papers can be published in this Special Edition.

Prof Rossouw von Solms
Guest Editor

NOTES

CONTEMPLATING SKILL-BASED AUTHENTICATION

Karen Renaud, Joe Maguire* and Johan van Niekerk† and Demetris Kennes‡

* *School of Computing Science, University of Glasgow, United Kingdom*
E-mail: karen.renaud,joseph.maguire@glasgow.ac.uk
† *School of ICT, Nelson Mandela Metropolitan University, South Africa*
E-mail: Johan.VanNiekerk@nmmu.ac.za
‡ *Enterprise Risk Services, Deloitte Limited, Limassol, Cyprus.*
E-mail: dkennes@deloitte.com

Abstract: Humans develop skills as they go through their lives: some are fairly common, such as reading, but others are developed to maximise employment opportunities. These skills develop over a long period of time and are much rarer. Here we consider whether we can exploit this reality in the security arena, specifically to achieve a stronger form of authentication. Authentication has traditionally been performed based on what users *know*, *hold* or *are*. The first is the most popular, in the form of the password. This is often referred to as "knowledge-based" authentication. Yet, rigorously following guidelines for password creation produces forgettable gibberish and nonsense strings, not knowledge. Nonsense is hard to remember and users engage in a number of coping strategies to ameliorate this, and these tend to weaken the authenticator. It would be beneficial to find a way of reducing this memorial load, to identify a more usable mechanism. This is hard: usually reducing the memorial load also makes the secret easier to guess. The challenge is in finding a way to reduce memory load while holding the line as far as strength is concerned. Here we contemplate exploiting recognition of artefacts resulting from experts practicing their craft: "skill-based" authentication. This should reduce the memorial load and effort, but also, crucially, make it harder for a random intruder to replicate. We report on how we trialled SNIPPET, a prototype of an authentication mechanism that relied on an expert programmer identifying his/her own code snippets from successive challenge sets. We found that our participants were all able to identify their own code snippets and that other participants were unable to guess these, even when they observed the legitimate person authenticating beforehand. These findings are not conclusive given the small number of participants but they do show promise and suggest that this is an area worth pursuing. We conclude by returning to the three NIST-identified forms of authentication and consider how SNIPPET can be positioned within the general authentication arena.

Key words: Authentication, Knowledge, Skills

1. INTRODUCTION

The PIN challenge issued by the ubiquitous ATM (Automatic Teller Machine) is a good example of an authentication mechanism encountered by the man and woman in the street in the course of their everyday lives. There is no report of complaints about having to remember the secret PIN when ATM machines were deployed in the 1960s. This is probably because in those days people only had to remember one or two PINs, not the multiple PINs and passwords they have to remember today.

As computers permeated all aspects of business life, the password was the obvious choice for restricing access, given the fact that the end-user had probably had experience of an ATM machine and could thus rely on a prior understanding of the concept. Fernando J. Corbató, the project leader behind one of the first systems to use passwords, Compatible Time Sharing System (CTSS) [1], explained that although passwords seemed theoretically strong, in practice many problems emerged. People routinely compromised security by choosing weak passwords [2], and by writing them down and sharing them [3]. A lot of this behaviour was driven by the fact that they had too many passwords [4, 5], and because they had previously forgotten passwords and had no desire to repeat the experience. Blaming the users is the natural response, and the obvious next step is to try to persuade or coerce them into abandoning these behaviours. This, while intuitively the right course of action, is bound to fail, since it does not eliminate the cause of the behavioural effect: users don't want the inconvenience of a forgotten password (Figure 1). If we can reduce the prevalence of the cause, the resulting uncceptable behaviours might be less likely to occur.

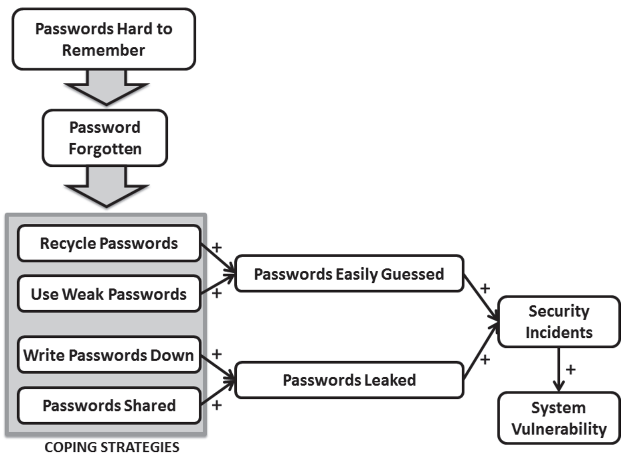

Figure 1: Coping Behaviours and Antecedents

Based on "SNIPPET: Genuine knowledge-based authentication", by K. Renaud, J. Maguire, J. van Niekerk and D. Kennes which appeared in the Proceedings of Information Security South African (ISSA) 2013, Johannesburg, 14 to 16 August 2013.

Perhaps in response to the wide-spread issues related to traditional secret-based mechanisms, Apple recently released an iPhone with a fingerprint sensor that essentially introduced biometrics into the mainstream consumer market. The device sold an estimated 9 million in its first few days after release [6]. Pankati [7] predicted at the turn of the century that biometric-based authentication was the future. He argued that since tokens were easily misplaced and it was easy to forget passwords, the only future direction for authentication was the dependable and indisputable biometric [7]. It is interesting that Apple appears to have come to the same conclusion, albeit 13 years later.

There are naturally concerns about the use of such an authentication mechanism. The approach is easily fooled by fake fingers [8]. Moreover, it appears to dissuade device sharing which is something many phone owners want to be able to do [9]. It is interesting that Apple has decided to include such a relatively novel authentication mechanism in their mainstream products. The convenience of access control for the device owner is probably considered a selling point although the recent revelations by Edward Snowden [10] might well give iPhone owners pause with respect to the potential destination of their fingerprint template [11, 12]. There is, however, a certain clarity in the choice of this authentication solution. In theory, the mechanism relies on both the owner and device being co-present and one can readily see the attraction and simplicity of such a guarantee as far as security is concerned.

Despite Apple's recent innovation, however, the reality is that biometric-based authentication remains relatively novel and passwords not only persist, they reign supreme, as the *de facto* authentication approach across the globe. In effect, passwords have become the default authentication solution for almost every context and user. This brings us back to the apparently intractable problem related to passwords: the tension between strength and memorability. Here we offer a way of ameliorating this problem.

The rest of the paper is structured as follows. Section 2. explores the concept of "What you Know" authentication. Section 3. explores the idea of a genuine knowledge-based authentication mechanism, leading to the concept of "skill-based" authentication. Section 4. reports on a survey of programmers to determine whether they thought they would be able to identify their own and others' programming code. The survey results suggested that empirical verification would be beneficial. Section 5. reports on a pilot study we carried out to test a "skill-based" authentication mechanism. Section 6. reconsiders authentication in general and positions our mechanism, SNIPPET, within the authentication arena. Section 7. concludes.

2. "WHAT YOU KNOW" AUTHENTICATION

"What you know" authentication is the process of confirming a claimed identity through knowledge of a secret, one known only to you and the authenticating party. Since it is a secret, individuals are advised to memorise it and not to record or share it. The secret itself could be a public event or record, but the use thereof must not be revealed.

The alphanumeric password is the best known implementation of "what you know" authentication. There are two reasons for this:

1. the concept of passwords is one which is centuries old and is easily understood by both users and developers; and
2. the interaction mechanism, i.e. the keyboard, is over a century old and one can easily enter passwords without additional training or expense.

This made passwords the authentication mechanism of choice for early systems, such as CTSS [1], and operating system designers such as Ken Thompson and Dennis Ritchie.

The problems with passwords emerged soon after their initial deployment. They immediately proved difficult to use and remember [13]. The situation has barely improved as technology has advanced. If anything, as the world becomes increasingly connected, the ubiquitous use of passwords becomes even more problematical. News stories detailing the problems caused by the improper use of passwords are not a rare occurrence. The Federal Trade Commission, for example, has recently taken legal action against Wyndham Hotels after the organisation failed to properly protect the financial information of 500,000 customers, resulting in damages of $10.6 million [14]. The organisation generated weak and simple passwords that were compromised by attackers and allowed them to install software to capture information.

The use of simple passwords is not particularly surprising as users will create simple passwords to avoid the inconvenience of not be able to complete a task, since they have probably forgotten a password previously and do not want to repeat the experience [15]. The following excerpt, extracted from a complaint submitted by the Federal Trade Commission, offers evidence of the use of simple passwords in the aforementioned case, as follows:

> "For example, to allow remote access to a hotel's property management system, which was developed by software developer Micros Systems, Inc,. Defendants used the phrase "micros" as both the user ID and password"
> **Federal Trade Commission Compliant [14, p. 11]**

The use of such simple strings for the convenience of a few individuals led to dramatic inconvenience for 500,000 paying guests. A great deal of expense, in terms of time and money, was spent rectifying the problems caused by this irresponsible authorisation mechanism.

However, passwords that are difficult to remember also incur costs for organisations. The estimated cost of password bureaucracy, such as replacement and recovery, is an estimated $17 per call [16]. Moreover, an estimated 30% of call volumes are associated with passwords [16]. Consequently, not only is there a cost associated with each call, there are also a considerable number of calls to cope with.

Despite these problems, the vast majority of authentication in 2013 falls into the "what you know" category. This is often termed *knowledge-based authentication*, which seems intuitively correct. This seems to be based on the assumption that there is a natural mapping, allowing one to substitute "what you know" with the word "knowledge". Actually we are going to argue that this is misguided, that the terms are not as interchangeable as they seem. In fact "what you know" may, over time, progress into knowledge, depending on its nature, but such a progression is by no means guaranteed. To support this argument we need to examine the distinction between data, information and knowledge (Figure 2).

- *Data:* Data is simply data: no use to anyone until someone provides the context. So, for example, consider the number: 2.5, a simple piece of data. There is no way of knowing what that number refers to.
- *Information:* If we add context and explain that this is the number used to convert a measurement from inches to centimetres, the data has become information, because it now has meaning. It is not yet knowledge, however.
- *Knowledge:* Knowledge is defined by the Oxford dictionary as: "*the theoretical or practical understanding of a subject*". In other words, knowledge implies an understanding of how to use the information to solve some problem. If one is given the dimensions of a room in inches and asked to calculate the area of the room in cm^2, the information just provided would be applied in order to solve the problem. The person would also have to know how to work out area using the width and breadth and know how to multiply the dimensions by the conversion value to arrive at the correct result. This implies an understanding of how to use the information, and success suggests that you do indeed possess that knowledge.

Knowledge and skills take time to develop, and this process cannot be short-circuited [17]. The benefit is that knowledge and skills are not easily disrupted. The nature

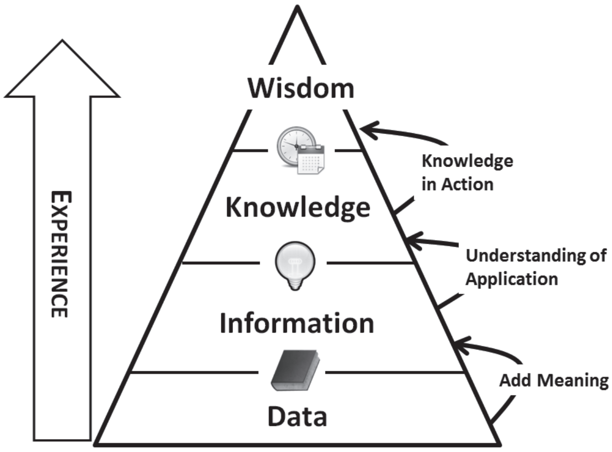

Figure 2: Data, Information, Knowledge, Wisdom (DIKW) Pyramid

of the knowledge and skill acquisition process seems to make a durable footprint on the user's mind that does not easily decay, even with age, especially when learnt before retirement [18]. Moreover, retrieving the knowledge requires less effort than recalling a nonsense data string effortfully memorised and possibly forgotten. Nonsense is forgotten because the brain is economical and performs neural pruning on networks that are not deemed essential [19]. The more interesting and stimulating something is, the more easily it will be remembered. Nonsense is neither stimulating nor interesting, and is deliberately pruned.

It is also of interest to note that the above mentioned "levels" as one progresses from data to knowledge also, to a certain extent, map to the first three levels of Bloom's well-known taxonomy of the cognitive domain [20]. The following lists the first three levels as presented by [20], and briefly shows how these levels relate to the distinction between data, information, and knowledge.

- *Remember:* This is the lowest level of cognition. Remember is the ability to *retrieve* relevant facts from memory but does not include the ability to relate the retrieved facts to a specific context.
- *Understand:* If we add context to remembered data a person has the ability to understand the data, "construct the meaning of instructional messages" [20, pp. 30], but does not necessarily have the ability to apply it correctly.
- *Apply:* The third level of the cognitive domain is being able use the information correctly in a given situation or context. This level of cognition thus clearly requires the person to have *knowledge*, as defined above.

Now consider authentication. Here is some advice given by CERT [21] for choosing a password:

- Don't use passwords that are based on personal information that can be easily accessed or guessed.
- Don't use words that can be found in any dictionary of any language.
- Develop a mnemonic for remembering complex passwords.
- Use both lowercase and capital letters.
- Use a combination of letters, numbers, and special characters.
- Use passphrases when you can.
- Use different passwords on different systems.

A password chosen according to these guidelines is more akin to data than it is to knowledge. If a password has meaning, it has become information. If it is information then attacks become easier to carry out. Users use information instead of data as passwords so that the password will not be forgotten, Such an information-based password has meaning, usually something related to the user him or herself. This action potentially weakens the password since an attacker who knows the user will be more likely to be able to guess it. Figure 3 shows how the drive for strong passwords conflicts with users' motivation to choose memorable and meaningful passwords.

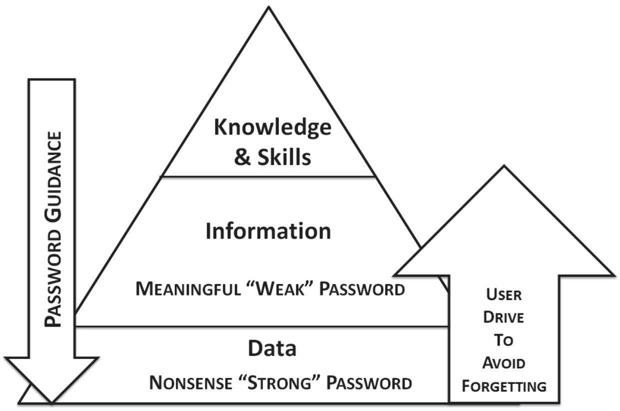

Figure 3: Passwords Positioned within the DIKW Pyramid

Thus a more realistic moniker for current recommended usage of "what you know" authentication would be "nonsense-based" authentication. This begs the question: what would actual knowledge-based authentication actually look like? Some pertinent aspects immediately become evident and will be referred to here as the *constraints* of genuine knowledge-based authentication, what we will call *skill-based authentication* (Figure 4).

C_1 *Appropriate Elicitation:* We have to test someone's skills or understanding of an area, which is much harder than asking them to produce an alphanumeric string. Moreover, testing this kind of knowledge requires provision of context, since knowledge is always applied within a particular context. Such context should not constitute a cue to any would-be intruder.

C_2 *Soundness:* It should not be possible for another expert in the area to authenticate: we have to ensure that the mechanism authenticates only the legitimate expert user [22].

C_3 *Cost-Benefit Balance:* It must be possible for a user to demonstrate this knowledge quickly and easily, so that authentication does not become too time-consuming or inconvenient [23].

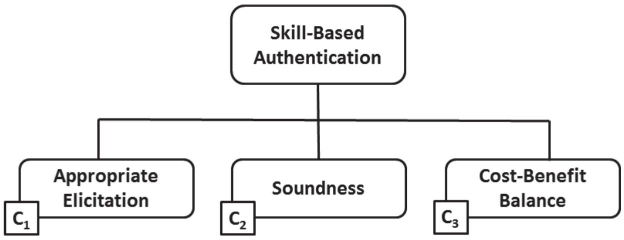

Figure 4: Constraints of S-Based Authentication

There are clear challenges inherent in testing genuine knowledge that meets these constraints. This kind of authentication is a relatively unexplored category, which is understandable given these constraints. The following section explores the issue of testing knowledge and skills in an authentication setting.

3. MOVING UP THE PYRAMID

Generally, one can test "what you know" in one of three ways: *recall, cued-recall* or *recognition* [24]. All of these require some memorial effort with effort decreasing from recall, to cued-recall to recognition. Testing recall-based memory offers the recaller no assistance: they are required to remember the item unainded. This becomes increasingly difficult as users age [25]. Moreover, since knowledge is applied in context, pure recall-based testing is unsuitable since it does not satisfy the first constraint.

Cued-recall mechanisms provide cues to help the user to recall the authentication secret. The provision of cues in this setting, while essential in testing knowledge, is problematical since cues have to assist the legitimate user but not any random intruder who happens to be skilled in the same area.

An example of the use of cues in authentication is the Cueblot mechanism [26] which displays an inkblot-like image to trigger the user's memory when they have to authenticate. Since the cueblot is sufficiently abstract it does not act as a cue for other users, but only for the legitimate user. What this paper confirms is the difficulty of providing a legitimate user with a cue that will not make sense to another user. The cueblot cue does not really test expert knowledge, however, so this particular technique will not be useful in implementing genuine knowledge-based authentication.

Another example of a cued-recall mechanism is Zviran's associative passwords which probe a user's personal experience [27]. This quiz-based approach extracts several pieces of knowledge from the user at enrolment. The individual is presented a series of *fact-based* and *opinion-based* questions. A fact-based question would be 'What was the first school you attended?', while an opinion-based question would be 'What is your favourite film?'. The problem with this mechanism is that it is too time-consuming at authentication, thus not satisfying the third constraint: adequate balance of cost and benefit. However, this example exploits an aspect of skilled practice that will be very useful to this research: the *experience* of the user. We might be able to exploit this to meet constraint number two: distinguishing different experts from each other, since every expert has different life experiences.

Cued-recall authentication provides the essential context the skilled user needs to demonstrate possession of skills but it does so in a way that makes authentication time-consuming, and, as such, is probably infeasible. We will thus explore the last remaining possibility: relying on recognition.

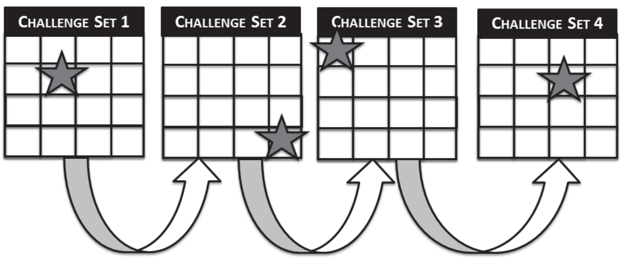

Figure 5: Authentication - Four Challenge Sets

Recognition-based mechanisms most often display grids of images and require the user to click on their own image from the challenge set (Figure 5). A number of these have been proposed [28–31] asking users to identify faces, representational or abstract images from challenge sets. Recognising is easiest for users, since all they have to do is click on their own secret image in order to authenticate: it is cognitively the least demanding mechanism. It meets the first constraint since it provides context. It also comes closer to meeting the third constraint since it takes less time than a cued-recall mechanism.

The second constraint is harder to meet. Most recognition-based authentication mechanisms do not personalise the images used by the mechanism, using the same images for the entire user population. Unfortunately, when users are allowed to choose from a common dictionary their choices are predictable [28, 32]. Perhaps they are still trying to find meaning in their secrets so as to prevent the secret from being forgotten.

How can we ensure that only the legitimate user can recognise and identify the correct image in the challenge set? Here we deploy the concept that Zviran [27] highlighted: the experience of the user. Experts often produce artefacts as they practice their skills. If we test recognition of these artifacts, rather than mere expert knowledge, we ensure that the user possesses both the skills and the actual experience. They should be able to remember that they engaged in a practice that produced the artifact. It must be admitted that not all skills leave artefacts: medical doctors, for example, do not necessarily produce artefacts. Other professions, though, do: examples include programmers and artisans such as carpenters and builders.

The second constraint, soundness, can be split into further sub-categories. Renaud and De Angeli [33] argue that the security (soundness) of an authentication mechanism means that it will be *unpredictable, abundant* and *undisclosed*. The first two seem to be focused primarily on the strength that comes from the size of the dictionary a secret authenticator is drawn from, which refers to the unpredictability of the mechanism. The third appears to be more related to the obscurity of the mechanism than the dictionary size: the need to keep knowledge of the secret from others. Moreover, this particular list of requirements does not include the need for the knowledge to be easily memorable, which undeniably contributes towards its soundness as an authenticator. Hence soundness must incorporate the following (Figure 6):

C_2(a) *Undisclosed:* The *non-availability* of the authenticator can be assured in two ways. The first is secrecy, ensuring the imposter does not gain knowledge of the authenticator. The second is security, keeping the autheticator out of the reach of would-be imposters even though it may not be secret. If the secrecy technique is used the authenticator does not need to be unique but if the authenticator is secured by keeping it out of reach then it has to be unique or at least arguably unique.

C_2(b) *Unpredictability* has two aspects:

C_2(b) i *Dictionary Size:* It should not be easily possible to attribute the artefact to the creator or at least to narrow down the possible identity of the artefact based on knowledge of the user. The size of the dictionary is only relevant when a potential intruder cannot predict which element someone will choose. Hence the selection process must be unpredictable, but, having made that choice, it should be impossible for someone easily to guess it. The artifact should not be in the public domain if a recognition-based mechanism is going to be used. So, for example, one could not make use of a famous artist's paintings to allow the artist herself to authenticate. Other examples of easy attribution exist. For example Argamon [34] shows that it is possible to determine the gender of a writer from their written text. Estival et al. [35] show how analysis of an email can tell you even more about the author. This means that a paragraph written by a skilled writer would be unsuitable for use in authentication.

C_2(b) ii *Abundance:* It must be possible to find viable distractor images. For example, if we make use of handwritten mathematical proofs to authenticate mathematicians, we would have to display the user's proof, and then as distractors in the challenge set a number of proofs written by other mathematicians. We would expect the expert to identify their own proof, in their own handwriting.

C_2(c) *Recognisability (Memorability):* It must be possible for users easily to recognise their own artefact. Since they have created the artefact themselves, this should help them to recognise it [36].

C_2(d) *Matching Process Capability:* Does the matching process deliver a definitive answer, or does it deliver a confidence level? The former constitutes more soundness than the latter.

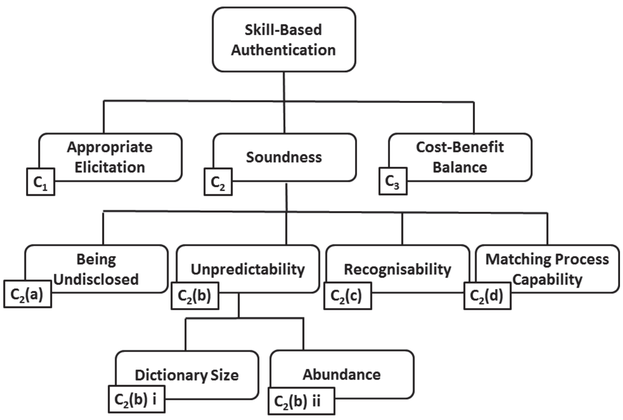

Figure 6: Extending Constraints of Skill-Based Authentication from Figure 4

We will attempt to meet these constraints by *personalising* the authentication secret. Users already do this intuitively when they choose passwords that are related to themselves i.e. information rather than data. Here we propose to advance another level up the pyramid (Fig. 3).

3.1 Personalising Authentication Secrets

Humans can recognise a lot of things about themselves. For example their own voices [37], their own handwriting [38, 39], their own performance (pianists) [40] odour [41] and gait [42]. Hence images that are related to the user should make them easy to recognise but it might well also make them easier to guess. There are other ways of maximising recognition success. For example, a graphical mechanism using facial images could be tailored to maximise recognition by tailoring the entire challenge set to the age [43], race [44] and gender [45] of the user. This would help the user but not make things easier for an attacker. All these variations would personalise the images to maximise the legitimate user's chances of being able to remember and identify their images.

Some authentication schemes have attempted to make use of personalised images. Dynahand [46] relies on the user being able to recognise his or her own handwriting (Fig. 7). It collects 10 examples of participants' handwritten numerals at enrolment. It then generates random PINs using the user's own handwritten numerals, and generates distractors from other users' handwritten numerals. Four challenge sets are displayed, and each time the user picks out the displayed PIN written in his or her own handwriting. A casual observer has less chance of gaining access to the user's account later because what is being tested, i.e. handwriting recognition, is relatively obscure and less easily cracked than a straightforward set of pictures. Moreover, it is completely effortless for the user.

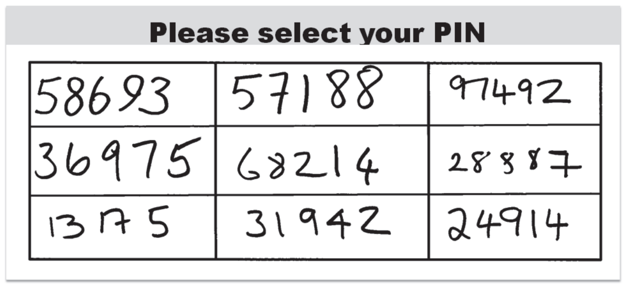

Figure 7: A Dynahand Challenge Set

Renaud [30] deployed this technique as one stage of the Handwing authentication mechanism to control access to a website used by a community group where the community members very successfully identify their own handwriting to authenticate. The mechanism also exploits the user's ability to recognise their own hand-drawn doodle and has been very successful — and is still being used 10 years later. Renaud [47] also tested the same concepts with a graphical authentication mechanism that used Mikon (my icons) images (Fig. 8). Users drew these using a browser-based engine. The majority of the participants in the study were able to remember all their Mikons successfully after a three month period of non-use. These examples serve to show that personalised images are recognisable but we don't yet know how predictable they will be.

3.2 Personalising Secrets for Experts

The schemes mentioned thus far did not exploit a particularly stringent or rare skill: almost everyone can write and draw. They do, however, demonstrate that people have the potential to remember, and to recognise artefacts resulting from their skilled practice. In the case of the drawn images, the images are more memorable than passwords because they rely on visual, lexical and kinaesthetic memory [48] rather than mere textual memory.

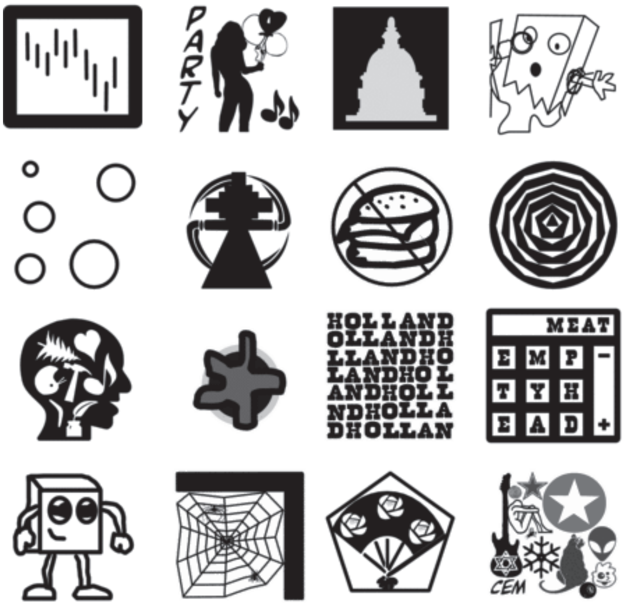

Figure 8: A Mikon Challenge Set

We propose to extend this concept to test whether experts can recognise the outputs from their own skilled actions, in this case programming language code. It takes thousands of hours to become a competent programmer [49]. Although there are millions of programmers in the world, the number is significantly smaller than those who can write and draw.

Let us consider programming code snippets in terms of the constraints introduced earlier in this section.

1. *Undisclosed:* Programming code is often not in the public domain — it is essentially hidden from view. Open source code is the obvious exception but it is not clear that a programmer's particular style would be recognised by anyone else.
2. *Unpredictability:* Requires empirical testing.
3. *Abundance:* Finding viable distractor images is trivial if the snippet is of a widely used programming language. It is also entirely possible to automate the generation of such distractors, which would make abundance a non-issue.
4. *Recognisability:* It should be possible for programmers to recognise their own code. Craik and Tulving [50] argue that the development of memory traces should be considered in terms of *depth of processing*. Programming is a cognitively demanding task and so the production of an artifact should lay down strong memory traces. The advantage is that using snippets of code would not require the user deliberately to memorise anything. This addresses the primary root cause of insecure password behaviours.
5. *Matching Process Capability:* We can perform an exact match at authentication.

From the above list, we see that *unpredictability* and *recognisability* need to be verified. Before we proceeded to testing these aspects though, we wanted to find out from skilled programmers whether they thought this scheme had any merit.

4. FACT FINDING

In order to determine whether this idea had any chance of succeeding, we started off by posting an online survey. We advertised it via developer forums and to our respective institutions' postgraduate students. 198 programmers responded to our survey. The majority (179) had been programming for more than 3 years with the largest group (60) in the 5-10 year category.

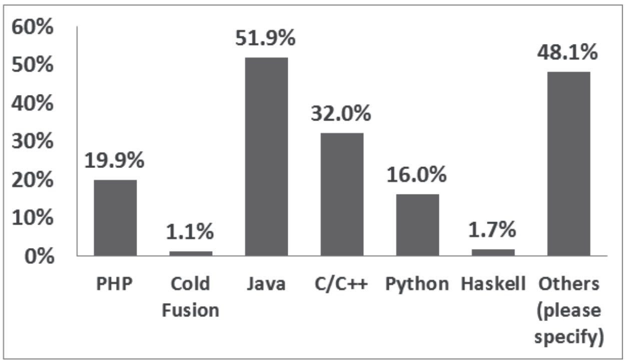

Figure 9: Which Programming Language did They Use

Figure 9 shows the distribution of programming languages used by the respondents. Some people mentioned C#, ASP, Javascript, PL/1, Perl and Assembler. The most commonly used language was Java. We provided a box for comments.

80% of the respondents agreed with the statement: "Every Programmer has his/her own programming style". Figure 10 presents the responses. This appears to confirm the findings that people develop personal styles [51]. Some comments from the respondents:

> *"programmers I knew all looked to add their own personalisation - it is their baby"*

> *"It's a mistake if a programmer doesn't have his/her own programming style as it is important to recognizing your own programs"*

> *"Programming is an expression of thoughts much like poetry. So a programmers individual style will be reflected in the piece of code that he/she develops. Bottom line is there can be several alternative solutions for a single problem, and different programmer may adopt different style. "*

> *"Yes, it's like writing where every author has his own writing style as well. "*

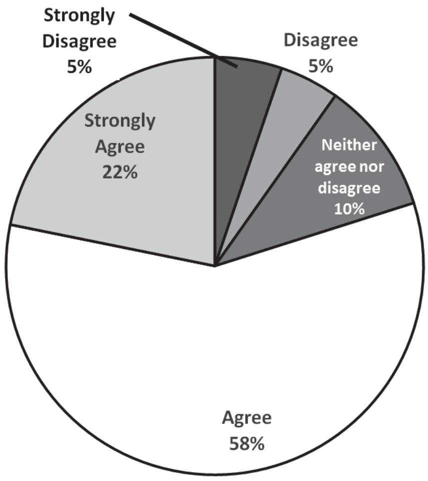

Figure 10: Every Programmer has his/her own programming style

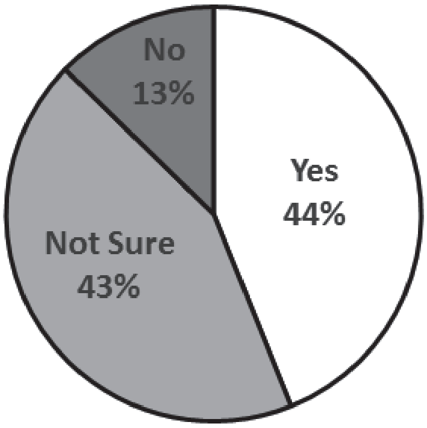

Figure 11: Could you Identify your own code from a group of code snippets 10 lines long?

44% felt sure they could identify their own code, with another 43% being unsure (Figure 11). One said: *"My own typing style is distinct (the whitespaces, the way I comment, variable naming, etc). I'm sure I can identify snippets of my own code among others"*. Only 20% felt they would be able to identify another programmer's code even if they knew the person well (Figure 12).

The survey responses convinced us that it would be beneficial to trial a scheme which tested whether (1) people could recognise their own code (recognisability) and (2) people could recognise each others' code (unpredictability).

5. EXPERIMENT

Our survey of programmers made it clear that while many of them felt they had a particular programming style, only 44% felt they would be able to identify their own code. Our findings had suggested that skill-based authentication demonstrated some promise, but it was clearly necessary to verify these soundness aspects empirically.

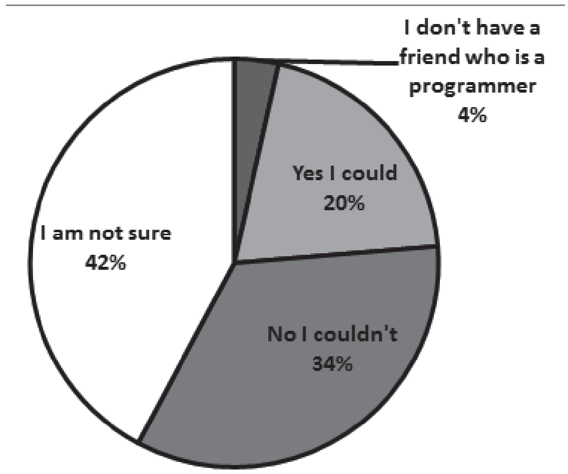

Figure 12: Could you identify a friend's code from a group of code snippets 10 lines long?

We carried out a proof of concept experiment into the use of "skill-based" authentication. The area of expertise we focused on was programming, since we possessed this skill ourselves and we worked in an environment that gave us access to a number of expert programmers. The aim was to design an authentication mechanism which would authenticate programmers based on their own programming skills, a genuine knowledge-based test. A recognition-based graphical authentication system which used snippets of code, instead of images, was implemented. We hoped to show that programmers would be able to recognise their own code, but that others, even those who are experts in the same language, would not easily be able to recognise the person's code snippet.

Our participants were 20 programmers, Masters students who had been together in the class for some 9 months and so knew each other fairly well. We asked them to provide five snippets of code in Java, since this was the most widely used language mentioned in our questionnaire. They were asked to avoid snippets containing comments. This constituted enrolment.

We then asked them to return a week later to see whether they could identify their own code from four challenge sets. Participants were required to identify their own code snippet from four challenge sets, each composed of 16 code snippets. Distractors, and targets, varied each time the user tried to authenticate since we had more code snippets than we needed for one authentication attempt. An example challenge set is shown in Figure 14.

To verify the two aspects identified as needing verification in Section 3, we tested recognisability (memorability) and predictability of the code snippets. Participants worked in parallel. For example, Participant A would authenticate while participant B watched. Then Participant B tried to replicate the attempt. Participant C, on the other hand, attempted to guess Participant A's code without observing A authenticating. Hence every participant observed another authenticating and tried to replicate the attempt. They also tried to guess one other person's codes without observing them authenticating.

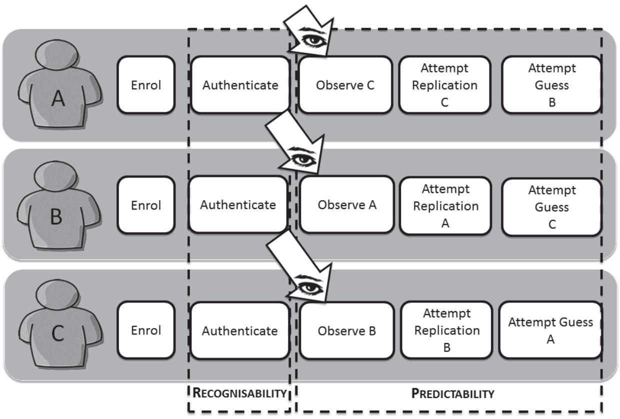

Figure 13: Participants Working in Pairs

5.1 Results

Recognisability: Identifying Their Own Code All participants were able to identify their code, some almost immediately, but some needing some time to examine the snippets in the challenge set. We did not record timings since these would not have supported analysis with so few participants. We asked the participants what particular aspect of the code made it memorable. Some of them stated that they identified their variables, others functionality or Java class names. One participant identified his secret sequence of images in less than a minute as the variables were expressed in his national language, whereas the others were in English.

Predictability: Guessing Another's Code None of the "attackers" managed to identify another's code images, both when they observed the authentication and when they just tried to guess it. This is probably due to the fact that the images and the distracters are varied so the attacker would need to identify the programmer's *style* and not one specific piece of code.

while (ss.moreStudents()) {
Student stud = ss.getNextStudent();
if (stud == null) break;
if (stud.degree.equals("PhD")) {
Staff.recordSupervision(stud.first,
FIRST_SUP);
Staff.recordSupervision(stud.secon
d,
SECOND_SUP);
}
	public class DB extends RSC {
	String connString="jdbc:mysql://";
	String username="";
	String password="";
	Connection conn=null;
	String driverName=
	"org.gjt.mm.mysql.Driver";
	boolean connError=false;
		try{ Statement stmt =
		conn.createStatement();
		ResultSet rsRota =
		stmt.executeQuery(query);
		while (rsRota.next()) {
		count = rsRota.getInt("cnt");} }
		catch (Exception ee) {
		System.out.println("Prob getting
		count " + ee.getMessage());}
		return count;
			public String name;
			public String initials="";
			public String email;
			public int id;
			public boolean external = false;
			public int tally1st=0;
			public int tally2nd=0;
			public int numoffers=0;

if (isExtensionRequest) {
int storeid =
storeExtensionRequest(studentNa
me,studentid,reason,courseworkNu
mber,courseid,getLecturerId(course
id));
sendEmailToStudent(from,student
Name,courseworkNumber,courseN
ame,storeid) ;
	p2.setForeground(Color.black);
	p2.setBackground(Color.white);
	p2.setLayout(new BorderLayout());
	JPanel p2small = new JPanel();
	greenquestion = new
	JLabel(questionicon);
	p2small.add(greenquestion)
	;
		JCheckBox sounds1 = new
		JCheckBox("Play Lightning Sound");
		JCheckBox sounds2 = new
		JCheckBox("Play Star Sound");
		JCheckBox sounds3 = new
		JCheckBox("Play Absent Sound");
		JCheckBox showStar = new
		JCheckBox("Show Gold Star");
		JCheckBox showFish = new
		JCheckBox("Show Absence Button");
			public class csvEntry {
			
			String coursename;
			String aenumber;
			String matric;
			String band;
			String dayslate="0";
			boolean instr=false;

String [] cmd = {"/bin/bash","-
c","sudo /usr/bin/gammu
deletesms 1 "+num};
java.lang.Process p =
Runtime.getRuntime().exec(cmd);
p.waitFor();
	try {
	System.out.println("Sleeping for 5
	minutes ...." + getTime());
	synchronized(waitObject) {
	waitObject.wait(delay);
	}}
	catch (Exception ee) {// do nowt
	ee.printStackTrace();
	System.exit(0);};
		int posit2 =
		s.substring(pos).indexOf("Status");
		String fromnumber =
		s.substring(pos+posit,pos+posit2);
		fromnumber=fromnumber.replace(
		"\"","");
		fromnumber=fromnumber.replace(
		"+","");
			public static boolean bad(File f) {
			for (int i=0; i<badfiles.size(); i++) {
			if (badfiles.elementAt(i).
			getAbsolutePath().equals
			(f.getAbsolutePath())) return true;}
			return false;
			}

public static void main(String[] args)
{ Scanner userInputScanner = new
Scanner(System.in);
System.out.print("What's your
name? "); String userInputName =
userInputScanner.nextLine();
System.out.println("Hello " +
userInputName + "!"); }
	public boolean
	monkeyTrouble(boolean aSmile,
	boolean bSmile) {
	if (aSmile && bSmile) { return true;
	} if (!aSmile && !bSmile) { return
	true; } return false;
		public void paint(Graphics g) {
		g.drawString("Click, drag, and type
		in this window.", 10, 20); } //
		Handle mouse events public
		boolean mouseDown(Event e, int x,
		int y) {
		showStatus(modifier_key_names(e.
		modifiers) + "Mouse Down: [" + x +
		"," + y + "]"); return true; }
			public int bunnyEars(int bunnies) {
			// Base case: if bunnies==0, just
			return 0. if (bunnies == 0) return 0;
			// Recursive case: otherwise, make
			a recursive call with bunnies-1 //
			(towards the base case), and fix up
			what it returns. return 2 +
			bunnyEars(bunnies-1); }

Figure 14: An Example Challenge Set

Participant Comments We asked participants to express their opinions about the mechanism when the experiment concluded. All reported finding it easy to locate their own code snippets. 17 of the 20 believed it would be impossible for anyone else to identify their images. Some specific comments:

> *"The idea of having code images as passwords is unique and I believe holds a good future"*

> *"First time I used this mechanism was a bit difficult but gradually it became easy for me. Moreover I believe it is easier to remember images than text-based passwords."*

6. DISCUSSION

To end off this paper we return to the issue of authentication in general in order to position SNIPPET within the arena. To position authentication in terms of access control, consider that a person who wishes to access restricted information or services has to prove that they have the right to do so. This is a two-step process: *identification* followed by *authentication*, proof that the person claiming the identity does indeed own it.

The identifier needs to be unique but does not have to be secret. The most often used identifier is a username or email address, neither of which is necessarily secret.

NIST published a guideline for authentication in 1977, which argued that authenticators could fall into one of three categories: what you *know*, what you *are* and what you *hold* [52]. This model is simple and easy to understand but, in 2013, probably fails to capture the nuances of a rapidly changing authentication and identification arena. It is time to pose two pertinent questions:

1. Are the NIST categories still all-encompassing?
2. Do instances of the "big-three" authentication categories meet the soundness constraints?

6.1 Are the NIST categories still all-encompassing?

A number of new mechanisms have been proposed in the intervening years since NIST published their categories. Here we will provide a few examples, and show how/whether they fit into one of the already-proposed categories. This list is not exhaustive, but does provide a flavour of the research activity in the interim.

- NIST Categories

  - *What you know*: This field has moved on from the humble password. Later developments require a user to draw a picture [53–56]. The latest incarnation of this kind of mechanism is the sketch-based mechanism on the Android [57]. Others require users to remember positions within an image [33]. There has also been a great deal of work relying on people's memory of images [31, 58] or faces [59, 60] rather than an alphanumeric string.
  - *What you hold*: Traditional card ownership is moving to mobile phone ownership. Al Fairuz and Renaud [61] utilise the mobile phone channel to deliver a one-time password to authenticate transactions. Other examples are wearable keys [62] and RFID tags [63]. A relatively new addition to this category is the *embedded chip*. These can be used to gain access to controlled areas such as homes and offices, and grant access to electronic devices such as mobile phones. These chips have proved to be an emotive issue with privacy and bodily integrity concerns [64].
  - *What you are*: in this area much work has been done in the intervening years. Fingerprint readers have started to appear in products such as laptops (eg. IBM Thinkpad) and mobile phones (eg. iPhone 5S). There is also a growing body of research focusing on behavioural biometrics: authenticating people according to the way they use their device [65, 66].

- New Categories
  - *What skill you can demonstrate*: An example of this is the work by Tao and Adams, who propose an authentication mechanism inspired by the ancient Chinese board-game, Go [67], which relies on the user knowing how to play the Go game.
  - *Who you know*: Brainard [68] proposed a new kind of authentication, based on some*one* you know rather than some*thing* you know. This adds a social aspect to authentication, which has traditionally been a solo exercise.
  - *What you associate*: Smith proposes the use of word association to authenticate [27, 69]. This is fairly unique because every human reasons in a slightly different way. However, it is very time consuming.

Hence the original three categories have been augmented in the interim but it must be noted that these additional kinds of authentication have not been embraced by industry, probably being seen as novel and, as yet, unproven. Moreover, there is contention about whether some of the mechanisms mentioned above are indeed authenticators or actually identifiers. The following section will consider the second question above for the three traditional authentication categories.

6.2 Authentication Category Soundness

Non-disclosure, by means of secrecy, seems an obvious requirement when it comes to "what you know" mechanisms: users know that they ought to withold their passwords from others. If the secret is remembered and retained the legitimate user will always gain access, and imposters will be resisted. Unfortunately the newspapers abound with stories that prove that passwords are often not retained. There is a suggestion that humans find it difficult to keep secrets [70] and that revealing secrets is cathartic [71]. It must be acknowledged that in the secret-based academic literature the kind of secrets being referred to are those that people tend to be ashamed of, so these findings might not apply to keeping passwords secret. Still, there is a social element to password sharing that suggests that there is more to divulging password secrets than mere carelessness [72–74]. Hence increased availability compromises the non-disclosure of the mechanism.

For tokens, availability is ensured by keeping the token secure, i.e. close at hand. They can, unfortunately, be lost or stolen quite easily. Tokens are thus usually paired with knowledge or a biometric so that they can serve as authenticators. Given that the soundness of the token is so easily compromised, and the fact that they require a second factor in order to support authentication, we should perhaps refer to tokens as *private identifiers*. They are more secure than self-proferred usernames because their availability is somewhat restricted. Yet on their own they do not reliably authenticate the card holder, so they cannot realistically qualify as authenticators.

The third NIST category is the biometric. The most popular of these is the fingerprint, perhaps unsurprisingly since it is has the most established use in other contexts, such as law enforcement [75], and readers are relatively inexpensive. Much has been written recently about the use of fingerprints to protect mobile devices, and this has been made a major selling feature of the new iPhone 5S, but unfortunately it is the case that these digital fingerprint readers are not infallable [76]. The Chaos Computer Club spoofed the iPhone fingerprint biometric within a week of it being released, merely by copying a person's fingerprint onto a piece of paper. [77]

This highlights one of the biggest problems with biometrics: the fact that they are not secret. Many countries collect them when people travel there, users leave them all over their homes, desks, wherever they go. Having obtained the fingerprint, there are some who know how to create a fake finger which can fool a biometric reader [8]. This means that possession of the biometric does not automatically authenticate the user: there is a chance that the person presenting the biometric is an imposter. Even if the legitimate user is presenting the biometric it sometimes fails to authenticate the user since the matching process is not an exact science.

This leads us to the second requirement: *soundness*. A password challenge leads to a binary decision: match or no match. There are no grey areas inbetween. With a biometric, on the other hand, there is a matching process that leads to a confidence level: the biometric reader is seldom if ever going to deliver a 100% match between the stored template and the currently presented biometric. Soundness depends on a number of factors, ranging from the quality of the reader to slight changes in the biometric that happen quite naturally, perhaps as users age. Hence a ruling made when a biometric is presented is more in the nature of "eliminating reasonable doubt" rather than being able to rule definitively in one direction or another.

Given that they fail the soundness constraint, perhaps biometrics, too, should be referred to as *private identifiers*, once again stronger than a user name, but perhaps not entirely suited to use in an authentication context.

All the authenticators from the three original categories seem to have flaws but tokens and biometrics seem particularly problematic. This confirms the fact that authentication in the digital world is much harder than it seems at first glance.

In proposing mitigation, we have chosen to focus on the the most common authentication mechanism, "what you know" authentication. This is the mechanism most users are familiar with, and it is most accessible and usable, so this is where amelioration might deliver the greatest benefit. It seems that passwords fail because humans cannot remember them and because they are so easy to divulge. This makes them choose information-rich, and easily guessed, passwords or compromise their secret passwords by recording them. Weidenbeck argues:

> **"A better way to overcome the password problem is to develop password systems that reduce fundamental memory problems"** [78, p. 105]

Hence we should try to address the memorability issue. If there were a way to ease the password's memorial load and to make guessability probematical we might well strengthen the mechanism.

The research reported here seems a viable direction to take in terms of strengthening "what-you-know" authentication since it addresses memorability issues, and because skill artefacts are more unpredictable than passwords. However, it could reasonably be argued that soundness could be compromised when one programmer wishes to guess another's password, since they share the same skill set. Thus SNIPPET adds another dimension: action-planning memory, thus exploiting the generation effect [36]. An imposter does not only have to have the same skill set, they have to have the other part of the secret, the personal involvement with the production of the authenticator artefact, in order to be able to identify the correct target image. The authenticator artifact is the *result* of an expert deploying their skills. Our small pilot study has shown that, even amongst Java programmers who knew each other well, this second dimension helped to resist guessing attempts.

	User Authentication		
	Know-ledge-	**Object-based**	**ID-based**
Commonly Referred to as:	Password	Token	Biometric
Security Defense:	Closely kept	Closely held	Forge-resistant
Security Drawback:	Less secret with each use. Hard to remem-ber	Can be cloned	Impossible to replace. Not secret. No exact match
Soundness:	Yes if kept secret	Needs addi-tional knowl-edge or biomet-ric (must be kept secure)	Context & Bio-metric Depen-dent
Obscurity:	Secret	Possession	Possession
Unpredictability:	Secret	Unpre-dictable	Unpredictable
Matching:	Exact	Exact	Confidence Level
Memorability:	Not memo-rable if strong	Can be lost or stolen	N/A
Convenience:	Depends on strength	Can be lost	Very, except for false rejects and reader issues

Table 1: Extending O'Gorman's categorisation of authentication approaches [79, p. 7]

6.3 Summary

We have to consider whether the distinction between identification and authentication has blurred in our digital age. Biometrics have traditionally been an *identification* mechanism in the pre-IT world, and not used as *authentication* mechanisms. When one tries to use an identifier as an authenticator you come up against all the same problems you would for any unproven identity.

Tokens, as exemplified by bank cards and others of their kind, are also identification mechanisms. The holder of the card always has to proffer further proof that they do indeed have the right to hold the card: to verify their identity. The driver's licence has a biometric: the person's face, and in South Africa their fingerprint too. Thus the combination of token and authenticator acts as convincing evidence that the holder of the card is entitled to claim the identity.

Only "what you know" mechanisms really keep the identifier completely separate from the authenticator. Given this desirable characteristic, it is definitely worth trying to bolster this mechanism to address its one big flaw: the memorial load imposed on users. A way to strengthen authentication is to remove need for the user to deploy coping mechanisms, i.e. to reduce the possibility that they will forget their authenticator. SNIPPET does this, by testing the expert's ability to identify artefacts that result from their practicing their trade, i.e. the evidence of their expert practice. From the evidence we have gathered this seems to be completely effortless, since it is encoded at a level in the brain that is not easily eroded.

Whereas this skill-based mechanism performed well, there is one issue that remains: the cost-benefit balance [22]. For the users the mechanism delivered a good cost-benefit balance since no effort was involved in recognising their own code snippets. They provided these snippets themselves, which gave the advantage of recognisability but since they were produced by skilled actions they were also less predictable than other schemes where users provided their own images [29]. Yet the manual selection of distractors, in order to ensure maximum strength, means that the system, as implemented, was not scalable. These images must be chosen carefully and should be purposely similar to the user's sequence of code snippets, in terms of programming language and perhaps the language used for the variables. In this way we could maximise the possibility that the distractors do not weaken the mechanism by making the target stand out. Clearly automatic generation of such distractors would be an interesting topic for further research.

Finally, even though users are less likely to forget their SNIPPET secrets, it is possible that this, in itself, will not deliver sufficient benefit to persuade organisations to expend the extra effort required to deploy SNIPPET.

7. CONCLUSION

In this paper we have argued that the common and garden password cannot reasonably be referred to as an instance of "knowledge-based authentication". Passwords are ideally meaningless and therefore more akin to nonsense than knowledge. We have also pointed out the flaws of the two most popular alternatives: tokens and biometrics, and concluded that they could perhaps be more aptly used as secure identifiers. They do not really satisfy the soundness constraint required of authenticators.

We have tested skill-based authentication, structured as a recognition-based graphical authentication mechanism. We found that it was possible successfully to test recognition of the artefacts resulting from the practice of skilled activities in an authentication setting. Moreover, such authentication appears to be both memorable and resistant to shoulder-surfing and guessing attacks. There is admittedly a problem related to scalability of the solution in an industrial setting and this is an area that merits further consideration. Certainly these preliminary findings suggest that further research is worthwhile.

REFERENCES

[1] F. J. Corbató, M. Merwin-Daggett, and R. Daley, "An Experimental Time-sharing System," in *Proceedings of the Spring Joint Computer Conference*. ACM, May 01-03 1962, pp. 335–344.

[2] B. Riddle, M. Miron, and J. Semo, "Passwords in use in a university timesharing environment," *Computers and Security*, vol. 8, no. 7, pp. 569–578, 1989.

[3] B. Schneier, "Two-factor authentication: Too little, too late." *Communications of the ACM*, vol. 48, no. 4, 2005.

[4] A. Conklin, G. Dietrich, and D. Walz, "Password-based authentication: a system perspective," in *System Sciences, 2004. Proceedings of the 37th Annual Hawaii International Conference on*. IEEE, 2004, pp. 10–pp.

[5] D. Florêncio, C. Herley, and B. Coskun, "Do strong web passwords accomplish anything?" in *Proceedings of the 2nd USENIX workshop on Hot topics in security*. USENIX Association, 2007, p. 10.

[6] Unattributed BBC News Item, "Apple sells 9 million of its new iPhone models," 23 September 2013, http://www.bbc.co.uk/news/business-24201526. Accessed 16 October, 2013.

[7] S. Pankanti, R. Bolle, and A. Jain, "Biometrics: The future of identification [guest eeditors' introduction]," *Computer*, vol. 33, no. 2, pp. 46–49, 2000.

[8] C. Barral and A. Tria, "Fake fingers in fingerprint recognition: Glycerin supersedes gelatin," in *Formal to Practical Security*. Springer, 2009, pp. 57–69.

[9] A. K. Karlson, A. Brush, and S. Schechter, "Can i borrow your phone?: understanding concerns when sharing mobile phones," in *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*. ACM, 2009, pp. 1647–1650.

[10] S. Landau, "Making sense from Snowden," *IEEE Security & Privacy Magazine*, no. 4, p. 5463, 2013, http://privacyink.org/html/MakingSense.pdf.

[11] HNBulletin, "Exclusive: Apple admits, 'iPhone 5s Fingerprint Database to be Shared with NSA'," 23 Sept 2013, hacker News Bulletin. [Online]. Available: http://hackersnewsbulletin.com/2013/09/apple-admits-iphone-5s-fingerprint-database-shared-nsa.html

[12] National Report, "Apple iPhone 5s fingerprint database to be shared with NSA," Sept 2013, hacker News Bulletin. [Online]. Available: http://nationalreport.net/apple-iphone-5s-fingerprint-database/

[13] Corbató, F.J., "On building systems that will fail," in *ACM Turing Award Lectures*. ACM, 1990.

[14] Federal Trade Commission, "Federal Trade Commission, Plaintiff, v. Wyndham Worldwide Corporation; Wyndham Hotel Group, LLC; Wyndham Hotels & Resorts, LLC; and Wyndham Hotel Management, Inc., Defendants (United States District Court for the District of Arizona)," August 2012. [Online]. Available: http://www.ftc.gov/os/caselist/1023142/120809wyndhamcmpt.pdf

[15] A. Adams and M. A. Sasse, "Users are not the enemy," *Comm. of the ACM*, pp. 40–46, 1999.

[16] G. Kreizman and A. Allan. (2006, November) Toolkit: Evaluating Enterprise Options for Managing Passwords. [Online]. Available: http://www.gartner.com/id=498322

[17] F. Cunha, J. J. Heckman, L. Lochner, and D. V. Masterov, "Interpreting the evidence on life cycle skill formation," *Handbook of the Economics of Education*, vol. 1, pp. 697–812, 2006.

[18] W. A. Rogers, "Assessing age-related differences in the long-term retention of skills," *Aging and skilled performance: Advances in theory and applications*, pp. 185–200, 1996.

[19] R. Smilkstein, "We're born to learn: Using the brain's natural learning process to create today's curriculum," 2003.

[20] L. Anderson, D. Krathwohl, P. Airasian, K. Cruikshank, R. Mayer, P. Pintrich, J. Raths, and M. Wittrock, *A Taxonomy for Learning, Teaching, and Assessing: A Revision of Bloom's Taxonomy of Educational Objectives, Complete Edition*, L. Anderson and D. Krathwohl, Eds. Longman, 2001.

[21] US-CERT: Official Website of the Department of Homeland Security, "Security tip (st04-002) choosing and protecting passwords," May 2009. [Online]. Available: http://www.us-cert.gov/ncas/tips/ST04-002

[22] S. Kurzban, "Easily remembered passphrases: a better approach," *ACM SIGSAC Review*, vol. 3, no. 2-4, pp. 10–21, 1985.

[23] H. Crawford, "A framework for continuous, transparent authentication on mobile devices," Ph.D. dissertation, 2013.

[24] M. Perlmutter, "Age differences in adults' free recall, cued recall, and recognition," *Journal of Gerontology*, vol. 34, no. 4, pp. 533–539, 1979.

[25] L. Bäckman and L.-G. Nilsson, "Prerequisites for lack of age differences in memory performance," *Experimental Aging Research*, vol. 11, no. 2, pp. 67–73, 1985.

[26] K. Renaud, T. McBryan, and P. Siebert, "Password cueing with cue(ink)blots," in *IADIS Computer Graphics and Visualization 2008 (CGV 2008) Conference*, Amsterdam. The Netherlands, 24 - 26 July 2008, pp. 74–81.

[27] M. Zviran and W. J. Haga, "Cognitive passwords: the key to easy access control," *Computers & Security*, vol. 9, no. 8, pp. 723–736, 1990.

[28] D. Davis, F. Monrose, and M. Reiter, "On User Choice in Graphical Password Schemes," in *Proceedings of the 13th USENIX Security Symposium*, 2004, pp. 151–164.

[29] T. Pering, M. Sundar, J. Light, and R. Want, "Photographic authentication through untrusted terminals," *Pervasive Computing, IEEE*, vol. 2, no. 1, pp. 30–36, 2003.

[30] K. Renaud, "A visuo-biometric authentication mechanism for older users," in *People and Computers XIXThe Bigger Picture*. Springer, 2006, pp. 167–182.

[31] R. Dhamija and A. Perrig, "Déjà Vu: A User Study Using Images for Authentication," in *Proceedings of the 9th Conference on USENIX Security Symposium*, 2000.

[32] R. English and R. Poet, "Measuring the revised guessability of graphical passwords," in *Network and System Security (NSS), 2011 5th International Conference on*. IEEE, 2011, pp. 364–368.

[33] K. Renaud and A. De Angeli, "My password is here! An investigation into visuo-spatial authentication mechanisms," *Interacting with computers*, vol. 16, no. 6, pp. 1017–1041, 2004.

[34] S. Argamon, M. Koppel, J. Fine, and A. R. Shimoni, "Gender, genre, and writing style in formal written texts," *Text*, vol. 23, no. 3, pp. 321–346, 2003.

[35] D. Estival, T. Gaustad, S. B. Pham, W. Radford, and B. Hutchinson, "Author profiling for english emails," in *Proceedings of the 10th Conference of the Pacific Association for Computational Linguistics*, 2007, pp. 263–272.

[36] N. J. Slamecka and P. Graf, "The generation effect: Delineation of a phenomenon." *Journal of experimental Psychology: Human learning and Memory*, vol. 4, no. 6, p. 592, 1978.

[37] C. Fernyhough and J. Russell, "Distinguishing ones own voice from those of others: A function for private speech?" *International Journal of Behavioral Development*, vol. 20, no. 4, pp. 651–665, 1997.

[38] M. Longcamp, J.-L. Anton, M. Roth, J.-L. Velay *et al.*, "Visual presentation of single letters activates a premotor area involved in writing," *Neuroimage*, vol. 19, no. 4, pp. 1492–1500, 2003.

[39] M. Longcamp, T. Tanskanen, and R. Hari, "The imprint of action: motor cortex involvement in visual perception of handwritten letters," *Neuroimage*, vol. 33, no. 2, pp. 681–688, 2006.

[40] B. H. Repp and G. Knoblich, "Perceiving action identity how pianists recognize their own performances," *Psychological Science*, vol. 15, no. 9, pp. 604–609, 2004.

[41] M. Schleidt, "Personal odor and nonverbal communication," *Ethology and Sociobiology*, vol. 1, no. 3, pp. 225–231, 1980.

[42] D. Jokisch, I. Daum, and N. F. Troje, "Self recognition versus recognition of others by biological motion: Viewpoint-dependent effects," *Perception*, vol. 35, pp. 911–920, 2006.

[43] J. S. Anastasi and M. G. Rhodes, "Evidence for an own-age bias in face recognition." *North American Journal of Psychology*, 2006.

[44] J. Stahl, H. Wiese, and S. R. Schweinberger, "Expertise and own-race bias in face processing: an event-related potential study," *Neuroreport*, vol. 19, no. 5, pp. 583–587, 2008.

[45] D. B. Wright and B. Sladden, "An own gender bias and the importance of hair in face recognition," *Acta psychologica*, vol. 114, no. 1, pp. 101–114, 2003.

[46] K. Renaud and E. Olsen, "Dynahand: Observation-resistant recognition-based web authentication," *Technology and Society Magazine, IEEE*, vol. 26, no. 2, pp. 22–31, 2007.

[47] K. Renaud, "Web authentication using mikon images," in *Privacy, Security, Trust and the Management of e-Business, 2009. CONGRESS'09. World Congress on*. IEEE, 2009, pp. 79–88.

[48] K. Renaud and A. De Angeli, "Visual passwords: cure-all or snake-oil?" *Communications of the ACM*, vol. 52, no. 12, pp. 135–140, 2009.

[49] M. Gladwell, *Outliers: The Story of Success*. Penguin, 2009.

[50] F. Craik and E. Tulving, "Depth of Processing and the Retention of Words in Episodic Memory." *Journal of Experimental Psychology: General*, vol. 104, no. 3, pp. 268–294, September 1975.

[51] J. J. Heckman, "The economics, technology, and neuroscience of human capability formation," *Proceedings of the National Academy of Sciences*, vol. 104, no. 33, pp. 13 250–13 255, 2007.

[52] NIST, "Guidelines on Evaluation of Techniques for Automated Personal Identification," National Institute of Standards and Technology, Tech. Rep. FIPS-PUB-48, April 1977.

[53] I. Jermyn, A. Mayer, F. Monrose, M. Reiter, and A. Rubin, "The Design and Analysis of Graphical Passwords," in *Proceedings of the 8th USENIX Security Symposium*. Washington DC, 23-26 August 1999, pp. 1–14.

[54] D. Lin, P. Dunphy, P. Olivier, and J. Yan, "Graphical Passwords & Qualitative Spatial Relations," in *Proceedings of the 3rd Symposium on Usable Privacy and Security*. ACM, 18-20 July 2007, pp. 161–162.

[55] M. Oka, K. Kato, Y. Xu, L. Liang, and F. Wen, "Scribble-a-Secret: Similarity-based Password Authentication Using Sketches," in *Proceedings of the 19th International Conference on Pattern Recognition*. IEEE, 2008, pp. 1–4.

[56] R. Weiss and A. De Luca, "PassShapes: Utilizing Stroke Based Authentication to Increase Password Memorability," in *Proceedings of the 5th Nordic Conference on Human-Computer Interaction: Building Bridges*. ACM, 2008, pp. 383–392.

[57] T. Matthews, D. Vogts, and K. Naudé, "Sketch-based interfaces: drawings to data," in *Proceedings of the South African Institute for Computer Scientists and Information Technologists Conference*, ser. SAICSIT '13. New York, NY, USA: ACM, 2013, pp. 359–368. [Online]. Available: http://doi.acm.org/10.1145/2513456.2513482

[58] 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*, vol. 63, no. 1, pp. 128–152, 2005.

[59] The Passfaces Corporation, "The Science Behind Passfaces," *White Paper, June*, 2004.

[60] K. Renaud and J. Maguire, "Armchair authentication," in *BCS-HCI '09: Proceedings of the 23rd British HCI Group Annual Conference on People and Computers: Celebrating People and Technology*. British Computer Society, Sep. 2009.

[61] M. Al Fairuz and K. Renaud, "Multi-channel, multi-level authentication for more secure ebanking." in *2010 Information Security for South Africa (ISSA 2010) Conference*, Johannesburg, South Africa, 2010.

[62] N. Matsushita, S. Tajima, Y. Ayatsuka, and J. Rekimoto, "Wearable key: Device for personalizing nearby environment," in *Wearable Computers, The Fourth International Symposium on*. IEEE, 2000, pp. 119–126.

[63] Y.-C. Lee, Y.-C. Hsieh, P.-S. You, and T.-C. Chen, "An improvement on rfid authentication protocol with privacy protection," in *Convergence and Hybrid Information Technology, 2008. ICCIT'08. Third International Conference on*, vol. 2. IEEE, 2008, pp. 569–573.

[64] K. R. Foster and J. Jaeger, "RFID inside," *Spectrum, IEEE*, vol. 44, no. 3, pp. 24–29, 2007.

[65] M. Shahzad, A. X. Liu, and A. Samuel, "Secure unlocking of mobile touch screen devices by simple gestures: you can see it but you can not do it," in *Proceedings of the 19th annual international conference on Mobile computing & networking*, ser. MobiCom '13. New York, NY, USA: ACM, 2013, pp. 39–50. [Online]. Available: http://doi.acm.org/10.1145/2500423.2500434

[66] E. Maiorana, P. Campisi, N. González-Carballo, and A. Neri, "Keystroke dynamics authentication for mobile phones," in *Proceedings of the 2011 ACM Symposium on Applied Computing*, ser. SAC '11. New York, NY, USA: ACM, 2011, pp. 21–26. [Online]. Available: http://doi.acm.org/10.1145/1982185.1982190

[67] H. Tao and C. Adams, "Pass-Go: A Proposal to Improve the Usability of Graphical Passwords," *International Journal of Network Security*, vol. 7, no. 2, pp. 273–292, 2008.

[68] J. Brainard, A. Juels, R. Rivest, M. Szydlo, and M. Yung, "Fourth-factor authentication: somebody you know," in *Conference on Computer and Communications Security: Proceedings of the 13 th ACM conference on Computer and communications security*, vol. 30, 2006, pp. 168–178.

[69] S. Smith, "Authenticating Users by Word Association," *Computers & Security*, vol. 6, no. 6, pp. 464–470, 1987.

[70] A. Wismeijer, "Secrets and subjective well-being: A clinical oxymoron," in *Emotion regulation and well-being*. Springer, 2011, pp. 307–323.

[71] A. E. Kelly, J. A. Klusas, R. T. von Weiss, and C. Kenny, "What is it about revealing secrets that is beneficial?" *Personality and Social Psychology Bulletin*, vol. 27, no. 6, pp. 651–665, 2001.

[72] J. Berg, "Surrogate decision making in the internet age," *The American Journal of Bioethics*, vol. 12, no. 10, pp. 28–33, 2012.

[73] M. Richtel, "Young, in love and sharing everything, including a password," *The New York Times*, 2012.

[74] A. Patrick, "Monitoring corporate password sharing using social network analysis," in *International Sunbelt Social Network Conference*, St Pete Beach, Florida, 22-27 January 2008.

[75] Anonymous, "DNA Fingerprinting: A powerful law-enforcement. Tool with serious social implications," *The Scientist*, vol. 3, no. 11, p. 10, 1989, 29 May.

[76] T. Van der Putte and J. Keuning, "Biometrical fingerprint recognition: don't get your fingers burned," in *Smart Card Research and Advanced Applications*. Springer, 2000, pp. 289–303.

[77] S. Gold, "Meeting the biometrics payment security challenge," *Biometric Technology Today*, vol. 2013, no. 10, pp. 5–8, 2013.

[78] S. Wiedenbeck, J. Waters, J. Birget, A. Brodskiy, and N. Memon, "PassPoints: Design and Longitudinal Evaluation of a Graphical Password System," *International Journal of Human-Computer Studies*, vol. 63, no. 1, pp. 102–127, 2005.

[79] L. O'Gorman, "Comparing passwords, tokens, and biometrics for user authentication," *Proceedings of the IEEE*, vol. 91, no. 12, pp. 2021–2040, 2003.

FORENSIC ENTROPY ANALYSIS OF MICROSOFT WINDOWS STORAGE VOLUMES

P. J. Weston* and S. D. Wolthusen†

* *Information Security Group, Royal Holloway, University of London, Egham, Surrey TW20 0EX, UK*
E-mail: pete.weston@ntlworld.com
† *Norwegian Information Security Laboratory, Gjøvik University College, N-2818 Gjøvik, Norway and Information Security Group, Royal Holloway, University of London, Egham, Surrey TW20 0EX, UK*
E-mail: stephen.wolthusen@rhul.ac.uk

Abstract: The use of file or volume encryption as a counter-forensic technique depends on the ability to plausibly deny the presence of such encrypted data. Establishing the likely presence of encrypted data is highly desirable for forensic investigations. We claim that the current or previous existence of encrypted volumes can be derived from studying file and volume entropy characteristics using knowledge of the development of volume entropy over time. To validate our hypothesis, we have examined several versions of the Microsoft Windows operating system platform over a simulated installation life-cycle and established file and volume entropy metrics. Similarly we verified the hypothesis that the ageing through regular use of an installation is identifiable through entropy fingerprint analysis. The results obtained and tests devised allow the rapid identification of several volume-level operations and also detect anomalous slack space entropy indicative of the use of encryption techniques.

Key words: File System Entropy, Installation Aging, Encrypted File Systems

1. INTRODUCTION

The use of volume encryption by itself is insufficient to keep data confidential or as a counter-forensic technique when access to key material can be obtained or enforced. This may occur e.g. pursuant to Part III of the *UK Regulation of Investigatory Powers Act 2000*, requiring that a suspect supply decrypted information and/or cryptographic keys to authorised government representatives [1]. A skilled adversary will hence aim to use a combination of cryptography and steganography* to achieve plausible deniability, whilst forensic investigators must identify the presence of encrypted volumes for further analysis as an in-depth manual inspection may not always be feasible.

Entropy is a measure of the amount of information present in a signal or file.** A low entropy measurement implies a well-ordered, well-structured signal whilst a high entropy measurement indicates a signal with little apparent order or structure. Encrypted information, however, must not show discernible order or structure lest this make the encryption vulnerable to various forms of statistical attack i.e. encrypted data must have high entropy. Measurements of entropy and randomness taken against a computer system should reveal some information about the current state of that system and the data stored on it even when file signature evidence has been removed or replaced.

The very presence of high entropy data on a system may therefore prevent a suspect from plausibly denying the presence of encrypted information. However, file compression may result in similar high entropy encoding, requiring a more careful analysis. We have hence sought to characterise the relative entropy found for encrypted and unencrypted (including compressed) data, but also the effects of utilising different operating system versions as well as usage patterns. The research reported in this paper aimed to determine experimentally the extent to which measures of entropy and randomness can differentiate between encrypted and unencrypted data, different computer operating system versions and configurations and typical and atypical computer usage. A large number of Microsoft Windows workstation installations are run through a simulated ageing process consisting of —

- applying patches and updates,
- installing and configuring applications, and
- creating and deleting large numbers of data files of known types.

Forensic images of each workstation were captured at key points during the ageing process and entropy and statistical randomness measurements were taken from each image for each storage volume and every installed image file. In order to verify the results obtained through simulation, a number of images captured from production workstations are analysed and compared.

As Microsoft Windows is by far the dominant platform in terms of desktop systems deployed, we focus here on this platform; analogue experiments for the Linux environment have, however, been conducted with broadly comparable results, only some of these are reported here in the interest of completeness. However, the

*see e.g. the United Nations Office on Drugs and Crime report "The use of the Internet for terrorist purposes", Sep. 2012 discussing steganographic mechanisms used for covert communication by terrorist entities with a case study involving the Revolutionary Armed Forces of Colombia (FARC).

**We rely on the standard Shannon entropy in the following.

Based on "Forensic entropy analysis of Microsoft Windows storage volumes", by P. J.Weston and S. D. Wolthusen which appeared in the Proceedings of Information Security South African (ISSA) 2013, Johannesburg, 14 to 16 August 2013.

existence of several alternative file systems and more frequent significant release changes reduces the time series available for equivalent analysis. The remainder of this paper is structured as follows: In section 2. we briefly outline related work, followed by a description of the experimental setup in section 3. used for the subsequent analyses in sections 4. and 5. for file and volume analysis, respectively. We discuss key findings in section 6. before describing our conclusions and future work in section 7.

2. RELATED WORK

Research into signature and content analysis forms the basis of many file identification techniques, while work on multimedia data in particular is seen as vital to digital forensics. As this is a crucial pre-requisite for effective carving in the presence of fragmented or deleted data, file system behaviour allowing the effective grouping and identification of fragments has been studied by a number of researchers; Holleboom and Garcia investigated and performed experiments on information retention in slack-space for micro-fragments of previous files occupying the same clusters [2] with extensions by Blacher for the case of NTFS [3]; this also provides a further bound on the entropy of such clusters that is to be expected over the life-cycle of a frequently-reused storage medium.

A recent overview of the state of the art of multimedia forensic investigations is given by Poisel and Tjoa [4] while Ahmed *et al*. give examples of advanced methods used to improve file identification [5]. Shannon's analysis of ASCII and entropy scoring building on his namesake's work is of particular interest [6], as is recent work by Wu *et al*. showing how entropy and randomness testing can be used on encrypted images [7]; the tool `TCHunt` by 16 Systems identifies TrueCrypt images specifically by combining a search for volume (including sparse volumes) characteristics of TrueCrypt with a simple entropy analysis. Statistical analysis of file system clusters can yield insights on file types even for isolated clusters as discussed by Veenman [8]; for more specific file analyses, Lyda and Hamrock describe an entropy-driven approach for detecting encrypted malware, albeit relying only on block frequency (binning) to obtain a relatively coarse metric [9]. For the case of packed malware — which is beyond the scope of the present paper — this may not be sufficient if counter-forensic techniques are employed as recently described by Ugarte-Pedrero *et al*. [10]. This is also closely related to the need to predict the composition of file fragments; algorithms for which have e.g. been studied by Calhoun and Coles [11] with related approaches for classification described by Roussev and Garfinkel [12].

3. EXPERIMENTAL SETUP

The focus of the present work is, without loss of generality, on the Microsoft Windows operating system platform. Initially, eight production disk images were captured from pre-existing Microsoft Windows workstation installations using forensic capture tools. A total of 7 variants of Microsoft Windows 7, Vista, and XP were installed in default VMware Fusion 3.1 virtual machines (VM) using the VMware "*Easy Install*" wizard [13] at this time. At a later date, 2 variants of Microsoft Windows 7 and 8 were installed in default VMware Fusion 5.0 virtual machines and a Windows 8 production disk image was captured. The Boot volume of each installation was analysed. In all cases, the file system used was NTFS, and the cluster size set to eight 512 byte sectors. Except as noted, all storage sectors were zeroed prior to installation of the operating system instances. All images captured during this project were captured from Windows installations after the operating system had been shut down. Microsoft Windows workstation installations are categorised in this paper as either "Home" or "Business" depending upon the usage pattern and the installed applications and data. A common application suite — consisting of Microsoft Security Essentials, Adobe Acrobat and Flash Player, and VMware Tools — was installed on all virtual machine images. In addition, Oracle Java and Microsoft Office were installed on business VM images. Open Office, Mozilla Firefox and Thunderbird, Google Picasa, Apple iTunes and the Steam client application were installed on home virtual machine images to reflect different usage patterns. All applications were the latest versions at time of installation (June 2012 or September 2013). Document, music, picture, video and archive files were added to each virtual machine image; business virtual machine images contained relatively more document files, whilst home virtual machine images contained relatively more media files of different types. The relative proportions of each file type is the same in all cases but a different set of files were used for the later workstation images.

A total of 69 VM images were created through a simulated production life-cycle consisting of patching Windows, copying and deleting data files on the Boot volume, and exercising the installed applications. In ascending order of age, the simulated life-cycle stages are referred to in this paper as "*Initial*", "*Patched*", "*Base*", and "*Copyx*" (where *x* is the number of iterations). Captured images are referred to as "*Actual*" with a number identifying the image and a letter suffix indicating life-cycle stage where known ("a" is older than "b", etc.). Newer images are identified by the "new" suffix. So as to obtain images reflecting realistic longer-term use, the authors relied on images obtained from volunteers for validation. However, whilst all images and scripts utilised in obtaining the results described here can be made freely available on request, this does not apply to these validation images for privacy reasons. Encrypted data was obtained by creating TrueCrypt containers of various sizes using AES, Serpent and Twofish encryption algorithms and RIPEMD-160, SHA-512 and Whirlpool hash algorithms; AES with RIPEMD-160 was the default configuration. The same (weak) password was used in all cases, although all algorithms are sufficiently robust to eliminate influence on the entropy of encrypted data.

Data sectors were extracted from the volume images using tools provided in *The Sleuth Kit* [14]. Entropy and randomness calculations were performed on extracted data

	7-Zip File	7-Zip Cluster	TrueCrypt File	TrueCrypt Cluster
Count	237	1539	99	1024
Mean	7.999	7.955	8.000	7.955
Median	7.999	7.955	8.000	7.955
Min.	7.996	7.940	8.000	7.942
Max.	8.000	7.967	8.000	7.967

Table 1: 7-Zip and TrueCrypt Entropy (Bits/Byte)

at the byte level using the ent utility [15]. Where byte-level entropy calculations are impractical or inappropriate, data compression has been used as an analogue; all compression ratios reported in this paper result from GZIP compression at the "-4" compression level, utilising the Lempel-Ziv algorithm at its core [16], albeit utilising the DEFLATE format [17]. We note that not all applications were available for each version of the platform, resulting in some results not being available for all points of a time series; these data sets are reported as far as compatibility was retained. The results do not address possible changes of behaviour of users over time such as migrating from one application or platform to another as such behavioural changes and hypotheses underpinning behaviour were beyond the scope of the research.

4. FILE ANALYSIS

Entropy and randomness was measured at the byte-level for a statistically significant number of test files of various types: the mean and median number of files of each type analysed were 716 and 255 respectively with the minimum number of files of any one type being 11. Media and modern document file formats were found to exhibit high mean entropy in the range 7.270 to 7.981 bits/byte; most executables and older document formats exhibited lower mean entropy in the range 3.822 to 5.989 bits/byte. This reflects the use of improved compression algorithms in the newer file formats.

For file archives, entropy results reflect the relative performance of the compression algorithms used: LZNT1 shows the lowest mean entropy (947 files analysed, mean 7.558 bits/byte) and (G)ZIP and 7-Zip the highest mean entropy (96 files, mean 7.981 and 237 files, mean 7.999 bits/byte respectively). Encrypted TrueCrypt files consistently exhibit the highest possible byte-level entropy. TrueCrypt and 7-Zip clusters analysed in isolation, however, exhibit very similar (lower) entropy to each other (see Table 1).

χ-square, byte mean value, Monte Carlo π, and serial correlation values were calculated for all tested files. The χ-square test indicated that, with few exceptions, only 7-Zip and TrueCrypt files exhibit uniform randomness at the byte level. At the file level it was also noted that TrueCrypt containers consistently return χ-square results slightly closer to uniform randomness than the tested archive files; this does not apply when TrueCrypt containers are viewed at the cluster level (see Table 2).

	7-Zip File	7-Zip Cluster	TrueCrypt File	TrueCrypt Cluster
Mean	256.353	253.776	254.078	254.652
Median	258.247	253.375	254.073	253.750
Min.	190.077	182.750	204.452	185.500
Max.	324.218	340.375	322.160	326.500

Table 2: 7-Zip/TrueCrypt Chi-Square Statistics

4.1 Media File Analysis

Specific analyses were conducted for a number of file types including text, formatted text (XML, PDF), document (different Microsoft Office formats as well as Open Document format files), and media (image, video, and audio) data. Of particular interest in the context of the present paper are compressed file formats, which are characteristic of media data, but also more recent modern file formats noted above. Here, we have analysed basic descriptive statistics for file samples

	MP3	M4A
Count	6465	1088
Mean Entropy	7.967	7.981
Standard Deviation	0.041	0.011
Sample Variance	0.002	0.000
Kurtosis	334.617	80.982
Skewness	-15.283	-7.736
Minimum	6.710	7.822
Maximum	7.995	7.994
Confidence Level(95%)	0.001	0.001

Table 3: Entropy - Descriptive Statistics (Music files)

Table 3 shows that the compressed MP3 and M4A music file formats exhibit very high mean levels of entropy (similar to the entropy levels exhibited by compressed image file formats). Music file formats similarly also exhibit very little variance or deviation from their mean entropy value. For the music file types tested, χ-square randomness indicators have values well above values expected for random data (see Table 4). We can see easily that the tested music file formats do not exhibit randomness at the byte level; this is to be expected given the internal structure of the formats used, but yields a usable and efficient distinguishing feature. It is clearly necessary to perform this type of analysis as can be seen from the compression levels found in common file types; figure 1 provides a summary of the mean entropy for file types; this distribution is notably different once entropy is studied at the block or cluster level.

4.2 System File Analysis

Entropy and randomness values were calculated for Windows system and meta-data files at various points in the (simulated) Windows life-cycle. Entropy-frequency plots of the complete set of files forming each Microsoft operating system were found to be quite different to a

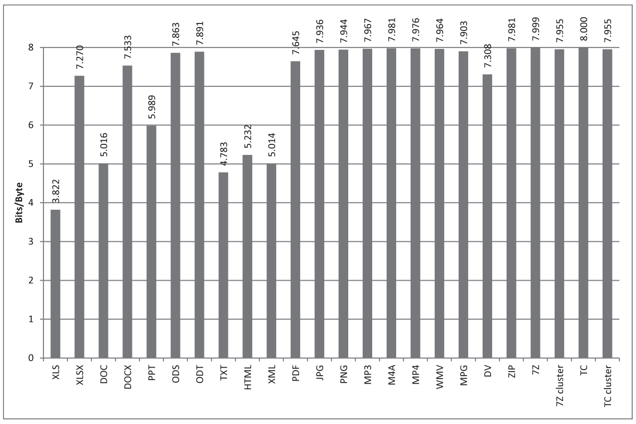

Figure 1: Overall File Mean Entropy (by Type). At least 1000 files per format were analysed.

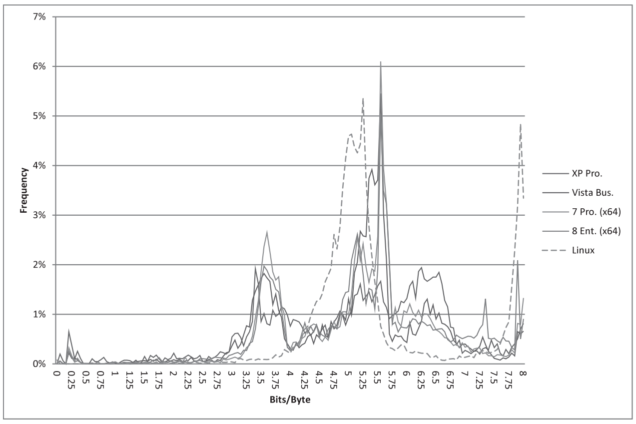

Figure 2: Operating System File Entropy (Initial Installation)

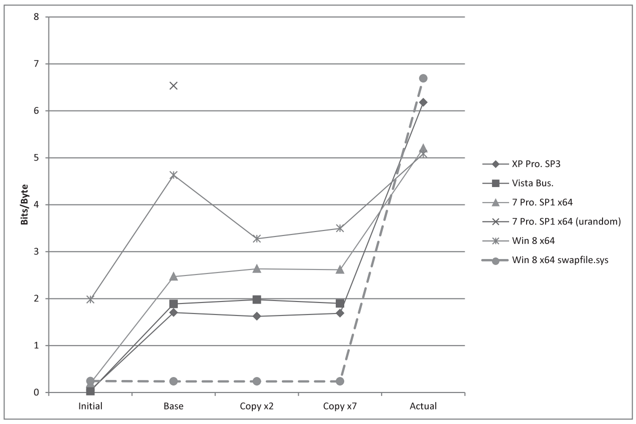

Figure 3: PAGEFILE.SYS File Entropy

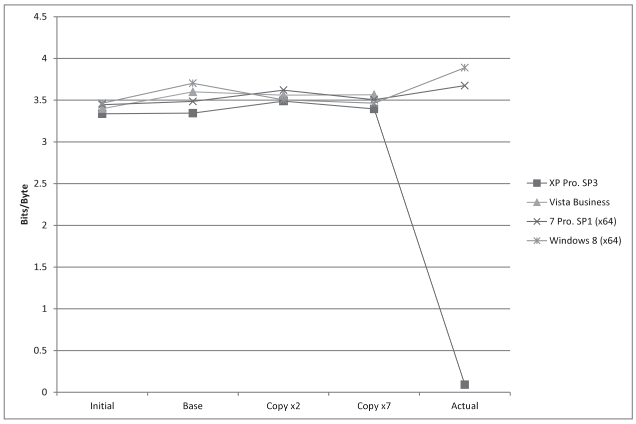

Figure 4: $LOGFILE File Entropy

similar plot of a recent Linux distribution. Figure 2 shows how different versions of Windows were observed to have relatively distinct entropy-frequency plots at initial installation. These distinctions tended to dissipate as data files were added to each installation although it was always possible to differentiate between Windows and Linux instances.

Windows uses the file `PAGEFILE.SYS` for virtual memory management. This paging file is not, by default, cleared when Windows is shut down [18] and hence entropy results can be expected to reflect data that has been swapped from main memory during Windows operation. Windows 8 "*Modern*" Apps use an additional `SWAPFILE.SYS` file to store their whole (private) working when suspended [19].

Figure 3 shows how the entropy of the Windows paging files generally increases over time. The test results indicate that the entropy of the paging file reflects the memory resources available on the host system. The Windows 8 (x64) test images, for instance, show significantly higher paging file entropy because they were generated on a virtual machine with half of the minimum memory requirement for this operating system. The greatest entropy of any paging file observed during testing was a "*Base*" Windows 7 image installed on a disk that had first been initialised by a UNIX urandom device. Paging files with unusually high entropy values may therefore be able to give some information about the history of a Windows system.

Windows 8 "*Modern*" Apps were not consciously exercised on the Windows 8 test images and consequently the Windows 8 `SWAPFILE.SYS` file exhibits extremely low entropy values on the test images. "*Modern*" Apps were, however, used on the production Windows 8 image and in this case swap file entropy is much higher. Entropy figures for the Windows 8 `SWAPFILE.SYS` file may therefore give some indication about the types of applications used on a Windows 8 sytem.

The NTFS journal attribute (`$LOGFILE`) is a fixed size circular log of 4 Kilobyte record pages where each journal page records the changes to be made to the file system [14, pp. 391-392]. Figure 4 shows that the NTFS journal entropy remains stable at around 3.5 bits per byte in most cases; this is to be expected given that the journal file is relatively small and journal pages are regularly reused. Note, however, that the Windows XP "*Actual*" journal contains few journal records and has an unusually low entropy. No explanation for this observation is available

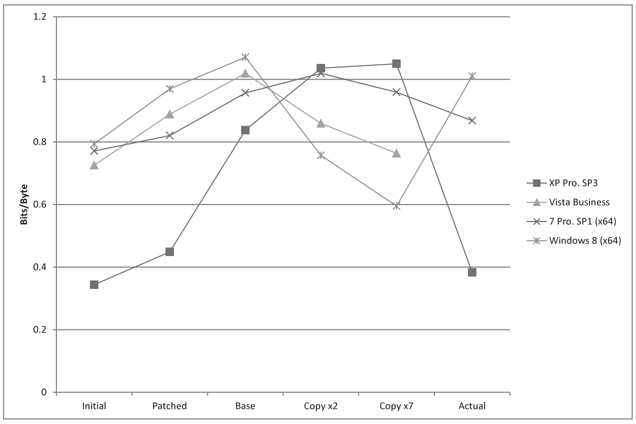

Figure 5: $BITMAP File Entropy

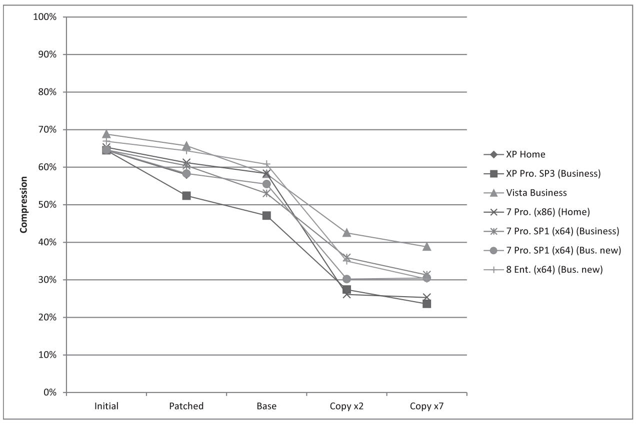

Figure 6: Boot Volume Allocated Space (GZIP)

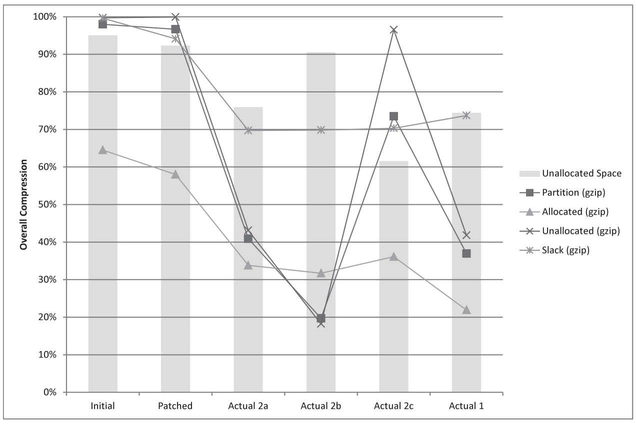

Figure 7: Boot Volume Compressibility (XP Home)

but such a significant deviation from the norm suggests that the journal file may be an area worthy of further investigation.

The entropy of the NTFS cluster allocation map attribute (`$BITMAP`) should remain low because Windows implements file allocation strategies that aim to minimise file fragmentation and performs regular de-fragmentation in the background. Test results confirm that the `$BITMAP` attribute has low entropy in all cases (see Figure 5).

5. VOLUME ANALYSIS

Storage clusters within the Windows Boot volume are flagged as either allocated or unallocated by the NTFS MFT `$BITMAP` attribute. Allocated clusters are known to contain current, live data whilst unallocated clusters may contain old, disused data. Sequence numbers within the NTFS MFT are incremented as MFT file and directory entries are (re)allocated and give an indication of the likelihood that unallocated and slack space will still contain the initial zeroed values. For the images described in this paper, maximum sequence numbers in the ranges 32 to 13015 and 11775 to 63655 were observed for the VM and production images, respectively.

Volume analysis was performed across the entire Boot volume. The MFT zone — a proportion of an NTFS volume reserved for MFT entries — is not considered in this analysis. When considering Windows XP volumes that have had more than 87.5% of their space allocated, it should be borne in mind that such volumes will have relatively fewer zeroed clusters than other XP volumes due to files having been allocated in the large MFT zone [20]. Figure 6 illustrates how the overall compressibility of allocated space on the Windows Boot volume decreases over time on all tested operating systems (i.e. entropy increases). The compressibility curve for all tested Windows versions begins to flatten in the 25–35% range as high entropy data files are added to the volume over time.

Overall compressibility of unallocated space on the Windows Boot volume decreases over time on all tested operating systems (i.e. entropy increases), as may be expected. Compressibility of unallocated space remains very high until used sectors containing high entropy data begin to be deallocated. The VM images show a smooth, gradual increase in the entropy of unallocated space as the Windows Boot volume ages. No such progression was seen on production images, however (see figure 7). The results for the "*Actual 2*" production Windows XP image demonstrate that it is possible for the entropy of unallocated space on a file system to be both lower or higher than allocated space entropy depending upon the usage history of the underlying media.

For the "*Actual 2*" image shown in figure 7, the allocated clusters from original media "*Actual 2a*" were copied onto media "*Actual 2b*" that had previously been used almost exlusively for media file storage. The file system was then subsequently copied onto new, zeroed media "*Actual 2c*". While both allocated and slack space compression ratios remained relatively constant during these relocations, compression ratios for unallocated space varied dramatically depending upon the original content of the new media.

The Microsoft NTFS file system stores data in fixed size allocation units called clusters. Files themselves, however,

	Chi-Square
MP3	9984.491
M4A	53119.803

Table 4: Minimum Chi-Square (Music files)

are seldom exact multiples of the cluster size and hence a certain proportion of the last cluster allocated to a file is not used and data could potentially be hidden there [21]. The unused space in a cluster is known as file slack. It is well-known that Microsoft Windows will fill unused space in the last sector into which data is written with zeros ("RAM slack"), but that it will not write data into any completely unused sectors in the cluster [14, 22].

Each file will therefore have one potentially lower entropy sector containing RAM slack plus potentially several further sectors which retain the data from whatever previously occupied them. In the case of a clean (zeroed) disk, therefore, slack space should overall have very low entropy because most of the sectors allocated to file slack are zeroed. Over time, however, as files are deleted and sectors are reallocated then the overall entropy of slack space should increase (although it should always remain comparatively low due to the zeroed RAM slack).

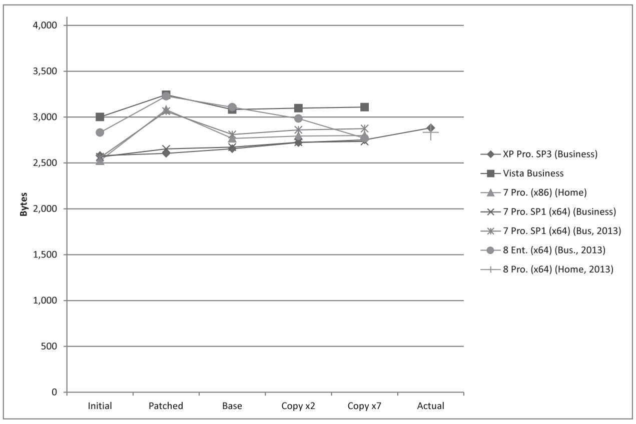

Figure 8: Mean Slack Space per File

Figure 8 shows that the mean slack space per file remains relatively constant over the life cycle of the Microsoft Windows operating system and applications, and that there is no significant difference in the mean value between versions of Windows investigated here. For all tested images the mean slack space per file on the Boot volume is well above the 2048 byte value that we would expect for purely random usage of 4096 byte clusters. This is potentially caused by Windows installations containing many files that are much smaller than half a cluster in size but may be an area worthy of further investigation.

Figure 9 shows that in the early part of the Windows life-cycle, slack space entropy — on initially zeroed storage media — is very low and then begins to gradually increase as the Windows installation ages and clusters are reallocated. The lowest slack space compression

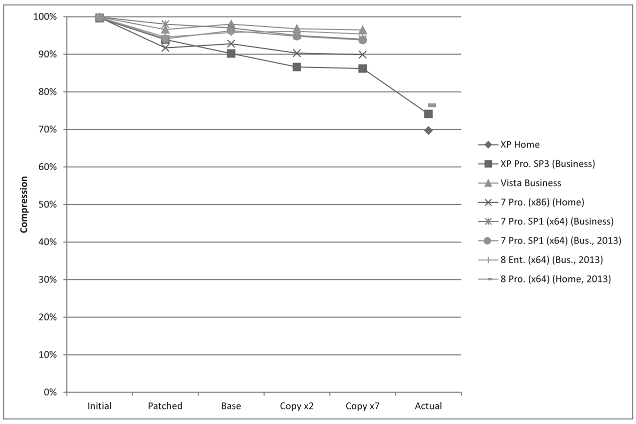

Figure 9: Slack Space Compressibility

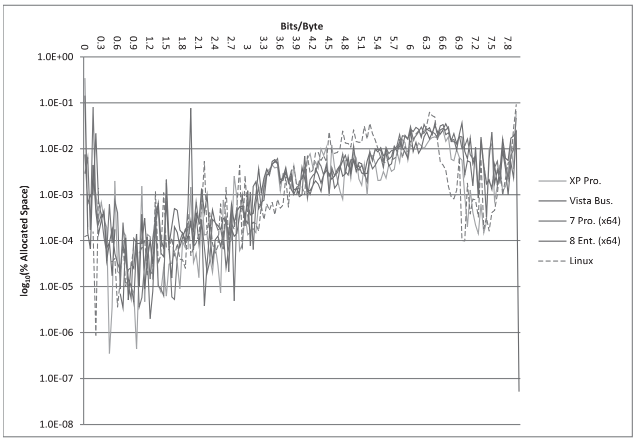

Figure 10: Allocation Unit Entropy (New Installation)

ratios were observed on the production Windows images and varied between 65% and 81%. In an attempt to identify an upper bound for slack space entropy, a boot volume was initialised to pseudo-random values (using a Unix "`urandom`" device) before Windows was installed; a slack space compression ratio of 57% was observed in this (approximate worst) case. We note that at the time of writing no actual volumes that had been in use for sufficiently long existed for Microsoft Windows 7, hence figure 9 only shows these data points for the case of Microsoft Windows XP volumes.

The "aging" of installation also is a potentially relevant element of information in that it not only affects the entropy of different elements of the volume such as overwritten but subsequently deleted or otherwise orphaned storage, but also serving as an indicator of an attempt to remove potential evidence by wiping a file system and subsequently replacing files; this may e.g. be the case if a system that had previously been infected with malware is replaced with a known good instance prior to the analysis taking place. Figure 10 shows the initial entropy per allocation units (normalised as bits per byte) plotted against the fraction of the volume occupied by allocation units (files) of this entropy.

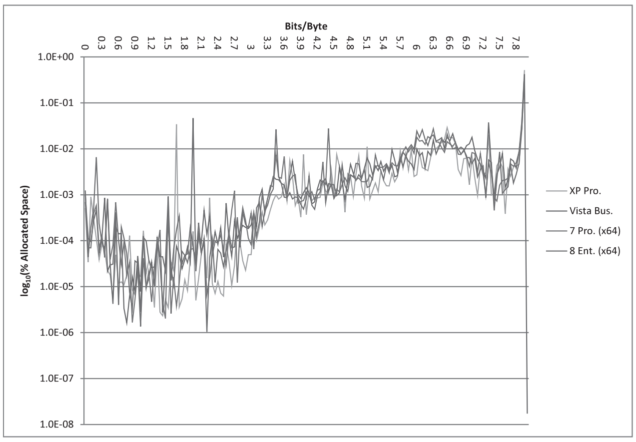

Figure 11: Allocation Unit Entropy (7 Generations of Use, Approx. 2 Years)

Figure 11 then shows the changes in entropy per allocation unit after seven aging iterations approximating 2 years of regular desktop use. Even without a detailed statistical analysis, the "aging" effect is clearly visible. However, whilst one would naïvely expect the entropy distribution to shift rightward, this is not necessarily the case.

We note that figure 10 also contains a plot for the initial distribution for a Linux installation; resource limitations did not allow analogous experiments to be conducted for Linux, so only results for different Windows variants are reported here.

6. ANALYSIS

A combination of entropy and randomness testing appears to be capable of detecting encrypted files from file content alone. Encrypted and highly compressed data prove to be effectively indistinguishable when only small amounts of each are analysed; a size boundary for reliable differentiation is not established in this paper.

Frequency analysis of the entropy of files on a Boot volume can give an indication about which type of operating system is installed. The system files located on a Windows Boot volume have relatively low entropy. High overall entropy in both allocated and unallocated space therefore indicates the presence of significant quantities of (high entropy) user data on the volume.

Allocated space compression ratios of around 30% appear to be typical for production Windows Boot volumes that have been used for some time; ratios significantly below this can be considered anomalous. Unallocated space on a Windows Boot volume does not appear to have a "typical" entropy value. An unallocated space entropy value significantly lower than the typical allocated space entropy value may merit further investigation.

Unallocated space with very low entropy indicates a recently created volume or one that has been deliberated wiped. When data is copied between volumes, it is likely that slack space will be transferred but unlikely that unallocated space will be transferred; this may lead to a mismatch between slack space entropy and unallocated space entropy from which we might infer a transfer or volume sanitisation.

Slack space compression of 65-75% is typical for a Windows installation that has been used for some time. Slack space compression ratios below 60% are unlikely to occur in normal Windows operation and would merit further investigation.

7. CONCLUSIONS

The results presented in this paper demonstrate that entropy and randomness measurements may be able to differentiate between encrypted and unencrypted data files with a reasonable degree of confidence, permitting automation. These same measurements may also help identify atypical Windows usage such as volume copies, volume wipes and unusually high entropy slack space. A strong result obtained in the analysis is the correlation of the compression ratio for slack space with the age of an installation, as any anomaly is likely to warrant further investigation. For the case of files we have found that — provided sufficiently large files are available for analysis — a combination of entropy and randomness tests will suffice to identify characteristics of encrypted data without having to rely on meta-data.

An adversary aware of these findings may attempt to cast doubt upon these measurements by highlighting that similar results can be obtained when analysing highly compressed files. When small amounts of data are involved then this can be an effective defence because entropy and randomness tests struggle to reliably differentiate encrypted and highly compressed data. For larger amounts of data, however, an adversary may be forced to use alternative defences such as filling unallocated storage with high entropy data and monitoring slack space entropy. Such countermeasures may themselves be identified as atypical usage which trigger further investigation.

Future work will seek to study the applicability of the results reported here to other types of (local) file systems and newer editions of the operating systems studied. We are particularly interested in analysing the characteristics of newer, log-based file systems as this has thus far not been studied to the best of our knowledge.

A further natural extension of the work described here is also the development of counter-forensic mechanisms that either avoid yielding tell-tale signatures identified, or to provide extensive decoys to increase the work-load and extent of manual investigation required as well as creating a plausible deniability scenario. Attention has been paid in the present work to facilitate automation of measurements as far as possible; it appears to be highly desirable to repeat measurements particularly at the file analysis level regularly as changes in file formats and e.g. encoders in case of multimedia files may change over time, skewing results.

The data and mechanisms used in generating the results described here are freely available from the authors subject to licensing conditions for the software used in the image files themselves except where privacy restrictions do not permit the release of personally identifiable information.

REFERENCES

[1] H.M. Government, "Regulation of Investigatory Powers Act 2000," Her Majesty's Stationery Office and Queen's Printer of Acts of Parliament, Jul. 2000.

[2] T. Holleboom and J. Garcia, "Fragment Retention Characteristics in Slack Space – Analysis and Measurements," in *Proceedings of the 2010 2nd International Workshop on Security and Communication Networks (IWSCN 2010)*. Karlstad, Sweden: IEEE Press, May 2010, pp. 1–6.

[3] Z. Blacher, "Cluster-Slack Retention Characteristics: A Study of the NTFS File System," Master's thesis, Department of Computer Science, Karlstad University, Karlstad, Sweden, Jun. 2010.

[4] R. Poisel and S. Tjoa, "Forensics Investigations of Multimedia Data: A Review of the State-of-the-Art," in *Proceedings of the Sixth International Conference on IT Security Incident Management and IT Forensics (IMF 2011)*, D. Guenther and H. Morgenstern, Eds. Stuttgart, Germany: IEEE Press, May 2011, pp. 48–61.

[5] I. Ahmed, K.-S. Lhee, H. Shin, and M. Hong, "On Improving the Accuracy and Performance of Content-Based File Type Identification," in *Proceedings of the 14th Australasian Conference on Information Security and Privacy (ACISP 2009)*, ser. Lecture Notes in Computer Science, C. Boyd and J. González Nieto, Eds., vol. 5594. Brisbane, Australia: Springer-Verlag, Jul. 2009, pp. 44–59.

[6] M. M. Shannon, "Forensic Relative Strength Scoring: ASCII and Entropy Scoring," *International Journal of Digital Evidence*, vol. 2, no. 4, pp. 1–19, Apr. 2004.

[7] Y. Wu, Y. Zhou, G. Saveriades, S. Agaian, J. P. Noonan, and P. Natarajan, "Local Shannon Entropy Measure with Statistical Tests for Image Randomness," *Journal of Information Sciences*, 2012, (article in press).

[8] C. J. Veenman, "File Fragment Classification — The Case for Specialized Approaches," in *Proceedings of the Third International Symposium on Information Assurance and Security (IAS 2007)*. Manchester, UK: IEEE Press, Aug. 2007, pp. 393–398.

[9] R. Lyda and J. Hamrock, "Using Entropy Analysis to Find Encrypted and Packed Malware," *IEEE Security & Privacy*, vol. 5, no. 2, pp. 40–45, Mar./Apr. 2007.

[10] X. Ugarte-Pedrero, I. Santos, B. Sanz, C. Laorden, and P. Garcia Bringas, "Countering Entropy Measure Attacks on Packed Software Detection," in *Proceedings of the 9th Annual IEEE Consumer Communications and Networking Conference — Security and Content Protection*. Las Vegas, NV, USA: IEEE Press, Jan. 2012, pp. 164–168.

[11] W. C. Calhoun and D. Coles, "Predicting the Types of File Fragments," *Digital Investigation*, vol. 5, no. (supplement), pp. S14–S20, Sep. 2008.

[12] V. Roussev and S. L. Garfinkel, "File Fragment Classification — The Case for Specialized Approaches," in *Proceedings of the 4th International Workshop on Systematic Approaches to Digital Forensic Engineering (SADFE 2009)*. Berkeley, CA, USA: IEEE Press, May 2009, pp. 3–14.

[13] VMWare, Inc., *Getting Started with VMware Fusion*, http://www.vmware.com/, Palo Alto, CA, USA, 2011, Last accessed 1 Nov. 2012.

[14] B. Carrier, *File System Forensic Analysis*, 1st ed. Reading, MA, USA: Addison-Wesley, 2005.

[15] J. Walker, "ENT – A Pseudorandom Number Sequence Test Program," http://www.fourmilab.ch/random/, Jan. 2008, Last accessed 1 Nov. 2012.

[16] J. Ziv and A. Lempel, "A Universal Algorithm for Sequential Data Compression," *IEEE Transactions on Information Theory*, vol. 23, no. 3, pp. 337–343, May 1977.

[17] L. P. Deutsch, "DEFLATE Compressed Data Format Specification version 1.3," IETF Network Working Group Request for Comments RFC 1951, May 1996.

[18] Micrsoft Corporation, "How to Clear the Windows Paging File at Shutdown," http://support.microsoft.com/kb/314834, Oct. 2010, Last accessed 1 Nov. 2012.

[19] ——, "Windows 8 / Windows Server 2012: The New Swap File," http://blogs.technet.com/b/askperf/archive/2012/10/28/windows-8-windows-server-2012-the-new-swap-file.aspx, Oct. 2012, Last accessed 28 Oct. 2012.

[20] ——, "How NTFS reserves space for its Master File Table (MFT)," http://support.microsoft.com/kb/174619, Oct. 2008, Last accessed 1 Nov. 2012.

[21] E. Huebner, D. Bem, and C. Kai Wee, "Data Hiding in the NTFS File System," *Digital Investigation*, vol. 3, no. 4, pp. 211–226, Dec. 2006.

[22] D. M. Purcell and S.-D. Lang, "Forensic Artifacts of Microsoft Windows Vista System," in *Intelligence and Security Informatics: Proceedings of the IEEE ISI 2008 International Workshops: PAISI, PACCF and SOCO 2008*, ser. Lecture Notes in Computer Science, C. C. Yang, Ed., vol. 5075. Taipei, Taiwan: Springer-Verlag, Jun. 2008, pp. 304–319.

MOVING REPUTATION TO THE CLOUD

C. Hillebrand* and M. Coetzee†

* *Academy of Computer Science and Software Engineering, University of Johannesburg, South Africa*
SAP P&I BIT Mobile Empowerment, Pretoria, South Africa
E-mail: channel.hillebrand@gmail.com
† *Academy of Computer Science and Software Engineering, University of Johannesburg, South Africa*
E-mail: marijkec@uj.ac.za

Abstract: Reputation is used to regulate relationships of trust in online communities. When deploying a reputation system, the requirements and constrains of the specific community needs to be accommodated in order to assist the community to reach their goals. This paper identifies a need for a framework for a configurable reputation system with the ability to accommodate the requirements of a variety of online communities. Such a reputation system can be defined as a service on the Cloud, to be composed with the application environment of the online community. Consequently, this paper introduces the concept of RaaS (Reputation-as-a-Service) and discusses a potential framework to support the creation of a RaaS. In order to define the framework, research is conducted into features of SaaS (Software-as-a-Service) architecture components, user requirements for trust and reputation, and features of current centralized online reputation frameworks that can be configured in order to support a reputation service on the Cloud.

Key words: Reputation-as-a-Service, trust, reputation, Software-as-a-Service, reputation framework, cloud service.

1. INTRODUCTION

Online shopping has grown significantly in the past years and it is predicted that such sales will increase annually by 10% for the next 4 years [1, 2]. People are influenced by product reviews to make purchasing decisions and therefore tend to buy from online stores with a good reputation [1]. As online shopping is characterized by insecurity, anonymity, lack of control and potential opportunism, online communities should take the necessary steps to ensure that participants are trustworthy.

For online trading communities such as eBay, a centralized online reputation system is used to compute and publish reputation scores for service providers, services, products or entities such as buyers and seller within a community. The reputation score reflects the collection of opinions or ratings that entities have about the objects. Ratings are provided to a reputation algorithm to compute reputation scores [3]. In order to be effective, reputation managers need to accommodate the specific needs of the communities where they are deployed.

Consider the example of Organization ABC, an online store for a start-up company that sells products to consumers over the mobile web. As trust and reputation is a major component to enable m-commerce, the online store of Organization ABC needs to deploy a reputation system to control trust relationships between consumers, suppliers and their portal. As there is no off-the-shelf reputation system to integrate into their application environment, and it is expensive to custom develop, the m-commerce web site may initially be implemented without it. Ideally, Organization ABC needs a reputation system that is simple to use, with easy to understand ratings between 0 and 5 to ensure the growth of the community. In another type of online community, where crime incidents are posted and recorded with mobile phones, a reputation system is needed to ensure that no malicious or false incidents are reported. The requirements for this reputation system may be very different to those of the online store of Organization ABC. This highlights that a configurable or customizable reputation system is needed that can support multiple online communities in a cost-effective and efficient manner.

Recently, a business model for software applications namely SaaS (Software-as-a-Service) has emerged which lowers the cost of development, customization, deployment and operation of applications [4]. As SaaS applications generally support the concept of software application configuration and customization, this research proposes to present a configurable reputation system as a SaaS solution. Here, a multi-tenant architecture is followed where organizations pay only for the features that they access, and are able to configure or customize the reputation system to suit their community's needs.

The contribution of this paper is to identify requirements and challenges in order to define a RaaS (Reputation-as-a-Service) framework. As trust and reputation systems can be very complex, the focus of this research is the definition of a RaaS framework that provides similar but configurable functionality currently supported by central online reputation systems.

In the next section, trust and reputation is defined for this research. Five general components of reputations systems are given which is referred to throughout the paper. The requirements for a RaaS component is identified by considering SaaS configuration aspects, user requirements for trust and reputation and finally requirements from

Based on "Towards reputation-as-a-service", by C. Hillebrand and M. Coetzee which appeared in the Proceedings of Information Security South African (ISSA) 2013, Johannesburg, 14 to 16 August 2013. © 2013 IEEE

reputation frameworks. A RaaS framework is presented and the paper is concluded.

2. TRUST AND REPUTATION

Trust and reputation is present in a variety of online communities. Trust is the individual's perspective on a particular service or product and reputation is a group's perspective on a particular service or product [5]. As trust and reputation are concepts that are often used interchangeably, they are now defined for the purposes of this research.

2.1 Trust

Trust is challenging to define as it manifests itself in many different ways in varying contexts. Almost every aspect of daily life is supported by some form of trust. For example, in Figure 1, consumer X, the trustor, orders products from organization ABC, the trustee. For this research, the following definition of trust is adopted. The trust of consumer X in organization ABC is defined as the level of subjective probability that organization ABC will deliver high quality products on time [6].

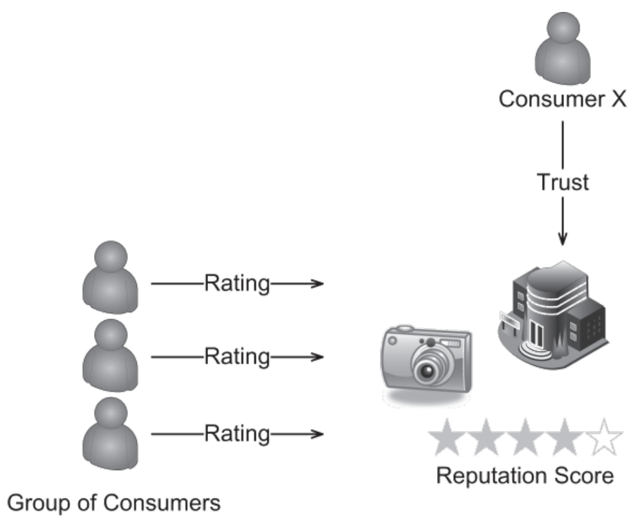

Figure 1: Trust and Reputation

The trust of consumer X in Organization ABC is affected by trust properties such as transitivity, subjectivity and the asymmetric nature of trust [7]. If Organization ABC has the reputation of delivering high quality products, consumers automatically assume that any product of Organization ABC is also of high quality due to the property of transitivity, suggesting that trust is transferable. But, as both consumer X and Z can have different levels of trust towards the same organization ABC, trust is subjective. The asymmetric property of trust is defined by the fact that consumer X needs to trust that organization ABC will deliver the necessary services, but organization ABC needs to trust consumer X to pay on time.

Closely related to trust, is reputation. In the next section the concept of reputation is addressed in order to identify elements that it consists of.

2.2 Reputation

Reputation can be considered as a collective measure of trustworthiness [8]. In order to better regulate relationships in online communities, opinions about interacting parties' past behavior is collected and aggregated in order to define a summary evaluation, or reputation. The predictive power of reputation supposes that past behavior of a participant is indicative of their future behavior.

In Figure 1, reputation is illustrated by a group of consumers' opinion on a specific product. The group of consumers in Figure 1 gives a product a good rating over time, ensuring that the product has a good reputation score [9]. In this paper the term "rater" is used to represent a participant or consumer who assigns ratings for others. Reputation is calculated by incorporating past experiences, direct experiences and recommendations using various algorithms and models for this purpose [10]. One party can thus trust another based on their "good" or "bad" reputation.

A reputation system is an application that facilitates the process of calculating and evaluating reputation for a specific community. The five main components found in reputation systems and models are [11]:

1. *Gathering behavioral information* where direct experiences, and experiences of acquaintances of consumers, recommendations from others, transaction history, pre-trusted entities and raters reliability are collected.

2. *Scoring and ranking* of entities are done next resulting in a reputation score, computed using averages, fuzzy logic, or Bayesian networks.

3. *Entity selection* is done next using the reputation score and other utility functions as specified.

4. *Transaction* is carried out with the selected entity.

5. *Reward and punishment* is finally given by assessing the transaction and giving a rating.

Most current reputation systems are built using these common components, but for a specific context and application domain, using proprietary vocabularies [12]. Each defines its own method to query, store, aggregate, infer, interpret and represent reputation information.

In order to be able to define a cloud-based reputation service that can be usable by different communities, the next section investigates the requirements that such a reputation service framework or RaaS framework should comply to.

3. REQUIREMENTS FOR A RAAS FRAMEWORK

In order to determine requirements to define a RaaS framework, this research now reports on the two main drivers for a RaaS framework namely general SaaS application requirements and current state-of-the-art reputation system requirements to create a comprehensive list of requirements for the RaaS framework.

3.1 Requirements for SaaS applications

SaaS applications are deployed on cloud infrastructures and exposed to applications or users to be consumed over the Internet. SaaS is pay-per use, meaning organizations only have to pay for the features they want to use in an application, making it a cost effective solution [13, 14]. The main motivation for organizations adapting SaaS applications is to reduce IT support costs, ensure business agility and outsource hardware and software maintenance. SaaS architecture, design attributes and application requirements are now investigated to identify requirements that a cloud reputation service needs to meet.

SaaS application architecture: A SaaS application such as a reputation service or RaaS is integrated with existing enterprise systems, as shown in Figure 2. The architecture is based on service-oriented architecture design principles.

To create a cohesive application that incorporates a RaaS, the orchestration of the flow of data from the service to the end user is crucial. For example, it is important to send data such as a valid rating for a specific product to the right retail system at the right time. Furthermore, SaaS-to-enterprise integration poses challenges such as semantic mediation, data quality, interface mediation and other logical operations when moving data between domains so that the data is useable when it reaches the target system. Depending on the business requirements and integration capabilities of the chosen SaaS product, the integration approach is generally not trivial. Even though SaaS applications are exposed via comprehensive APIs to ease the integration process, it may still be the case that a custom SaaS integration layer is needed.

To support application integration, SaaS application architecture generally consist out of 3 main layers namely the consumption, service and data layer [13]. Starting at the bottom of Figure 2 the service and data layers are found in the *SaaS component*. Here are reusable software components and their data are exposed as services preferably using REST APIs (Representational state transfer) [15]. Messages are formatted in in JSON (JavaScript Object Notation) [16] format style or in XML (Extensible Markup Language) [17] format.

The service layer is defined by multiple sub-layers such as the service wrapping, schedule and service technology application layers. Support is provided to administrators of tenants to customize services. To persist SaaS specific data, the data layer stores rating, reputation and other relevant information.

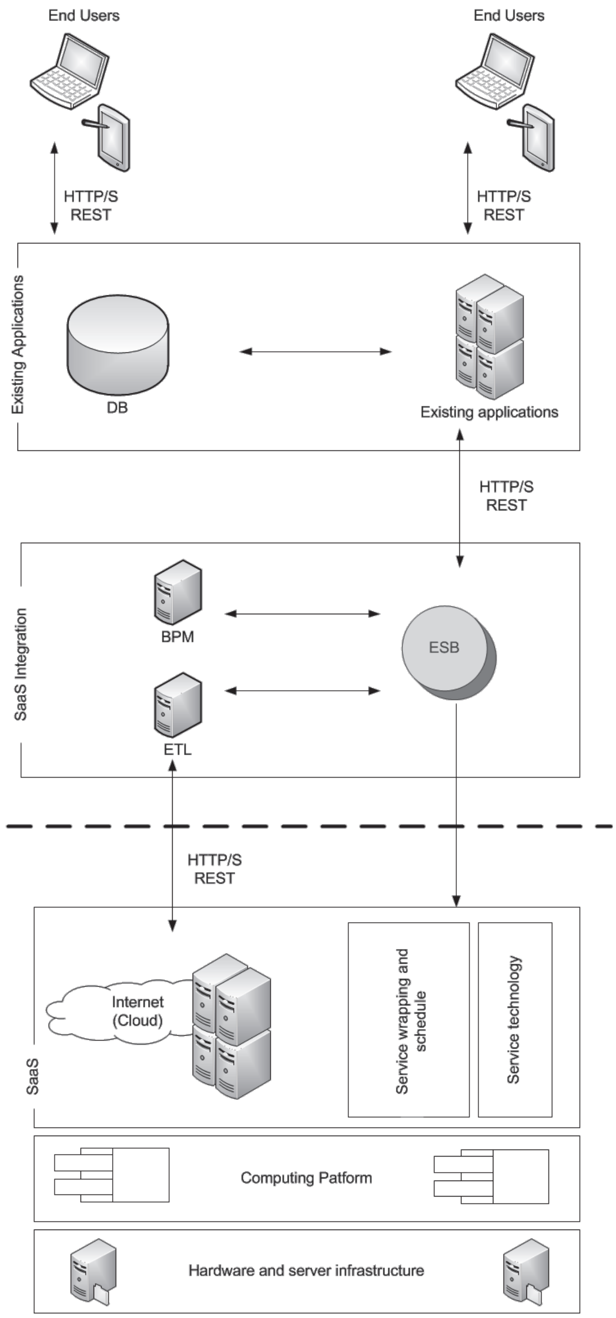

Figure 2: SaaS Architecture Layers

Towards the middle of the architecture, enterprise integration is supported by the *SaaS integration component* using integration products such as an ESB (Enterprise Service Bus) that provides orchestration capabilities. In this way, enterprise activities performed by existing enterprise applications are combined with activities of SaaS applications.

Finally, the consumption layer presents end users with an integrated view of information generated by these orchestrated applications.

To be successfully integrated into other enterprise applications, an important requirement for SaaS applications is to ensure that integration with other applications is seamless. To ensure that many types of enterprises can integrate SaaS applications, design attributes, discussed next should be addressed.

SaaS design attributes: In order to design a RaaS component, several design attributes should be taken into consideration to ensure conformance to a typical SaaS application [18]:

- Multi-tenancy to ensure that a single instance of a RaaS can be used by multiple tenants and their clients with different needs and functionalities.
- Single version provides the capability that only one version of the RaaS is exposed to clients.
- Logical data separation accommodates the storage of different tenants' data such as configuration and rating data is in their own data domain.
- Application integration, as mentioned in the previous section, should be supported so that applications of tenants can integrate functions of the RaaS into their applications with ease.

To ensure the flexibility of SaaS applications, an application is configured in such a way to support design attributes and integration processes. Next, SaaS application requirements for RaaS are discussed to support configuration.

SaaS application requirements for RaaS: The focus of a RaaS model is to deliver software functions to many clients over the Web with a single instance of a software application running on a multi-tenancy platform [19]. However, every tenant that needs to use a RaaS supported with this model can be unique, requiring changes to the reputation system.

Tenants may have a different industry focus, their customers may behave differently, they may support diverse product offerings and have different regulations, organizational culture and operational strategy. These features require RaaS components to be tailored, by leveraging two major approaches namely configuration and customization [19].

Configuration does not involve source code change of the RaaS application and support differences through setting pre-defined parameters, or leveraging tools to change application functions within pre-defined scope, such as adding data fields, changing field names, modifying drop-down lists, adding buttons, and changing business rules. On the other hand, customization involves RaaS application source code changes to create functionality, leading to a more costly approach for both SaaS vendors and clients.

There are seven fundamental configuration and customization requirements that can be tailored, to make the RaaS component as flexible as possible [19] namely:

- Support for different organization structures require the ability to add, delete and changes roles.
- Support for different types of data can be made possible by adding custom fields and types, and deleting data not needed.
- Support for different processes requires tasks to be switched, added and reordered and their roles to be changed.
- Business rules can be modified by changing or setting rules and the rule triggers.
- Reputation computations can be made more generic by adding or changing actions or triggering actions at different points.
- The user interface can be changed with respect to the look and feel, the data presented and the addition of data.
- Reporting can be changed with respect to style, dataset used and query rules.

In summary, the RaaS should be developed to have standardized software features to serve as many clients as possible using a configuration approach. The RaaS developer needs a strategy to enable self-defined configuration by their tenants without changing the SaaS application source code for any individual tenant [19].

The RaaS environment needs to be thoroughly analyzed to determine the common configuration requirements. In conjunction, a sophisticated web based tool is needed to allow clients to configure the RaaS service themselves.

The next step is to investigate the second main driver to determine RaaS requirements, namely those stemming from trust and reputation systems.

3.2 Trust and reputation system requirements for RaaS

Centralized online reputation systems, which is the focus of this research, collects users' opinions on products, transactions and events as reputation information, to aggregate and publish it. Many trust and reputation models have been proposed, each targeting different contexts, with their own unique features. While most research focuses on addressing the ever-increasing complexity, not much attention has been paid to the process of integrating reputation systems into applications. The next section has the aim of identifying a set of basic requirements to be addressed the RaaS by firstly investigating real-world user requirements, and then general trust and reputation system components.

User requirements for trust and reputation systems: Previous research [10] collected formal user requirements for trust and reputation systems from system developers. It was found that users required a clear, layered and pluggable architecture for representing the calculation process of the trust score. Categorized user requirements were found to be closely coupled with the previously discussed five components found in reputation systems [20].

User needs for each of the first three components were identified as follows:

- Information Gathering
  - The success of each interaction needs to be rated and quality parameters continuously monitored.
  - Simple and intuitive rating scales should be used.
  - The quality parameters of a service should be controlled and certified by a trusted party; ratings of such a party can be used as a starting point for trust computation.
  - Raters reliability must be controlled as they could provide dishonest ratings.
  - Initial rating should not influence or bias subsequent votes.
  - Similarity between recommenders' preferences should be considered.
  - Trust values decay and become invalid over time.
- Scoring and Ranking
  - There is a need for a single trust rating which is calculated by taking into account different service aspects and their weights.
  - The calculation should be an aggregation of all weighted aspects, similar to an "average".
- Entity Selection
  - When entities or services are selected, they should be sorted according to their trust rank and made comparable to each other.

By considering such a user-centered design approach, the proposed RaaS component can be created to fulfill the needs of users as far as possible. General reputation framework requirements are discussed next.

Reputation system framework requirements for RaaS: The focus of this section is to identify major characteristics of reputations systems to identify requirements for the RaaS component. In order to achieve this, an adapted framework is defined from the work of others [21–23]. There are eight elements which are discussed following the phases of reputation management components from information gathering, to scoring and ranking and entity selection. The elements, shown in Figure 3 include:

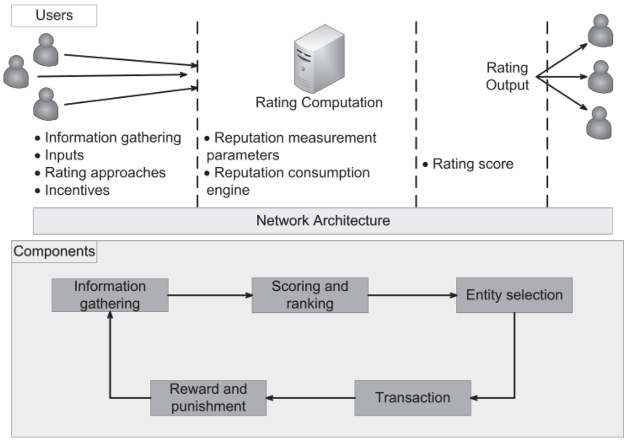

Figure 3: Reputation Manager Framework

1. Network architecture
2. Information gathering
3. Inputs
4. Rating approaches
5. Incentives
6. Reputation measurement parameters
7. Reputation computation engines
8. Rating score

Figure 3 shows how the eight elements at the top of the diagram fit in with the first three components at the bottom. Network architecture supports the framework across all components, whereas information gathering, inputs and rating approaches and incentives are found under the *Information Gathering* component. Reputation measurement parameters and reputation computation engines fall under the *Scoring and Ranking* component and rating score under the *Entity Selection* component.

Each of these elements are now described, starting with the network architecture.

a) Network architecture: The network architecture of a reputation system can either be a centralized, decentralized or hybrid architecture [8]. The network architecture determines how information is gathered and stored. For this research, the RaaS component needs to follow the cloud architecture, thereby limiting the scope of architecture choice. In a centralized architecture, all data is stored in a central repository with all reputations scores publicly available to participants. For distributed or hybrid reputation systems, there is no central point where ratings are submitted or feedback can be obtained. Instead, each participant is given the responsibility to collect ratings from others. For a RaaS component this is not a viable option as the cost and complexity level would be too high. A centralized architecture is simple and cost-efficient, and conforms to the RaaS and user requirements identified previously. This choice directly influences the discussion of the next elements, as they need to comply with this requirement.

Next the elements related to the *Information Gathering* component are discussed. For this research, a transaction between two participants is the basis of a rating. Generally, a participant cannot rate another one without having had a transaction with him. After a transaction, participants usually have no direct incentive for providing rating about the other party. The information gathering phase should be carefully designed to address this issue.

b) Information gathering: The information gathering phase collects rating inputs over a period of time. There are a numbers of important aspects to consider such as [24]:

- the collection channel,
- the information sources,
- the number of raters, rating granularity and reputation of raters,
- collection costs.

Collection channels can be direct or indirect. Direct channels collect information from raters just after the transaction, by sending emails asking them to do a rating or by using a 3rd party for rating collection. Indirect channels collect information from other reputations systems, increasing complexity of information gathering.

Most online reputation systems consider reputation ratings from a global perspective. For example, eBay's feedback forum provides feedback profiles of sellers and buyers publicly to the all users. The shortcoming of such a global view is that these values lack personalization [5]. Information can be gathered from past experiences, direct experiences and recommendations [20]. The gathered information plays a major role to calculate a reputation score for a particular user or product. This information might come from several sources such as direct experiences with the targeting entity, neighbours of participants, acquaintances, the group the participant belongs to or organizations. In this regard it is important to consider the set of raters, their expertise and credibility [24].

A sufficient number of raters who rate transactions can help a reputation system to avoid personal bias whereas a restriction on the number of raters may influence level of detail between raters and objects being rated. A reputation system can be defined to have no restrictions on the number of raters leaving ratings, which means anyone can rate; or only registered participants can provide a rating; or only some registered raters can provide a rating after a transaction has finished such as eBay allows. It should not be allowed to rate a transaction or object more than once, for example, in eBay if buyers and sellers transact, the reputation system will only allow one rating per transaction to avoid the manipulation of the reputation score.

Directly related to the number of raters who rated objects is granularity [25], which indicates if the model is context-dependent or not. As raters may have a good reputation for their expertise in one domain, and a low reputation for another, granularity identifies how information sources associates to the reputation object. When a system allows any raters to do a rating, the granularity is usually very loose. If a reputation system requires information sources to have a good credibility to leave reviews this increases the cost for a rater to provide a rating which in turn reduces the number of invalid ratings.

The reputation of the rater should be considered by having other participants to give feedback on those ratings. Some reputation systems have a ranking mechanism for their users, called the Karma mechanism that records every action of a user and gives points to it [23].

Finally, the input collection costs should be considered. This is the cost that indicates how much time it takes to collect a single unit of reputation information, where collection channels can have an important effect on this cost [24].

Next, the type of information source is described.

c) Inputs: Different information formats can be chosen based on the way in which they will be used in a reputation system. Some reputation systems support arithmetic operations and other evidence where numeric quantification is more appropriate. It can also be possible to provide a mapping from qualitative to numeric labels. For example, ratings such as a score between 0 and 10 can easily be aggregated to an overall score, to give a comparable value between reputation objects. On the other hand, text reviews contain detailed information which can be very useful.

Generally, a rating can be expressed as either a quantitative or Boolean format [21]. A quantitative metric is a measurable input such as a value between 0 and 10 whereas a Boolean format is either 0 or a 1 to represent "like" or "dislike". As it is important that the reputation score is useful to the community where it will be used, the RaaS can be configured for this purpose.

In order to ensure the completeness of ratings collected, rating approaches are discussed next.

d) Rating approaches: A larger variety of rating information can give a better view of a reputation object as it provides a more complete picture. For example, travel reputation systems can allow participants to rate hotels for their value, rooms, location, cleanliness and service separately [24].

In single-criterion rating systems or binary rating systems, participants reveal their general opinion with regards to a reputation object, resulting in reputation information that is not too reliable and accurate.

In systems where multiple-criteria can be used, better quality reputation scores can be defined. A set of criteria needs to be defined and a rating is provided for each. This can allow a participant to choose a partner based on specific criteria that matches his own. On the down side, many rating criteria may reduce the evaluators' motivation on leaving ratings. This can be overcome by making some criteria optional to rate.

Next the role of incentives in information gathering is discussed.

e) Incentives: Raters of a reputation system may have different motivations for providing ratings. Incentives are important as their absence drives only some of the users to voice their opinions and report feedback where those with a moderate outlook are unlikely to provide ratings [25]. This results in an unrepresentative sample of ratings and opinions. For example, reputation systems have incentives for raters such as sellers to behave honestly in order to be chosen by buyers as this can increase their profit through the increased amount of transactions. These incentives are necessary because fabricated ratings can promote specific sellers or to discredit others - e.g. authors can write fake reviews on Amazon in order to boost the sale of their own books. In order for RaaS to be implemented successfully, the motivations for providing a rating should be identified.

There are various types of motivations [23] such as altruistic motivation which is in favour of doing good to users being rated and can be classified as tit-for-tat, friendship and exploiting opinionated incentives. Commercial motivation, is used to generate revenue and is categorized as direct revenue incentives and branding incentives. Egocentric motivation is used for self-gratification and is categorized as fulfilment incentives and recognition incentives.

By explicitly rewarding participants for reporting feedback, rewards made by the reputation systems must cover the cost of reporting feedback to encourage more participants to report, giving a more representative set of ratings. In addition, rewards must be designed so that selfish participants are convinced to rate truthfully to advance themselves [25].

The next section now considers the next reputation component namely the *Scoring and Ranking* of ratings. Here, the reputation computation engine and rating approaches are discussed.

f) Reputation computation engines: One of the most critical features of a reputation system is the reputation computation aggregation algorithm. Such an algorithm integrates ratings into one score, and at the same time needs to ensure that bad raters are identified and removed to obtain accurate ratings. There are many complex aggregating algorithms that have been proposed such as fuzzy models and Bayesian systems.

Currently, most online reputation systems as eBay and Amazon choose to use simple algorithms [8], such as summation, average or percentage. Simple summation adds all of the ratings, regardless if it is positive, neutral or negative and the calculation is easily understood and adopted by users [8, 21]. Unfortunately, this feedback metric is flawed, for example, if a user has 10 positive feedback points out of 10 transactions and another has 20 positive and 10 negative feedback points out of 30 transactions, they would have the same reputation score [5].

Average rating is based on the same principle as simple summation, however average rating is perceived as more accurate. Ratings can also be calculated by means of weighted average ratings. This infers that each user has a credibility score that determines their weight ratio [5]. Many interesting aggregating algorithms have been proposed that can be classified into five categories [26]:

- By averaging ratings, simplicity in algorithm design is ensured and low cost in system execution.
- Weightings are introduced by weighting the ratings of acquaintances but those of strangers are averaged.
- Only ratings from witnesses are used, who have interacted with the entity being rated. In such a weighted majority algorithm only the ratings from witnesses are aggregated, and the weight of witnesses is decreased if it differs from self-own recognition.
- The weight of ratings is based on the similarity of the experience between the rater and the other participant to improve accuracy.
- Ratings can be aggregated and weights of raters can be updated through deriving the expectation of the Beta distribution.

In simulation [26] it was found that most complex algorithms will have better results. However, in several circumstances the simple algorithm can outperform the complicated algorithms. In particular, the first average algorithm is found to be more resistant to different type of bad raters [26].

To configure the reputation aggregation algorithm for a RaaS, one of these aggregation algorithms can be chosen as they may be able to accommodate a variety of communities and would be understood and adopted by users [5].

g) Reputation measurement parameters: There are crucial parameters which may increase the accuracy of the expected reputation score namely transitivity rate and time [21].

Transitivity rate represents the fact that recommendations from third-hand ratings with a transitivity degree of three may have the least influence on the trustworthiness measurement. Therefore, in a recommendation chain, recommendations from known participants who already have had interaction with the requested party should have more weight as first-hand recommendations, than those who are known but have not had any previous interactions with the requested party or those who are unknown.

Time influences the effect of ratings on the computation as the most recent rating will have a higher weight ratio than ratings that are older. Thus ratings decay over time. The advantage is that users benefit from having a rating value that reflects how the most recent services performed. These parameters attempt to ensure that ratings are more accurate as weight ratios are an effective way to counteract "bad" raters.

Finally, the *Entity Selection* component is considered where the resultant reputations is now used.

h) Rating score: The reputation system finally reports its results to users in two different formats namely aggregated reputation scores and individual ratings and opinions [24]. Reputation scores are the result of the scoring and ranking component, whereas the individual ratings are collected through the information gathering component.

When reputation scores are presented, the time line it represents should be provided to assist users with decision-making. Reputation information is disseminated to end users via different access methods such as web sites, emails or RSS (Rich Site Summary) feeds. Certain information may be made publicly available, whereas others may require a subscription fee.

Next, the requirements for a RaaS framework are presented.

3.3 Requirements for a RaaS framework

Table 1 and Table 2 briefly provides the most relevant requirements for a RaaS framework. The requirements are given according to the two main drivers and the first three components of reputation systems. It has been indicated on the table where a high cost is associated with a set of requirements. In the last case, the list of requirements identifies a set of configurable features that should be present.

SOFTWARE-AS-AS-SERVICE REQUIREMENTS			
SAAS application architecture	• SOA layered architecture • Well-designed interfaces • Standards-based messaging • Support for integration and data semantics		
SAAS design attributes	• Multi-tenancy • Single version • Logical data separation		
	Information gathering	**Scoring and ranking**	**Entity selection**
SAAS application requirements using configuration	• Add, delete and changes roles • Custom fields and types, and deleting data not needed • User interface look and feel, data presented.	• Tasks switched, added and reordered and task roles to be changed • Business rules modified by changing or setting rules and the rule triggers. • Reputation calculation - adding or changing actions or triggering actions at different points. **(HIGH COST)**	• Reports changed with respect to style, dataset used and query rules.

Table 1: SaaS Requirements for RaaS

SaaS requirements: From the list of requirements in Table 1, the design of the RaaS component is driven by considering integration features, SaaS design considerations and configuration features. SaaS application architecture and design attributes are requirements that should all be applied when designing and implementing the RaaS and are not configurable in nature.

Configurable features: SaaS application feature requirements shown in the last section in Table 1 provide an indication of the possible configurable options that should be present to administrators of tenants. Important configuration options are those allowing the configuration of input data such as roles, data types such as rating and reputation scores, rules that apply the strictness by which types of raters are allowed to rate or incentives, actions by which algorithms are applied, and reporting of ratings and reputation scores. This list only provides an indication of the types of configurable features that should be present. These features can be fleshed out in more detail after Table 2 is reviewed.

Trust and reputation system requirements: Table 2 summarises the requirements elicited from users, and those from investigating trust and reputation system components. Between these two sets, some overlap can be noted. Both sets highlight that the manner in which ratings are done - preferably after each and every transaction - is important to apply. Furthermore, that the results produced by the reputation algorithm can be understood with ease, that the reputation of the rater needs to be verified, the decay of trust and the sorting of reputation scores needs to be implemented.

Configurable features: Trust and reputation system

TRUST AND REPUTATION SYSTEM REQUIREMENTS			
	Information gathering	**Scoring and ranking**	**Entity selection**
User trust and reputation requirements	• Each interaction needs to be rated • Simple and intuitive rating scales • Initial ratings should be treated differently • The quality parameters of a service should be controlled and certified by a trusted party • Raters reliability must be controlled • Similarity between recommenders is important. • Trust values decay and become invalid over time.	• A single trust rating calculated by taking into account different service aspects and their weights. • Computation should be an aggregated of all weighted aspects, similar to an "average". **(HIGH COST)**	• Services should be sorted according to their trust rank • Providers should be made comparable to each other.
Reputation system framework requirements	• Direct or indirect collection channels • Restrictions on who can rate • Context-dependant ratings • Collection costs • Simple rating format • Single/multiple criteria • Incentives **(HIGH COST)**	• Complexity of aggregation algorithm • Weighting of recommendations • Decay of trust **(HIGH COST)**	• Reputation score / individual ratings • Time reported

Table 2: Trust and Reputation Requirements for RaaS

requirements provide more detail as to the configurable options that should be present to administrators of tenants. Configurable features are now presented according to the three reputation system components.

For the *Information Gathering* component, important configurable features are the type of collection channel, the type of rating information such as from direct experience, the context of the rating such as cost and/or quality of a product, the reputation of the rater and its maintenance, the format of the score and the incentives provided to raters. A high cost factor is the use of collection channels used to source ratings and feedback.

The most complex set of configurable features are those of the *Scoring and Ranking* component, where reputation computations are performed. There are many possible configurable features such as the order of tasks executed, choice of algorithms, and rules and weightings of criteria. The *Scoring and Ranking* component is not trivial to apply and is very complex in nature.

The result of the reputation computation is used in the *Entity Selection* component, where aspects such as the reputation scores and individual ratings can be provided to end users.

Next a RaaS framework is proposed in light of the identified requirements.

4. RAAS FRAMEWORK

A RaaS framework is defined using the requirements defined in this paper. First the architecture is given by considering only RaaS functional components and not business components such as billing and metering. RaaS framework configuration is described followed by the RaaS-to-enterprise integration process.

4.1 RaaS architecture

The RaaS component is integrated with the applications of tenants. As organizations may make use of more than one service eg PayPal, the RaaS is can be more accurately described as a SaaS Mashup service.

Figure 4 gives the architecture of the RaaS component integration, based on how SaaS integration is implemented. At the bottom of Figure 4, the RaaS component is defined over a basic cloud infrastructure found in data centres where hardware and software are used to define virtual machines that are provided to tenants to run their applications on.

The RaaS component exposes REST APIs to tenants to support various features such as input collection channels, users, roles, incentives, weightings, rating measures, calculations, rater reputations, rating criteria, entity selection and reputation reports. For each tenant, the context of interaction needs to be maintained, uniquely supported by their set of configurable features. A configuration tool allows administrators of tenants to configure features.

RaaS API calls and results are composed with those of the existing applications of the tenant using integration tools. End users of a tenant such as Organization ABC do not directly communicate with the RaaS, but interacts with the RaaS component via web interfaces. Reputation data is sent into the system programmatically, using a REST API.

From the list of configurable features given previously, it is

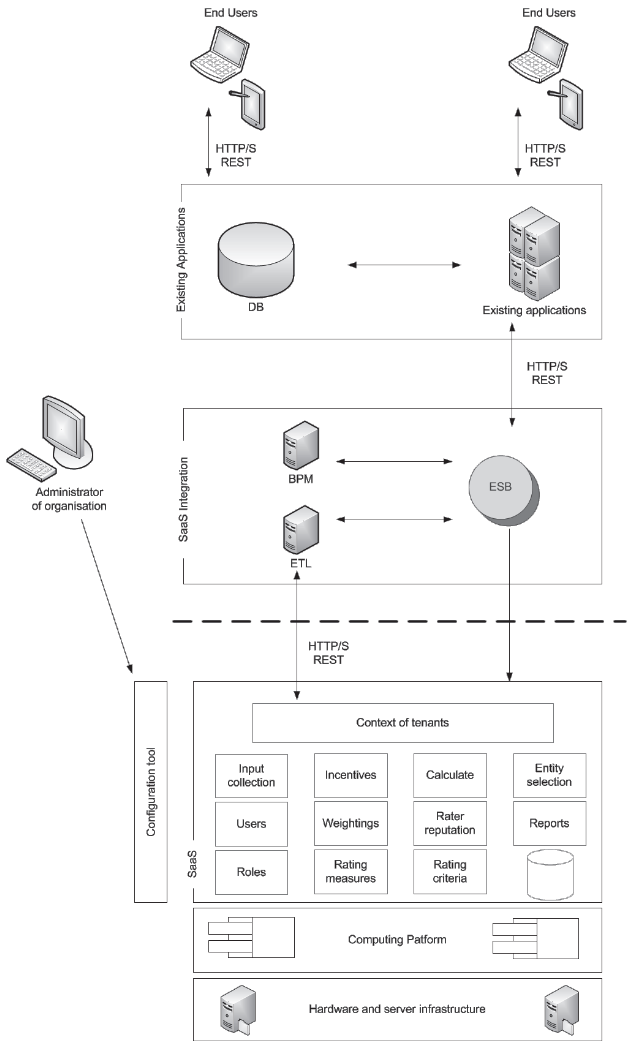

Figure 4: RaaS Architecture

clear that RaaS configuration is not straightforward. This aspect is described next in more detail.

4.2 *RaaS configuration*

The RaaS architecture supports a configuration tool to allow administrators of tenants to uniquely define features to suit their community needs. The RaaS configuration tool should be designed to be easy and intuitive for the administrators, but at the same time be able to satisfy the needs of tenant requirements. Without this feature, it would be impossible to use the single instance of the software for different tenant applications.

The RaaS component supports many functions such as input collection channels, incentives, users and roles, ratings and reputation calculations. The configuration of these functions is no trivial matter, and much intelligence is required to ensure that options are set that would ensure a reputation score that is a true reflection of the behaviour of quality of the entity being rated. Careful consideration should be given to aspects such as the type of data that is exposed, the type of rating format is required by specific algorithms, whether weights can be set or not, which groups of raters may be granted the ability to rate, which objects can be rated and the number of criteria to be used.

Considering the above mentioned complexities in configuring reputation computation this research now proposes a two level reputation configuration approach, one for novice users, and one for knowledgeable users who understand the implications of their choices.

For novice users, there may be a few options available, based on the risk the organization may experience, to select from such as:

- Low - reputation computation with basic summation of values.
- Medium - reputation computation that encourages strangers by initially bootstrapping their trust to a level to ensure participation.
- Strict - reputation computation that is strict with "bad" behavior as the risk is high.

An advanced configuration panel may be made available to knowledgeable users to select a variety of options.

In both cases, the RaaS should make available a simulation feature that will illustrate to the administrator what the effect of the choice will be, in order to avoid any misunderstandings.

Next, a RaaS application integration example is described.

4.3 *RaaS application integration example*

The utilization of the RaaS is now discussed according to a set of sample requirements. This discussion now returns to the case of Organization ABC. This organization is an online store for a start-up company that sells products to consumers over the mobile web. It requires a reputation system that is simple and easy to understand, easy to integrate into their applications, favour good behaviour, encourage users to rate, decay older ratings and allow any product to be rated that is sold to customers.

The administrator of Organization ABC configures the RaaS via a configuration menu page after he/she is authenticated by the system. The configuration menu provides the administrator with the capability to view past settings of the RaaS, configure new settings and view reports on the activity of the users within the RaaS.

When an administrator configures a new RaaS, he/she choose the "Low" configuration option based on the organizational requirements that sets initial values for a number features. A few features are further configured as follows:

- Input data such as those relating to the types of entities to be rated, user profiles and rating values (out of 5) are set. Initial values are provided by the "Low" configuration features that can be adjusted.

- Input collection channel is set to prompt the user directly after a transaction to save on collection costs.
- Incentives are set to be of altruistic nature to encourage users to rate. Rewards are given every time a rater rates a transaction. The rater is given a public reputation score to encourage others to rate.
- Reputation calculation is set to a simple average rating calculation so that users can easily understand and interpret it.
- Rating criteria is set to single, indicating that a product purchased is rated by the buyer.
- Trust decay is set so that previous ratings decay by 50% when older than 6 months and by 100% when older than 18 months.

After configuration features are set, the RaaS is programmatically integrated with the orchestrated calls and responses of the existing applications of Organization ABC.

Figure 5 illustrates a typical web interface displaying rating data. Buyer X views the product search page of Organization ABC on the web browser or app via the mobile device. He has searches for a product and sees its ratings displayed as star ratings out of 5. The page is programmatically created, by sending a REST request such as "https://www.example.com/sendReputation/11", where 11 is the product code that is sent to the RaaS, and displaying the result.

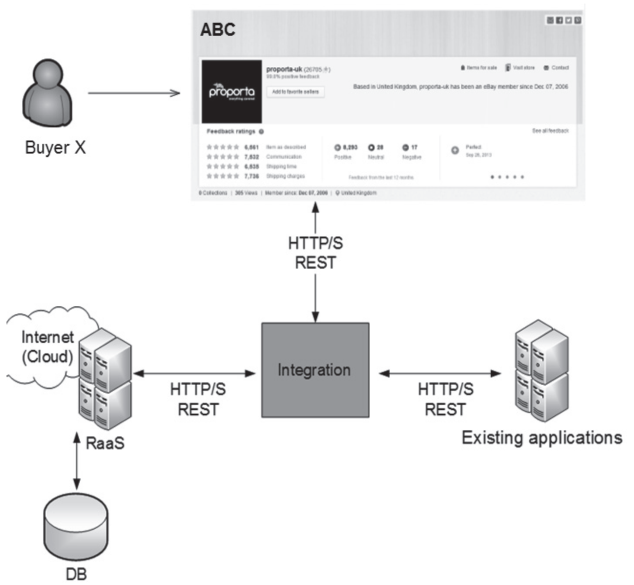

Figure 5: RaaS Communication

In the background, for all transactions processed, ratings are sent to the RaaS to enable the calculation of a reputation score of meaning. Reputation scores are stored with their product profiles. Rater reputation is maintained in the same manner to support the altruistic incentives configuration.

The RaaS thus gives organization a competitive edge as they do not experience a delay in time to market for their new online store, and at the same time they have a sophisticated reputation system to support their online store.

4.4 RaaS framework challenges

There are many challenges that stem from the creation of a RaaS framework:

- Organizations may be challenged by the fact that their data is housed externally. Even though SaaS providers have better security than many of their clients, the transfer of control over corporate data is a difficult decision.
- Configuring reputation computation and behavior is complex. As workflows allow automation of processes involving human and machine-based activities it may be important to apply it in this context.
- Each tenant has specific needs with respect to their data requirements. To address this, a template for storing data can be provided that meets most requirements, with options to add fields to tables.
- As tenants of the RaaS component have a large variety of users, and the responsibility for creating individual accounts for end users, and granting access to resources lies with the tenant, a well-developed access control component should be provided.
- The management of raters and other identities is complex. In most cases, user accounts are managed and stored independently by each tenant and authentication occurs within the organizational boundary. This means that the identity of the user, with any relevant credentials is sent to the RaaS to allow identification and access control. Different types of identities and credentials need to be managed.
- A key factor for SaaS is availability. For mission-critical applications, network availability is a dangerous point of failure.

5. CONCLUSION

The main aim of this paper was to identify requirements for a RaaS framework. Here, this has been done only at a theoretical level. Although a comprehensive set of requirements have been identified, more research and analysis is still needed. To the best of our knowledge, this is the first attempt at identifying requirements for such a framework.

The example scenario indicates that it is plausible to create a configurable reputation system to accommodate a variety of online communities. This research focussed on elements that exist in reputation systems and how these elements can be configured. Further, RaaS framework requirements were identified by considering SaaS application requirements, user requirements and reputation framework requirements. RaaS exposes services by utilizing the proposed architecture.

Future work will address the definition of more detailed configuration features and how they affect reputation scores in different scenarios. A RaaS prototype is to be defined for which simulations are to be performed to experiment with the different configurable components to identify which elements are more appropriate to configure.

6. ACKNOWLEDGEMENT

The support of SAP P&I BIT Mobile Empowerment and the National Research Foundation (NRF) under Grant number 81412 and 81201 towards this research is hereby acknowledged. Opinions expressed and conclusions arrived at are those of the authors and not necessarily to be attributed to the companies mentioned in this acknowledgement.

REFERENCES

[1] Biz Community, "Mastercard worldwide reveals online shopping survey," 2012. [Online]. Available: http://www.bizcommunity.com/Article/196/168/73927.html

[2] S. Mulpuru, C. Johnson, and D. Roberge, "US online retail forecast, 2012 to 2017," 2013. [Online]. Available: http://www.forrester.com/US+Online+Retail+Forecast+2012+To+2017/fulltext/-/E-RES93281?objectid=RES93281

[3] Y. Wang and J. Vassileva, "Toward trust and reputation based web service selection: A survey," *International Transactions on Systems Science and Applications (ITSSA) Journal*, vol. 3, no. 2, pp. 1–38, 2007.

[4] F. Braithwaite and M. Woodman, "Success dimensions in selecting cloud software services," in *37th EUROMICRO Conference on Software Engineering and Advanced Applications*, 2011.

[5] H. Li, "A configurable online reputation aggregation system," Ph.D. dissertation, University of Ottawa, 2007.

[6] D. Gambetta, *Trust Making*. Oxford, 1990, ch. Can We Trust Trust?

[7] T. D. Huynh, "A personalized framework for trust assessment," in *SAC' 09*, 2009.

[8] A. Josang, R. Ismail, and C. Boyd, "A survey of trust and reputation systems for online service provision," *Decision Support Systems*, vol. 43, no. 2, pp. 618–644, 2007.

[9] D. Fahrenholtz and W. Lamersdorf, "Transactional security for a distributed reputation management system," in *Proceedings of the Third International Conference on E-Commerce and Web Technologies (EC-Web)*, 2002.

[10] M. Vinkovits, "Towards requirements for trust management," in *Tenth Annual International Conference on Privacy, Security and Trust*, 2012.

[11] F. G. Marmol and G. M. Perez, "Towards pre-standardization of trust and reputation models for distributed and heterogeneous systems," *Computer Standards & Interfaces*, vol. 32, no. 4, pp. 185–196, 2010.

[12] R. Alnemr, M. Schnjakin, and C. Meinel, "Towards context-aware service-oriented semantic reputation framework," in *International Joint Conference of IEEE TrustCom-11/IEEE ICESS-11/FCST-11*, 2011.

[13] H. Liao, "Design of SaaS-based software architecture," in *International Conference on New Trends in Information and Service Science*, 2009.

[14] X. Jiang, Y. Zhang, and S. Liu, "A well-designed SaaS application platform based on model-driven approach," in *Ninth International Conference on Grid and Cloud Computing*, 2010.

[15] R. T. Fielding, "Architectural styles and the design of network-based software architectures," Ph.D. dissertation, University of California, 2000.

[16] json.org, "Introducing JSON." [Online]. Available: http://www.json.org/

[17] World Wide Web Consortium, "Extensible markup language (XML)." [Online]. Available: http://www.w3.org/XML/

[18] J. Espadas, D. Concha, and A. Molina, "Application development over Software-as-a-Service platforms," in *The Third International Conference on Software Engineering Advances*, 2008.

[19] W. Sun, X. Zhang, C. J. Guo, P. Sun, and H. Su, "Software as a service: Configuration and customization perspectives," in *IEEE Congress on Services Part II*, 2008.

[20] F. Marmol and M. Perez, "TRMSim-WSN, trust and reputation models simulator for wireless sensor networks," *IEEE*, 2009.

[21] Z. Noorian and M. Ulieru, "The state of the art in trust and reputation systems: A framework for comparison," *Journal of Theoretical and Applied Electronic Commerce Research*, vol. 5, no. 2, pp. 97–117, 2010.

[22] G. Swamynathan, "Reputation management in decentralized networks." [Online]. Available: http://www.powershow.com/view/11e08-ndu1z/reputation_management_in_decentralized_networks_powerpoint_ppt_presentation

[23] F. R. Farmer and B. Glass, *Web Reputation Systems*. O'Reilly, 2010.

[24] L. Liu and M. Munro, "Systematic analysis of centralized online reputation system," *Decision support systems*, vol. 52, no. 2, pp. 438–449, 2012.

[25] R. Jurca, "Truthful reputation mechanisms for online systems," Ph.D. dissertation, 2007.

[26] Z. Liang and W. Shi, *Collaborative Computing: Networking, Applications and Worksharing*, 2005, ch. Performance Evaluation of Rating Aggregation Algorithms in Reputation Systems.

NOTES

SAIEE AFRICA RESEARCH JOURNAL – NOTES FOR AUTHORS

This journal publishes research, survey and expository contributions in the field of electrical, electronics, computer, information and communications engineering. Articles may be of a theoretical or applied nature, must be novel and must not have been published elsewhere.

Nature of Articles

Two types of articles may be submitted:

- Papers: Presentation of significant research and development and/or novel applications in electrical, electronic, computer, information or communications engineering.
- Research and Development Notes: Brief technical contributions, technical comments on published papers or on electrical engineering topics.

All contributions are reviewed with the aid of appropriate reviewers. A slightly simplified review procedure is used in the case of Research and Development Notes, to minimize publication delays. No maximum length for a paper is prescribed. However, authors should keep in mind that a significant factor in the review of the manuscript will be its length relative to its content and clarity of writing. Membership of the SAIEE is not required.

Process for initial submission of manuscript

Preferred submission is by e-mail in electronic MS Word and PDF formats. PDF format files should be 'press optimised' and include all embedded fonts, diagrams etc. All diagrams to be in black and white (not colour). For printed submissions contact the Managing Editor. Submissions should be made to:

The Managing Editor, SAIEE Africa Research Journal,
PO Box 751253, Gardenview 2047, South Africa.
E-mail: researchjournal@saiee.org.za

These submissions will be used in the review process. Receipt will be acknowledged by the Editor-in-Chief and subsequently by the assigned Specialist Editor, who will further handle the paper and all correspondence pertaining to it. Once accepted for publication, you will be notified of acceptance and of any alterations necessary. You will then be requested to prepare and submit the final script. The initial paper should be structured as follows:

- TITLE in capitals, not underlined.
- Author name(s): First name(s) or initials, surname (without academic title or preposition 'by')
- Abstract, in single spacing, not exceeding 20 lines.
- List of references (references to published literature should be cited in the text using Arabic numerals in square brackets and arranged in numerical order in the List of References).
- Author(s) affiliation and postal address(es), and email address(es).
- Footnotes, if unavoidable, should be typed in single spacing.
- Authors must refer to the website: http: //www.saiee.org.za/arj where detailed guidelines, including templates, are provided.

Format of the final manuscript

The final manuscript will be produced in a 'direct to plate' process. The assigned Specialist Editor will provide you with instructions for preparation of the final manuscript and required format, to be submitted directly to:
The Managing Editor, SAIEE Africa Research Journal, PO Box 751253, Gardenview 2047, South Africa.
E-mail: researchjournal@saiee.org.za

Page charges

A page charge of R200 per page will be charged to offset some of the expenses incurred in publishing the work. Detailed instructions will be sent to you once your manuscript has been accepted for publication.

Additional copies

An additional copy of the issue in which articles appear, will be provided free of charge to authors.
If the page charge is honoured the authors will also receive 10 free reprints without covers.

Copyright

South African Institute for Electrical Engineers (SAIEE)
PO Box 751253, Gardenview, 2047, South Africa
Tel: 27 11 487 3003 | Fax: 27 11 487 3002
E-mail: researchjournal@saiee.org.za | Website: www.saiee.org.za