Formalising Human Recognition: a Fundamental Building Block for Security Proofs

Kenneth Radke[1] **Colin Boyd**[2] **Juan Gonzalez Nieto**[3] **Mark Manulis**[4]
Douglas Stebila[1]

[1] School of Electrical Engineering and Computer Science
Queensland University of Technology, Australia
Email: `k.radke, stebila@qut.edu.au`

[2] Department of Telematics
Norwegian University of Science and Technology
Email: `colin.boyd@item.ntnu.no`

[3] BAE Systems Detica
Email: `juan.gonzalez@baesystemsdetica.com`

[4] Department of Computing
University of Surrey, United Kingdom
Email: `m.manulis@surrey.ac.uk`

Abstract

A fundamental part of many authentication protocols which authenticate a party to a human involves the human recognizing or otherwise processing a message received from the party. Examples include typical implementations of Verified by Visa in which a message, previously stored by the human at a bank, is sent by the bank to the human to authenticate the bank to the human; or the expectation that humans will recognize or verify an extended validation certificate in a HTTPS context. This paper presents general definitions and building blocks for the modelling and analysis of human recognition in authentication protocols, allowing the creation of proofs for protocols which include humans. We cover both generalized trawling and human-specific targeted attacks. As examples of the range of uses of our construction, we use the model presented in this paper to prove the security of a mutual authentication login protocol and a human-assisted device pairing protocol.

Keywords: Ceremony; human; HTTPS; TLS; provable security; authentication; HPA; protocol.

1 Introduction

Practice-oriented provable security, for authentication protocols, was introduced by Bellare and Rogaway in 1993 [1]. Since this time, many authentication protocols have been proven secure in theory, only to fail to meet this level of security when used in practice by a human. Increasingly there has been a realisation that to create secure protocols which involve humans, human capabilities need to be an explicit consideration of the security model, as exemplified by Shostack and Stewart's statements, "...our approach to information security is flawed" and "the way forward cannot be found solely in mathematics or technology" [11].

This work was supported in part by Australian Research Council Linkage Grant LP120200246.

We focus on authentication protocols involving a human, in which the human is expected to authenticate the party they are communicating with. This may be either mutual authentication or one-way authentication. For such protocols, the human is expected to *recognise* an authenticator, or perform some task which accepts some security information as input and outputs either accept or reject. In this way we build on the *recognise* function introduced by Gajek et al. [5,6] in their work on authentication to a human in a protocol using TLS. We focus specifically on the recognise functionality a human must perform, and create a formalisation which may be reused in all such protocols.

A formal construction of the human's recognise capabilities can be used in a variety of types of protocols. As already mentioned, there are uses such as the Gajek et al. protocol, which is similar to protocols that have been implemented by numerous financial institutions. Similarly, the construction allows for formal analysis of implementations of the widely used Verified by Visa protocol. In this protocol a message that the human entered on first use of the system is sent back to the human to authenticate Visa to the human in all future executions of the protocol.

Less obvious real world protocols, which our construction may allow for formal analysis of, are authentication protocols over a telephone between two humans. A common example is where an investor calls her stockbroker using the telephone, says a password to the stockbroker, and the stockbroker ensures that the password said by the investor matches the password stored for that investor. Even login messages meant to allow users to compare the last time the system recorded their login credentials were used, against the last time the human remembers logging in, could be analysed for their security properties.

The formalisation captures the distinction between a *targeted* attack, where the adversary knows the identity of the victim and may conduct research and specific social engineering attacks against that person, and a *trawling* or *general* attack [2], where the adversary has no direct knowledge of who the victim is, and therefore must rely on population-wide trends. Schechter et al. highlighted the significance of targeted attacks, when they researched the security of using personal questions as an authentication mechanism to reset passwords [10]. At one level, geographic homogeneity was a factor in allowing the successful guessing of participant answers 13% of the time within five attempts. As the attack became more targeted, success rates increased, with non-trusted and semi-trusted acquaintances being able to guess correctly 17% of the time, and trusted acquaintances being able to guess correctly 28% of the time [10]. This means that the severity and likelihood of success of an attack varies greatly, depending on whether the attack is a targeted attack or a trawling attack.

We will show a range of uses for our formalization, with two examples of adaptations of security proofs. We will show how our approach may be applied in the case of a web-based mutual authentication protocol (see Section 3), and in the case of human-assisted pairing of two bluetooth devices (see Section 4.1). The former will cover the case of human-selected authenticators, while the latter will provide an example of device-selected authenticators. In both cases the central human recognise step remains critical and constant.

2 Security Model for Human-Based Recognition

The security model describes the human's role in recognising information sent to the human, which is a typical process in a protocol where the human authenticates a second party. The model formally describes how an attacker interacts with the human, and what capabilities and constraints the attacker has.

2.1 Formalisation

We begin by describing the situation where a human generates a HPA which they memorize. A HPA is a *human perceptible authenticator*. Subsequently, some information, HPA', is generated by another party and is sent to the human. The human assesses the HPA' by comparing the received HPA' to the HPA they have stored in memory. A concrete illustrative example may be the case where the human selected HPA and the HPA' generated by the other party are images and the human checks to see if the two images HPA and HPA' are equal.

A central ideal we have incorporated is that this behaviour will be different from human to human. That is, *HPA*s that a human generates will be human specific. We shall call *HPA*s that are specific to a human HPA_H. While in one context an example use of the HPA is that the human shares the HPA with an entity, so that the entity can use the HPA to authenticate to the human, this may not be the case all of the time as *HPA*'s could be generated by the entity and given to a human in a setup stage. Therefore we focus exclusively on the *recognise* step common to most device-to-human authentication protocols. That is, a HPA' is received by the human, the human compares the HPA' with their HPA, and either accepts or rejects that HPA' is the same as HPA.

2.1.1 HPA Scheme

We defined *HPASpace* to be the space of *HPA*s. To use the traditional example of alphanumeric passwords, for a specific protocol this space may be bounded by the 94 character possibilities, consisting of 26 lowercase, 26 upper case, 10 numerals and 32 special printable characters, and the number of characters accepted for the password. For example, for eight characters, this is 94^8 possibilities.

The value of HPA_H is the output of a probabilistic algorithm *GenHPA* which is specific to each human, and hence accepts as inputs the human H and the *HPASpace* of the protocol being analysed,

$$\mathit{HPA}_H \leftarrow \mathit{GenHPA}(H, \mathit{HPASpace}).$$

To return to the example of an alphanumeric password, it is widely known that while the possibility of selecting a randomly selected eight character password may be 94^{-8}, or approximately 53 bits of security, non-random human selection typically brings this figure closer to 30 bits of security for humans in general [3]. However, for a specific human, the set of alphanumeric passwords generated may be far smaller [3], and hence the output HPA_H will be part of the human's specific *HPASpace* i.e.

$$\mathit{HPASpace}_H = \{\mathit{HPA}_H | \mathit{HPA}_H \leftarrow \mathit{GenHPA}(H, \mathit{HPASpace})\}.$$

Notice that $\mathit{HPA}_H \in \mathit{HPASpace}_H \subseteq \mathit{HPASpace}$.

The function *Recognise* is defined to model the human's ability to take two inputs, HPA and HPA', one potentially in memory and the other being presented to the human, and compare the two inputs. If, in the human's opinion, there is a match between HPA and HPA' then *Recognise* outputs a one, otherwise *Recognise* outputs a zero. This models the human's assessment of "The two values are the same." Therefore, the *Recognise* algorithm depends on the human:

$$0/1 \leftarrow \mathit{Recognise}(H, \mathit{HPA}, \mathit{HPA}')$$

Of course, no human performs the recognise function perfectly 100% of the time. There will be both false positives and false negatives from the *Recognise* function. False positives occur when the human assesses there is a match between HPA and HPA' and yet the two values are different. False positives are seen as being a result of human's inability to distinguish between two objects, if they are similar enough though not identical, and are discussed in detail below. False negatives result when $\mathit{HPA} = \mathit{HPA}'$ and yet the human assesses there is no match, and the recognise function outputs a 0. False negatives, for example the human being presented with two identical pictures and yet assessing them as different, are seen as an error condition, modelled by an error probability ϵ. False negatives are in agreement with Hopper and Blum's (α, β, t) method of describing human protocols, where value β is the probability of the human not successfully executing the protocol [7].

2.1.2 Human Indistinguishability

We denote the set of different HPA's which the user recognises as being indistinguishable from their chosen HPA as being the set $W_{H,\mathit{HPA}}$. This is the set of the actual HPA and false positives. Note that a HPA' that is recognised by a human as being indistinguishable from their HPA may come from their human specific $\mathit{HPASpace}_H$ or from the general *HPASpace*. For the sake of analysis of a specific protocol, we explicitly exclude any other object in existence which is not from *HPASpace*. That is, if the *HPASpace* of a system is defined as a four digit personal identification number, then a five digit personal identification number would not be a valid HPA. This commonly accepted constraint is a limitation of our approach, since there may be objects from outside *HPASpace* which the human will accept as being their HPA.

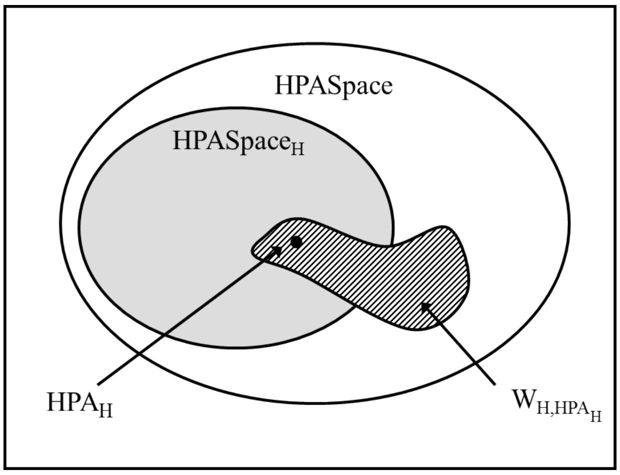

Figure 1: How $W_{H,HPA}$ relates to *HPASpace* and *HPASpace*$_H$.

To aid in the understanding of how the set of false positives $W_{H,HPA}$ interacts with *HPASpace* and *HPASpace*$_H$, the relationship is depicted in Figure 1. These false positives form a set of HPAs, specific to a human, which are similar enough (or human indistinguishable) that the *Recognise* function will output a 1 for a given *HPA*:

$$W_{H,HPA} = W'_{H,HPA} \cup W''_{H,HPA}$$

where $W'_{H,HPA}$ is the set of *HPA*s from *HPASpace*$_H$ and $W''_{H,HPA}$ is the set of *HPA*s from *HPASpace* which are not in *HPASpace*$_H$. Formally:

$$\begin{aligned} W'_{H,HPA} = \{&HPA' \in HPASpace_H | \\ &Recognise(H, HPA_H, HPA') = 1 \\ &\text{with } HPA_H \leftarrow GenHPA(H, HPASpace)\} \end{aligned}$$

and

$$\begin{aligned} W''_{H,HPA} = \{&HPA' \in \{HPASpace \setminus HPASpace_H\} | \\ &Recognise(H, HPA_H, HPA') = 1 \\ &\text{with } HPA_H \leftarrow GenHPA(H, HPASpace)\}. \end{aligned}$$

The relationship between $W'_{H,HPA}$ and $W''_{H,HPA}$ and *HPASpace* is shown in Figure 2.

2.1.3 Security and Usability

An interesting distinction between false positives and false negatives is made when considering security and usability. False positives result in a less secure system. That is, the adversary can now produce not just the exact *HPA*, but any of potentially many *HPA′* (i.e. $|W_{H,HPA}|$) which the human will accept as indistinguishable from *HPA*.

In contrast, false negatives do *not* impact security. If a human is presented with *HPA′* which equals *HPA* but does not assess that this is a match, then the protocol will be aborted and hence the system will remain secure. However, false negatives *do* impact usability in that if the protocol does not successfully proceed when $HPA' = HPA$, then the human will not be able to use the system, or at the very least the human will need to execute the protocol one time more than they needed to. This is similar to the well known trade-off in biometrics.

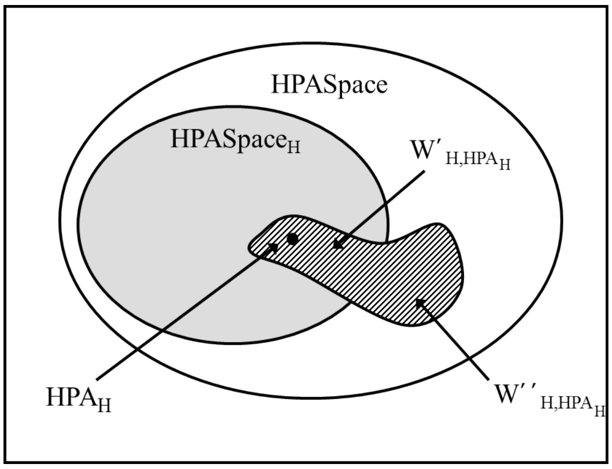

Figure 2: How $W'_{H,HPA}$ and $W''_{H,HPA}$ relate to *HPASpace* and *HPASpace*$_H$.

2.1.4 Probabilistic versus Deterministic

In our formalism, $W_{H,HPA}$ represents the set of *HPA′*s which the user recognises (mistakenly, except in the case where $HPA' = HPA$) as being indistinguishable from their chosen *HPA*. We have captured that this will vary from human to human. We also, in Definition 2, capture that sometimes the real *HPA* will not be recognised as being the real *HPA* by the human. Finally, we capture that the set of *HPA*s that a specific human may pick will be different from human to human, in our definition of *GenHPA*.

However, while this variability is captured and gives useful results, the set $W_{H,HPA}$ is constant and deterministic for a human in our model, whereas in reality such a set may vary over time particularly with context. Further, the set *HPASpace*$_H$ is constant in our model, and again this set may vary with context.

2.2 Security Definition for Human Recognition

In our model, we define security in terms of the adversary $\mathcal{A}$'s ability to obtain a *HPA* value which will cause the human to output a 1 from the *Recognise* function.

The security game proceeds as follows. A *HPA* is generated for a specific human *H* using the *GenHPA* algorithm. This models a human selecting their *HPA*.

$$HPA_H \leftarrow GenHPA(H, HPASpace) \tag{1}$$

The adversary $\mathcal{A}$ knows *HPASpace* and gets oracle access to $Recognise(H,\ HPA_H,\ \cdot)$ and $GenHPA(\cdot, HPASpace)$.

$$HPA' \leftarrow \mathcal{A}^{Recognise(H,HPA_H,\cdot),GenHPA(\cdot,HPASpace)}$$

Access to the *GenHPA* oracle models $\mathcal{A}$'s ability to gain information about expected *HPA*s from humans, including the target human. That is, we have not limited $\mathcal{A}$ to using the *GenHPA* oracle on only the target human. This allows $\mathcal{A}$ to use the *GenHPA* oracle to effectively construct *HPASpace*$_H$, including frequency distribution, for both the target *H* and other humans. We call this constructed *HPASpace*$_H$, "$HPASpace'_H$". This is now a targeted attack, which is more damaging than a general attack. Access to the *Recognise* oracle allows $\mathcal{A}$ to test $\mathcal{A}$'s selected *HPA′* to see if the selected *HPA′* is accepted.

$$Recognise(H, HPA_H, HPA') = 1.$$

2.2.1 Upper Bound on $\mathcal{A}$'s Probability of Success

The adversary can either work from *HPASpace* or generate a $HPASpace'_H$ for the human using the *GenHPA* oracle. As shown in (1), $HPASpace_H$ contains at most q_{gen} elements, where q_{gen} corresponds to the number of $GenHPA(H, HPASpace)$ queries.

If the adversary selects from *HPASpace* then the attack is described as a general attack, whereas if the adversary selects from $HPASpace_H$ then the attack is described as a targeted attack.

Targeted Attack $\mathcal{A}$ selects an authenticator *HPA* from $HPASpace_H$, creating a targeted attack. In this case,

$$Succ_{\mathcal{A}\ \text{Human}} = \Pr[\mathcal{A}\text{ wins}] \leq \max_{HPA^* \leftarrow GenHPA(H,HPASpace)} \frac{q'|W'_{H,HPA^*}|}{|HPASpace_H|} \cdot \Pr[HPA^*] \quad (2)$$

where q' is the number of queries to the *Recognise* oracle to test a *HPA* generated using queries to the *GenHPA* oracle, and $\Pr[HPA^*]$ is the probability of HPA^* being generated by *GenHPA*. This is the case of the targeted attack against a human, by $\mathcal{A}$ somehow having knowledge of *HPA* choices for that human (perhaps by knowing *HPA*s used by that human on other systems).

Prior work by Gajek et al. [5] considered only non-human-specific indistinguishability, and from the entire *HPASpace* ($|W_{HPA}|$). In addition to adapting this to the human-specific targeted attack setting, the upper bound on $\mathcal{A}$'s probability of success must take into account the likelihood that a *HPA* is generated by $GenHPA(H, HPASpace)$.

Intuitively, a HPA^* with a large $|W_{H,HPA^*}|$ is unlikely to be the upper bound on $\mathcal{A}$'s success probability if the likelihood that HPA^* is picked as the authenticator is minute. This introduces the concept of the *frequency of use* of HPA^*. Therefore, for the targeted attack, informally the maximum of the frequency of use $\Pr[HPA^*]$ combined with the size of set W_{H,HPA^*} is the upper bound on $\mathcal{A}$'s success.

Trawling Attack $\mathcal{A}$ picks an authenticator from the general *HPASpace*. In this case,

$$Succ_{\mathcal{A}\ \text{Human}} = \Pr[\mathcal{A}_{wins}] \leq \max_{HPA^* \leftarrow HPASpace} \frac{q''|W_{H,HPA^*}|}{|HPASpace|} \quad (3)$$

where q'' is the number of queries to the *Recognise* oracle with corresponding no prior query to the *GenHPA* oracle. This is the case of the general trawling attack, with no prior knowledge regarding the human's choices, such that the adversary has to select from the entire *HPASpace*.

Usually we can expect that the targeted attack is more likely to be successful than the trawling attack, but there could be exceptional cases in which this is not true. Therefore, in general the probability from Equation (2) may be a more exact upper bound on the adversary's success probability. However without access to *GenHPA* the upper bound remains as in Equation (3). We can now define what it means for our schemes to be secure and correct.

Definition 1 (δ-security). *We say a HPA scheme is* δ-secure, *meaning that the scheme can be used as an authentication scheme for an entity to a human, if*

$$Succ_{\mathcal{A}} \leq \delta.$$

Definition 2 (ϵ-correctness). *We say a HPA scheme is ϵ-correct if, for all HPA_H in HPASpace, where $HPA_H \leftarrow GenHPA(H, HPASpace)$,*

$$\Pr[Recognise(H, HPA_H, HPA_H) = 1] \geq 1 - \epsilon$$

where ϵ represents the false negative rate of correctness. For correctness, we are not concerned about false positives, which is covered by $W_{H,HPA}$.

2.3 Analysis and Discussion

The *HPASpace* will be system specific. Using a classic human authenticating scenario which may be adapted to create mutual authentication, a system for banking Personal Identification Numbers (PINs) may have a *HPASpace* limited to four numerical digits, while other systems may have graphical or alphanumeric *HPASpace*s. The effect of *GenHPA* taking as an input *HPASpace* is that comparison between security results for different systems can be made.

Giving the adversary $\mathcal{A}$ oracle access to the human specific *GenHPA*, allows the modelling of the effect of *HPA* reuse and of the preferences of the user. If the size of $HPASpace_H$, the output of *GenHPA*, is less than the size of *HPASpace*, then the adversary receives an advantage. There may be instances where $\mathcal{A}$ does not get this capability, depending on whether the attack is a targeted or trawling attack. A targeted attack is by far the stronger and more damaging attack, as in the real world this would model the case where an adversary has knowledge of a human's HPA choices. This knowledge may exist because the adversary may be a legitimate server where the human logs in elsewhere, and thus the adversary has seen many prior examples of the human's HPA choices.

In general, ensuring that *GenHPA* and *Recognise* are human specific functions, allows for modelling of targeted attacks at a specific human. Giving $\mathcal{A}$ oracle access $Recognise(H, HPA_H, \cdot)$ to the human specific *Recognise* function means that an adversary with infinite resources could create the set $W_{H,HPA}$ for a given *HPA*.

3 Human-Specific *HPAGen*

In this section we will describe the case where a human chooses the HPA. In the next section, we will cover the alternative case where a device selects the HPA for the human.

Our formalisation, as defined allowing for targeted attacks, is ideal for use in existing device-to-human (D2H) authentication scenarios, such as the protocol by Gajek et al. [5], or protocols involving authentication by humans in general.

3.1 Gajek et al. Browser Based Mutual Authentication over TLS

Gajek et al. have created a mutual authentication protocol including a human and a *HPA*. A sketch of the Gajek et al. protocol, including a description of where the *HPA* is used and how the human *recognise* function is applied, follows [5, 6]:

1. The protocol is between a server, a human's computer running a web browser (which has state), and the human.

2. Before the protocol begins, the human has selected a *HPA* and provided that *HPA* to the server. The *HPA*s suggested by Gajek et al. are a personally selected image or voice recording.

3. Both the server and the human's computer have authentication certificates and associated private keys, and a secure TLS connection is established between the browser and the server, when the browser on the human's computer opens the server's webpage. This process authenticates the server to the human's browser and the human's browser to the server.

4. The server sends the human the *HPA* that the human has stored with the server (by completing a lookup of the human's browser-specific certificate, to know whose *HPA* to send), via the web browser which *renders* the *HPA* for the user. This step authenticates the server to the human if the human *recognises* the *HPA*.

5. Having *recognised* the *HPA*, the human sends the server their traditional login and password. This step authenticates the human to the server.

Our formalisation makes proof of such a protocol more complete by replacing game 20 of their proof [6]. The sketch of their proof is that SSL is proven in games 0 to 19, and in game 20 the ability of the adversary to guess a *HPA* that the human will recognise is considered. Further, the analysis presented [6] could be simplified since the initialisation stage can now be explicitly comprehended as running the *GenHPA* algorithm, and the process of recognition realised as an invocation of the *Recognise* function. Including the concepts of Gajek et al.'s specific proof, the result of this game would become:

$$|\Pr[Win_{20}] - \Pr[Win_{19}]| \leq Succ_{\mathcal{A}\ \text{Human}}$$

where $Succ_{\mathcal{A}\ \text{Human}}$ is defined by either Equation (2) or Equation (3). This is in contrast to the corresponding equation in the original proof [6], using our notation:

$$|\Pr[Win_{20}] - \Pr[Win_{19}]| \leq \frac{q|W|}{|HPASpace|}$$

where q is the number of executions of the protocol.

The above demonstrates how our formalism can be used in a certificate or SSH-based key exchange protocol. Our proof goes beyond the Gajek et al. proof in the following areas:

1. In the Gajek et al. proof, $\mathcal{A}$ selects from all of *HPASpace*, roughly equivalent to our trawling attack, whereas our proof allows for a targeted attack where $\mathcal{A}$ selects from the human-specific $HPASpace_H$.

2. Our model covers the concept of *frequency of use* of a *HPA*, not just the size of $|W|$.

Furthermore, the technique used here can be applied to any authentication protocol, password authenticated key exchange (PAKE) protocol or key agreement protocol which requires the human to authenticate a message in the protocol. While these examples are network based, there are examples of use involving just the human and a device, such as a trusted computing scenario where a computer's trusted platform module (TPM) could be used to securely assess the computer and securely present a *HPA* to the user.

An example trusted computer scenario may involve a login procedure where a picture is constructed by the trusted module for the human. That is, the TPM creates a list of hashes from different stages of a computer's boot sequence, and these hashes are graphically presented to the user in some way. Thus, the computer's TPM could be used to securely assess the computer and securely present a *HPA* to the user where the *HPA*'s construction (rendering by the TPM) depends on the status of the computer. As long as nothing has changed on the computer, then the same *HPA* will always be shown to the user to recognise each time she logs in, otherwise a completely new *HPA* will be sent to the user (i.e. in the same way that a one bit change will create a completely new HASH value). So a possible concrete implementation may be a 64 pixel black and white picture, 8 pixels high by 8 pixels wide, so the size of *HPASpace* would be 2^{64}. *GenHPA* would be generated by the trusted platform module to generate *HPA*s uniformly over *HPASpace*. A typical $W_{H,HPA}$ may consist of any picture with a similar number of white (or black) pixels as *HPA*. Since the *HPA*s are device controlled, $\mathcal{A}$'s advantage would be limited by Equation (3). Modelling the login procedure in this way would allow adjustment of security parameters such as the number of pixels and the number of colours in the image constructed by the TPM and presented to the user when they login, to arrive at the desired security for the login procedure.

3.2 Human-based Recognition with Non-human Controlled Authenticators

This leads us to the many practical protocols in use where *HPA*s have been chosen by devices or the system, rather than by the human. From a usability perspective, a human may be able to better remember a *HPA* they have selected. However, the essence of a D2H authentication protocol, where a human recognises some information (*HPA*) sent by the device, is that the *HPA* has been previously agreed on by the human and the device. Whether a human registers a *HPA* at a bank, or whether the bank sends the human a *HPA* in a setup stage, is irrelevant from the perspective of whether the protocol will function. In either case, the *HPA* can be sent to the human by the bank in all future protocol runs to authenticate the bank to the human.

Having the *HPA* chosen by the device means that $HPASpace_H$, the subset of *HPASpace* which the *HPA* will be in, encompasses all of *HPASpace*. This is because the device will choose the *HPA* from the entire *HPASpace* by a probability distribution, presumably a random distribution. From a security perspective, this maximises the space that the adversary has to select from to acquire a HPA' which the user will recognise as their *HPA*. Having the *HPA* chosen by something other than the human, shifts the upper bound available to the adversary from a more powerful *targeted* attack to a weaker *trawling* attack.

We now adapt our formalisation towards non-human specific *HPA* selection, where non-targeted general trawling attacks apply. We focus on more general cases, such as human-assisted pairing protocols.

4 Non-Human-Specific *HPAGen*

Humans are often called on to play a part in protocols for the authentication of devices. For example, an authenticated key agreement (AKA) protocol employing short authenticated strings (SAS) may be used to manually pair wireless devices by having the user check matching values on each device [8].

In such device pairing protocols, two *HPA*s are sent to a human, and if the human recognises the two *HPA*s as matching and accepts, then the human takes an action and the devices become paired. As always, the *HPA*s can be any human perceptible authenticator, such as two series of sounds, two series of flashing lights, or text or images displayed on a screen. We formalise this process by setting the *HPA* variable to the output of a probabilistic algorithm *GenHPA* which selects from the *HPASpace* of the protocol being analysed, i.e.

$$HPA \leftarrow GenHPA(HPASpace)$$

Note the removal of the human from the *GenHPA* step, and hence the space of the generated *HPA* is *HPASpace* not $HPASpace_H$ as it has been in the human generated *HPA* case. The *Recognise* function remains human specific, and the adversary's win condition remains:

$$Recognise(H, HPA, HPA') = 1$$

Since $HPASpace_H$ is now the size of *HPASpace* and hence the upper bound is as for a general trawling attack, the bound on $\mathcal{A}$'s probability of success for a device chosen *HPA* is

$$Succ_{\mathcal{A}\ \text{Device}} = \Pr[\mathcal{A} \text{ wins}] \leq \max_{HPA^* \leftarrow GenHPA(HPASpace)} | \frac{q|W_{H,HPA^*}|}{|HPASpace|} \cdot \Pr[HPA^*] \, |.$$

4.1 Human-based Recognition with Two Devices

An example implementation, which could now be rigorously examined using our formalisation, is pairing protocols for Bluetooth devices. When using human assisted pairing of devices using Bluetooth, one method is for protocols based on short authenticated strings (SAS). In these protocols, the SAS are somehow represented on two devices for a human to compare, using for example audible tones or flashing lights. Now these protocols involving a human can be examined formally using our model.

We will demonstrate how to incorporate our formalism by providing a security proof of Prasad and Saxena's human-assisted Bluetooth pairing protocol [9] which is based on Pasini and Vaudenay's SAS protocol shown in Figure 3 [8]. Pasini and Vaudenay's protocol is designed to allow for the authentication of a potentially large message, by comparing two short strings. Note that by ensuring the messages came from a specific party, i.e. authentication, this also ensures integrity has been maintained. That is, if a third party changes the message, thus breaking integrity, then the message no longer originates from the initial party that sent the message. So the intention of the Pasini and Vaudenay protocol is to have no confidentiality of the messages sent, but have assurances of integrity and know who has sent the messages.

In Pasini and Vaudenay's protocol, there is a critical final step over an *authenticating* channel. Using Vaudenay's definition of this channel from [13], "The authentication channels provide to the recipient of a message the insurance on whom sent it as is. In particular, the adversary cannot modify it (i.e. integrity is implicitly protected). . . . (the) channels are not assumed to provide confidentiality." The SAS protocol in Figure 3 employs a commitment scheme (commit, open) and a keyed hash function. An overview of the protocol is:

1. Alice selects a string K of length κ at random.
2. Alice commits to the value K and their message m_A. This commitment is c. The commitment is such that Alice cannot later manipulate the value of K having seen what value Bob later sends.
3. Alice sends c and m_A to Bob.
4. Bob then picks R at random and sends R and their message m_B to Alice.
5. Alice sends d, used to open the commitment scheme, to Bob.
6. Bob uses m_A, c and d to create (or "open via the commitment scheme") K. The hats on the values (shown in Figure 3), that is, comparing c with $\hat{c}$, indicates that the value has passed over an insecure channel. Thus the value at Bob, $\hat{c}$, may no longer be the original c created by Alice.
7. Finally, both parties create a string, SAS, using a keyed hash function. Both parties XOR ($\oplus$) R with the output of a hash, keyed with K, of the value m_B.

Informally, to give an intuitive understanding of the benefits of the protocol, note that by both parties XORing R with the output of a hash of the value m_B, with the hash keyed with K, and by comparing the SAS values the two parties create, if the SAS values are equal then the following values must have been the same:

K Otherwise the hashes would output different values.

R Otherwise the result of the XOR would be different. Note the use of the commitment scheme which generates K and the order of the protocol ensures R is sent and known before the output of the hash can be computed.

$\boldsymbol{m_B}$ Otherwise the hashes would output different values.

$\boldsymbol{m_A}$ Otherwise the opening of the commitment at Bob would have resulted in a different K at Bob, which would have meant the keyed hashes would have output different values.

Thus, if the two SAS values are the same, then the messages m_A and m_B have been successfully sent between the parties without being changed by a third party.

Prasad and Saxena's adapted protocol exchanges and authenticates public keys by instantiating m_A from Figure 3 as pk_A, and, critically, the human makes the authentication assessment, thereby taking the role of Pasini and Vaudenay's authenticating channel. Thus, by comparing two short strings, the (far longer) public keys are authenticated. Note that the protocol of Pasini and Vaudenay's is well suited for exchanging public keys, as the keys are public but there must be assurances that the keys belong to the original senders, as is the goal of the Pasini and Vaudenay protocol.

The authentication assessment is made possible by transforming the constructed SAS string into a series of audible tones (beeps) or flashing lights (blinks) for the human to compare and, if the assessment is that the devices are the same, to *accept* and pair the devices. We shall call this protocol the *Beep-Blink* protocol. The adapted authentication stage of the protocol [9] is shown in Figure 4.

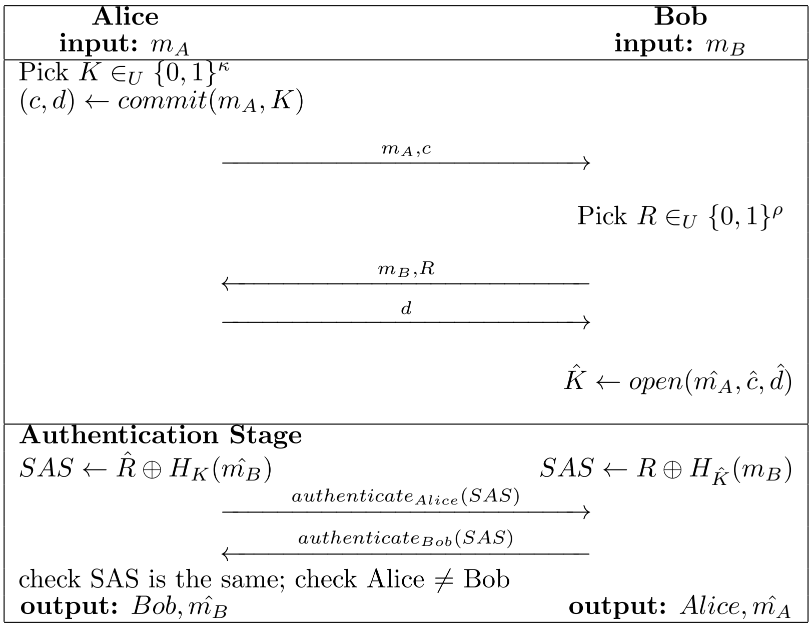
Alice
input: m_A
Bob
input: m_B
Pick $K \in_U \{0,1\}^\kappa$
$(c,d) \leftarrow commit(m_A, K)$
m_A, c
Pick $R \in_U \{0,1\}^\rho$
m_B, R
d
$\hat{K} \leftarrow open(\hat{m_A}, \hat{c}, \hat{d})$
Authentication Stage
$SAS \leftarrow \hat{R} \oplus H_K(\hat{m_B})$
$SAS \leftarrow R \oplus H_{\hat{K}}(m_B)$
$authenticate_{Alice}(SAS)$
$authenticate_{Bob}(SAS)$
check SAS is the same; check Alice $\neq$ Bob
output: $Bob, \hat{m_B}$
output: $Alice, \hat{m_A}$

Figure 3: Pasini and Vaudenay's SAS Protocol

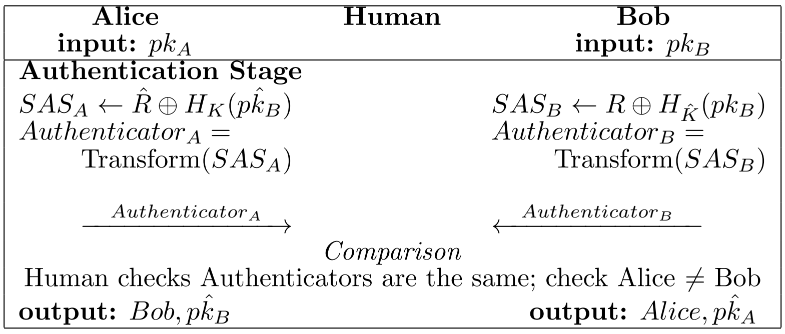
Alice
input: pk_A
Human
Bob
input: pk_B
Authentication Stage
$SAS_A \leftarrow \hat{R} \oplus H_K(\hat{pk_B})$
$Authenticator_A =$ Transform(SAS_A)
$SAS_B \leftarrow R \oplus H_{\hat{K}}(pk_B)$
$Authenticator_B =$ Transform(SAS_B)
$Authenticator_A$
$Authenticator_B$
Comparison
Human checks Authenticators are the same; check Alice $\neq$ Bob
output: $Bob, \hat{pk_B}$
output: $Alice, \hat{pk_A}$

Figure 4: Prasad and Saxena's human-assisted device pairing protocol based on SAS protocol [9]

Prasad and Saxena [9] state, "The security of our scheme is equivalent to the security of the underlying SAS protocol under the assumption that the user does not commit any errors." This leaves open to what extent human errors may impact the security of the protocol and how these can be dealt with in the security analysis. We have shown that it is necessary to go beyond the presumption of humans acting perfectly. Indeed, their human study [9] clearly showed firstly that the humans did not act perfectly, and secondly that the level of error depended on the type of *HPA* used.

An interesting philosophical point is that even 1s and 0s for voltage levels in a circuit have already undertaken a transform to be human recognisable. So the SASs of Pasini and Vaudenay could be considered *HPA*s, and the subsequent lights by Prasad and Saxena are also, though different, *HPA*s from different *HPASpace*s. In this way, the SAS-family of protocols, or the transformed SAS into beeps and blinks, can be seen as *GenHPA* algorithms, the probability distribution of which may be very well known, which output a *HPA* in the form of a visual or audible SAS.

Instead of the human being pre-loaded with a *HPA* and being sent a HPA' as the two *HPA* inputs to the *Recognise* function, HPA_A will be on device A, and HPA_B will be on device B. The formalisation for the human specific version of *Recognise*, including the concept of $W_{H,HPA}$ as the set of *HPA*s which are indistinguishable for that human, still apply. While the intention of the Prasad and Saxena protocol would seem to be that the human either *accepts* on both devices or *rejects* on both devices [9], there clearly exists the case where on one device the human selects "Accept" and on the other device the human selects "Reject". As such, for device A, their (potentially transformed) SAS is held as *HPA* and the SAS supplied by device B will be HPA'; and similarly for device B. The authentication stage of Prasad and Saxena's protocol, using our formalism, is shown in Figure 5. We now use our contribution to present a proof of the Prasad and Saxena protocol, capturing some useful human considerations.

Prasad and Saxena's protocol [9] may be seen as an instantiation of Pasini and Vaudenay's protocol [8], with the human in Prasad and Saxena's protocol playing the role of Pasini and Vaudenay's magical authentication channel. The protocol outlined in Figure 5 is a more concrete description of how the protocol of Prasad and Saxena could be implemented. Beyond capturing the inability of humans to perfectly execute a protocol, and hence that the security of Prasad and Saxena's instantiation is not at all equivalent to that of Pasini and Vaudenay, our formalisation allows for capturing and analysis of a far more realistic set of issues. For example, using the concrete protocol out-

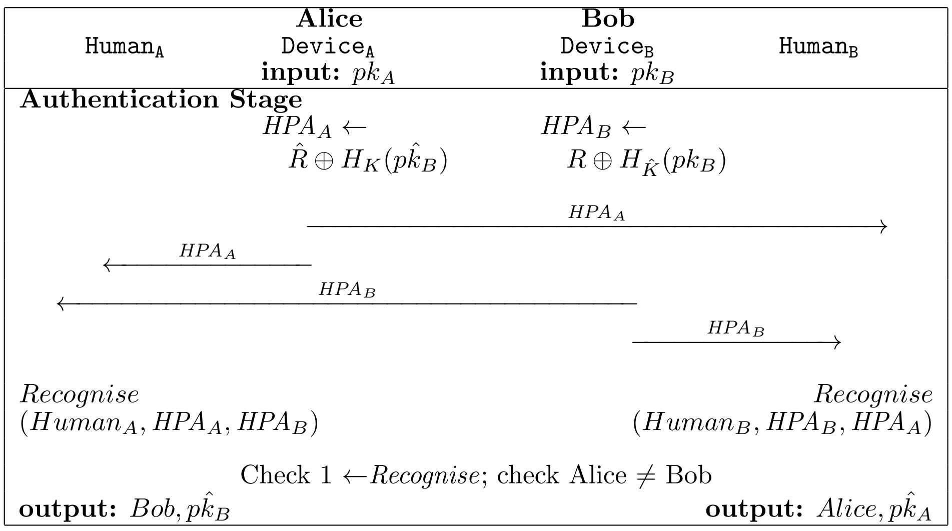
Alice
Bob
Human_A
Device_A
Device_B
Human_B
input: pk_A
input: pk_B
Authentication Stage
$HPA_A \leftarrow \hat{R} \oplus H_K(\hat{pk_B})$
$HPA_B \leftarrow R \oplus H_{\hat{K}}(pk_B)$
HPA_A
HPA_A
HPA_B
HPA_B
$Recognise(Human_A, HPA_A, HPA_B)$
$Recognise(Human_B, HPA_B, HPA_A)$
Check 1 ←*Recognise*; check Alice ≠ Bob
output: $Bob, \hat{pk_B}$
output: $Alice, \hat{pk_A}$

Figure 5: Prasad and Saxena's human-assisted device pairing protocol considered using our formalism

lined in Figure 5, there is no reason why there could not be two potentially spatially separated humans doing the comparison (one for device A and one for device B) rather than one human in the one location; and there is no reason why the protocol cannot end with device A believing it is paired with device B, without device B believing it is paired with device A (and vice versa). This lack of equality between the pairing status of the devices may be through either *Recognise* failing or else through time delays in the human making a choice.

4.2 Proof of human-assisted device pairing protocol

A Bellare-Rogaway 1993 based model [1] was used to provide a security proof for Pasini and Vaudenay's SAS protocol [8], upon which Prasad and Saxena based further human centred protocols [9]. However, in Prasad and Saxena's work, no formal security analysis could be given to the *BEEP-BEEP*, *BLINK-BLINK* and *BEEP-BLINK* variations of the underlying SAS protocol. Rather, the claim was made that the derived protocols had equivalent security to the underlying SAS protocol [8] under the assumption that the human did not commit any errors [9].

Now, with our formalisation, the security framework exists to formally analyse such protocols.

4.2.1 Adversarial model

Recall the model for short authentication string-based pairing security, described by Pasini and Vaudenay [8] based on Bellare and Rogaway's model [1]. An outline of Pasini and Vaudenay's model is described here for completeness:

Launch (n, role, x) launches a new protocol instance on node n playing *role* (e.g. either Alice or Bob) with input x. It returns a new instance tag π_n^i.

Send (π, y) sends an incoming message y to the instance π. It returns an outgoing message z, or the final output of the protocol if it completed.

Corrupt (n) injects a malicious code in node n so that its behaviour is no longer guaranteed.

These queries are standard in cryptographic models. For example, the Send query allows the adversary to run the protocol normally and to inject messages of his choice, reflecting the assumption that the adversary controls communications between protocol participants. The Launch oracle creates a unique tag π_n^i, which allows node n to have multiple protocol instances running. Corrupt allows the adversary to effectively *take over* a node, meaning that any code the adversary wishes could be injected at the corrupted node such that the node would do what the adversary wants.

In the model, the participants ID_n are located at *nodes* in the network. In Pasini and Vaudenay's proof for their protocol [8] the winning condition for the adversary in such a message cross authentication protocol is "if some instance ended on an uncorrupted node with a pair (m, ID) but no instance on the node of identity ID with input m was launched." For a complete description of the model, see [8].

Theorem 1 (Success probability of Beep-Blink protocol)**.** *Let ζ be $\mathcal{A}$'s success probability of the SAS protocol of Pasini and Vaudenay [8], then $\mathcal{A}$'s success probability of the Beep-Blink protocol of Prasad and Saxena [9] is bounded by $\zeta + Succ_{\mathcal{A}_{Device}}$.*

Proof. We employ the Shoup's game hopping proof technique [12] augmented by Dent [4]. We employ a sequence of two games, the first game being the security game for the protocol shown in Figure 5 which is the human protocol of Prasad and Saxena represented with our formalism. We transform this to the security game for the original protocol of Pasini and Vaudenay shown in Figure 3 [8], bounding the adversary's advantage between the two. As such our proof augments the proof of Pasini and Vaudenay to cater for the human considerations of the Prasad and Saxena protocol. We denote Win_i as the probability of the adversary winning game i.

Game G_0 describes the real protocol, as run by Prasad and Saxena [9], using our formalism (see Figure 5 and Figure 3). The game is played between a probabilistic polynomial time (PPT) bound adversary $\mathcal{A}$ and a simulator. The simulator simulates protocol participants as specified in the natural protocol specification, and answers all of $\mathcal{A}$'s queries.

Game G_1 describes a game which is the same as Pasini and Vaudenay's SAS protocol [8]. The difference between Game G_1 and Game G_0 is that in Game G_1, the original SAS protocol, the authentication comparison is based on equality, whereas in game Game G_0 the authentication comparison is based on our human formalism. Hence, remembering $Succ_{\mathcal{A}_{\text{Device}}}$ defined above,

$$|\Pr[\text{Win}_1] - \Pr[\text{Win}_0]| \leq Succ_{\mathcal{A}_{\text{Device}}}.$$

Therefore,

$$\Pr[\text{Win}_0] \leq Succ_{\mathcal{A}_{\text{Device}}} + \zeta.$$

$\square$

Pasini and Vaudenay [8] describe a win condition of at most $2^{-\rho}$ plus the adversary's advantage against the hash function, where ρ is the number of bits of the SAS. They approximate ρ by $\log_2 \frac{N^2R^2}{2p}$, where N is the number of participants, R is the number of runs of the protocol, and p is the attack probability, with the example given for ATM-like PIN numbers of $p = 3 \cdot 10^{-4}$. Our formalism gives more meaningful values for p, taking into account human imperfection, and allows for the comparison of representational transform techniques (such as *beeps* and *blinks*). This means a more accurate security parameter of ρ could be calculated.

In this way, a general framework may be created for all such unauthenticated key exchange protocols followed by *HPA* recognition.

5 Conclusion and Future Work

This work presents a method of accumulating data which will allow for the comparison of schemes in which the human will need to recognise some data. When represented formally using our technique, schemes can be compared using: the size of *HPASpace* (maximise), $HPASpace_H$ (maximise) and the ratio between the two (bring to equality); the size of $W_{H,HPA}$ (false positives, minimise); the size of ϵ (false negatives, minimise); and the frequency of use distribution (normalise).

We have provided an upper bound on the adversary's probability of success, both for the case of a human generated *HPA* and a device generated *HPA*. We have shown how our formalism may be included easily into existing proofs, providing a more complete model in the case of mutual authentication over TLS, and creating a formal proof of human-assisted device pairing protocols to be created for the first time. Many similar examples of protocols involving humans where our formalism will be directly useful exist. Such an example would be standard Verified-by-Visa protocol implementations, where, due to the large numbers of people and the large numbers of protocol runs, useful values for each of the variables in our formalism will be available. At the softer end of the scale, our formalism could be applied to human protocols which exist completely in the human realm, for example where a human may have to authenticate themselves to another human which is typically based on some sort of recognition.

This paper presents a significant building block in security proofs which is useful in the sense that the formalism can be used in security proofs which allow the comparison of practical and real-world implemented human authentication schemes. However, further work can be pursued in at least two main areas. Firstly, for the purposes of creating a bound on the adversary's success, in our model we have prevented $\mathcal{A}$ from selecting a $HPA^{'}$ that is outside of *HPASpace*. Allowing $\mathcal{A}$ to select a $HPA^{'}$ that is outside of *HPASpace* would increase the size of $W_{H,HPA}$ in Figure 1. Secondly, exploration and formalisation of the concept of *context*, thus making the analysis probabilistic not deterministic, will be a significant expansion of our work.

References

[1] M. Bellare and P. Rogaway. Entity Authentication and Key Distribution. In D. R. Stinson, editor, *CRYPTO*, volume 773 of *LNCS*, pages 232–249. Springer, 1993.

[2] J. Bonneau, M. Just, and G. Matthews. What's in a name? In *Financial Cryptography*, volume 6052 of *LNCS*, pages 98–113. Springer, 2010.

[3] W. E. Burr, D. F. Dodson, and W. T. Polk. NIST Special Publication 800-63 Electronic Authentication Guideline v1.0.2, 2006.

[4] A. W. Dent. A note on game-hopping proofs. *IACR Cryptology ePrint Archive*, 2006:260, 2006.

[5] S. Gajek, M. Manulis, A.-R. Sadeghi, and J. Schwenk. Provably Secure Browser-Based User-Aware Mutual Authentication over TLS. In M. Abe and V. D. Gligor, editors, *ASIACCS*, pages 300–311. ACM, 2008.

[6] S. Gajek, M. Manulis, and J. Schwenk. User-aware provably secure protocols for browser-based mutual authentication. *International Journal of Applied Cryptography*, 1(4):290–308, 2009.

[7] N. J. Hopper and M. Blum. Secure Human Identification Protocols. In C. Boyd, editor, *ASIACRYPT*, volume 2248, pages 52–66. Springer, 2001.

[8] S. Pasini and S. Vaudenay. SAS-Based Authenticated Key Agreement. In M. Yung, Y. Dodis, A. Kiayias, and T. Malkin, editors, *Public Key Cryptography*, volume 3958 of *Lecture Notes in Computer Science*, pages 395–409. Springer, 2006.

[9] R. Prasad and N. Saxena. Efficient device pairing using "human-comparable" synchronized audiovisual patterns. In *ACNS*, volume 5037 of *Lecture Notes in Computer Science*, pages 328–345, 2008.

[10] S. E. Schechter, A. J. B. Brush, and S. Egelman. It's no secret. measuring the security and reliability of authentication via "secret" questions. In *IEEE Symposium on Security and Privacy*, pages 375–390. IEEE Computer Society, 2009.

[11] A. Shostack and A. Stewart. *The New School of Information Security.* Addison-Wesley Professional, Upper Saddle River, N.J., 2008.

[12] V. Shoup. Sequences of games: a tool for taming complexity in security proofs. *IACR Cryptology ePrint Archive*, 2004:332, 2004.

[13] S. Vaudenay. Secure communications over insecure channels based on short authenticated strings. In V. Shoup, editor, *CRYPTO*, volume 3621, pages 309–326. Springer, 2005.