Stayin' Alive:
Aliveness as an alternative to authentication

Jonathan Anderson and Robert N M Watson

University of Cambridge Computer Laboratory
{jonathan.anderson,robert.watson}@cl.cam.ac.uk

Abstract. Authentication protocols attempt to discern whether or not a user is who she says she is based on what she has, is or knows. In many situtations, however, such as protecting Wikis from robots and Distributed Hash Tables from sybils, identity is less important than liveness: it's not *who you are* that matters, it's whether or not you are alive. We propose extensions to the Kerberos authentication which allow systems to test whether or not they are interacting with a real person, optionally disregarding their identity. We demonstrate how such extensions could be used to support realistic user interactions with requiring shared definitions of global identity.

1 Introduction

Last year, one of the authors of this paper attempted to contact a professor at a notable US university in order to arrange a lunch appointment. This professor had — perhaps out of an attempt to avoid spam — created a system by which one could not e-mail him unless one had demonstrated their ability to receive e-mail by submitting their own (publicly accessible) e-mail address into a Web form and replying to a generated single-use address. This system may be effective at blocking spam, but it is also effective at dissuading legitimate would-be corresponders and collaborators from initiating conversation.

Disregarding this particular implementation, there is an insight to be gained from this situation. Sometimes, e.g. when attempting to fend off unsolicited mass e-mail, it is not important to know whether or not a potential correspondant's identity corresponds to a unique name within a global namespace; the important thing is whether or not that correspondant is really there at all. When controlling access to a Wiki, we do not need to know the name or government-issued ID number of

a contributor, we would simply like to know that edits come from a real person. We may also like to know that a long string of thoughtful, well-researched edits have come from the same person, but that knowledge does not require a globally-agreed-on notion of the user's identity.

Today, many websites attempt to use CAPTCHAs to distinguish anonymous users from anonymous bots. This approach is not applicable to other networked services such as mail servers, anonymous FTP servers or distributed hash tables since the CAPTCHA protocol executes within the context of a Web session. Outside of the Web context, existing authentication and authorization tools are a poor fit for the general case of networked services.

In this paper, we propose extensions to the well-known Kerberos suite of tools that allow users to establish communications channels by demonstrating liveness without requiring *a priori* agreement on globally-unique identities. Our extensions use a new security protocol primitive called *unboxing*, and we demonstrate how traditional attacks against authentication protocols only exist in a *denatured* form.

2 Kerberos

The Kerberos protocol [4,5,6], a simplified form of which is depicted in Figure 1, provides authentication and authorization for networked services such as file servers and SMTP servers. Kerberos allows security-sensitive authentication logic to be separated from the "business logic" of servers.

For instance, let us consider a scenario in which a user Bob wishes to avail of services (e.g. a mail delivery queue) provided by an SMTP server. Rather than implement its own authentication service, the SMTP server relies on a centralized Authentication Service (AS) to authenticate Bob. Bob first communicates with the AS, proving knowledge of a secret key derived from a password, and is granted a cryptographic Ticket-Granting Ticket (TGT). Bob can then reveal the TGT to a Ticket-Granting Server (TGS) in exchange for a Service Ticket (ST). This ticket is a packet encrypted under a key shared by the TGS and the Service Server (SS), i.e. the SMTP server in our case. The ticket certifies the SS that Bob's identity has

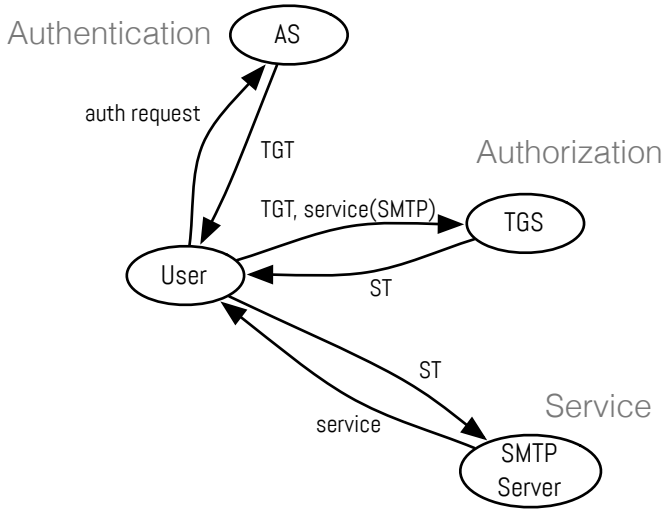

Fig. 1. Kerberos v5 protocol.

been verified by the AS and provides a session key which Bob and the SS can use in support of their confidentiality and integrity requirements.

A simplified model of the exchange between Bob and the AS is:

$$\texttt{KRB\_AS\_REQ}: \quad B \xrightarrow{B,A} A \tag{1}$$

$$\texttt{KRB\_AS\_REP}: \quad B \xleftarrow{TGT=\{B,k_{BT}\ldots\}_{k_{AT}},\{k_{BT}\}_{k_B}} A \tag{2}$$

$$\texttt{KRB\_TGS\_REQ}: \quad B \xrightarrow{B,S,TGT,S,\{B,t\}_{k_{BT}}} TGS \tag{3}$$

$$\texttt{KRB\_TGS\_REP}: \quad B \xleftarrow{\left\{ST=\{B,k_{BS},\ldots\}_{k_{TS}},k_{BS}\right\}_{k_{BT}}} TGS \tag{4}$$

$$B \xrightarrow{\{T,\ldots\}_{k_{BS}}} S \tag{5}$$

in which B and S are the names of Bob and the SMTP server, t is the current time (to demonstrate freshness), k_B is Bob's "reply key" (generated from Bob's password) and k_{XY} is a symmetric key used for communication between principals X and Y. Using this protocol, Bob can demonstrate knowledge of his password to the AS without having to reveal it to the SMTP server, both Bob and the SMTP server can know

to whom they are speaking and each server can focus on the one clear function for which it has been designed: authentication, authorization or delivering mail.

In the vanilla Kerberos protocol, message 1 can be replayed by an attacker, inducing the AS to send the attacker a `KRB_AS_REP` message. This message does not reveal any key material, since keys are encrypted under k_B and k_S, but such a replay could aid a known-plaintext attack on k_B, which is usually based on a user-chosen password. In order to prevent such an attack, the AS can require "pre-authentication" data to be sent as part of the `KRB_AS_REQ` message. This data can be as simple as the current timestamp encrypted under k_B (to demonstrate knowledge of Bob's password-derived key), but it can also be an arbitrary multi-round challenge-response protocol between a client-side plugin and a server-side plugin.

Kerberos solves an important problem: it allows a trusted third party to be leveraged by clients and servers to set up secure communications with mutual authentication. This neatly fulfills the requirements of mail *relays* that only forward mail for known users, but it fails to address the mail *submission* problem: an SMTP server delivering mail directly to local recipients does not need to be able to authenticate mail senders, but it will use authentication as a proxy certification that the sender is a real user and not a spam bot.

We propose a Kerberos extension to support the direct validation of "aliveness" claims as well as — or instead of — traditional authentication. This extension uses existing spaces in the Kerberos protocol intended for preauthentication data to carry the results of an interactive Turing test approximation, allowing services to be accessed by users not on the basis of *who* they are, but *whether* they are.

3 Liveness Extensions

Using the existing Kerberos preauthentication framework [2] and plugin APIs, we propose to extend the user's interaction with the AS according to the communication graph depicted in Figure 2. We introduce additional messages into the initial Kerberos authentication exchange, adding a demonstration of *aliveness* via some instantiation of Turing's *imitation game* [8].

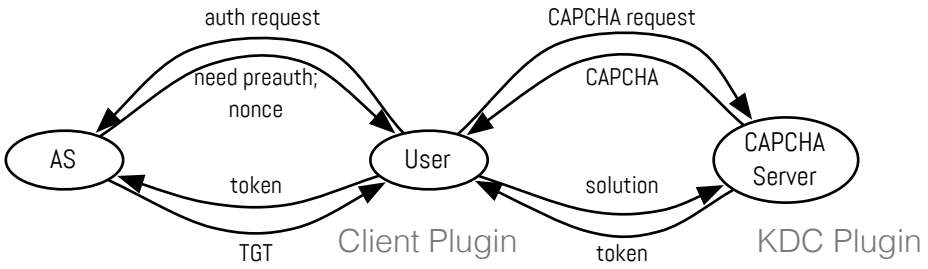

Fig. 2. Proposed Kerberos extensions.

3.1 Notation for Human Readability

Security protocols typically use a curly-brace-and-subscript notation to indicate that a message is encrypted, e.g. $\{B, k_{BT}, t\}_{k_A}$. In the protocol that follows, we introduce a new notation for messages that are not encrypted, but are rather encoded in a form that can be easily decoded by humans but is less easily decoded algorithmically.

The most common form of such a message is the CAPTCHA [1], but other forms of "human computation" are possible. The creator of the CAPTCHA, Luis von Ahn, denoted a *human algorithm game* which is easy for humans but hard for computers by a one-way function, $G(x) = y$ [9]. We define *human decoding* as a human algorithm game in which x is an encoding of information — such as a CAPTCHA — that can easily be decoded by humans but not computers. We will denote such a message with the notation $[x]$ and expect that a human can trivially decode $[x]$ to x.

3.2 Protocol

Our complete protocol with "liveness" extensions to Kerberos is depicted in Figure 3.

In the initial communication between Bob and the authentication service Alice, Bob requests a Kerberos Ticket-Granting Ticket (TGT) from Alice for the server S using the identity "anonymous". Alice replies with a standard Kerberos error code `KDC_ERR_PREAUTH_REQUIRED` which specifies that Kerberos authentication must be preceeded by a "pre-authentication" step, which in this case is an interactive proof of aliveness to a CAPTCHA service.

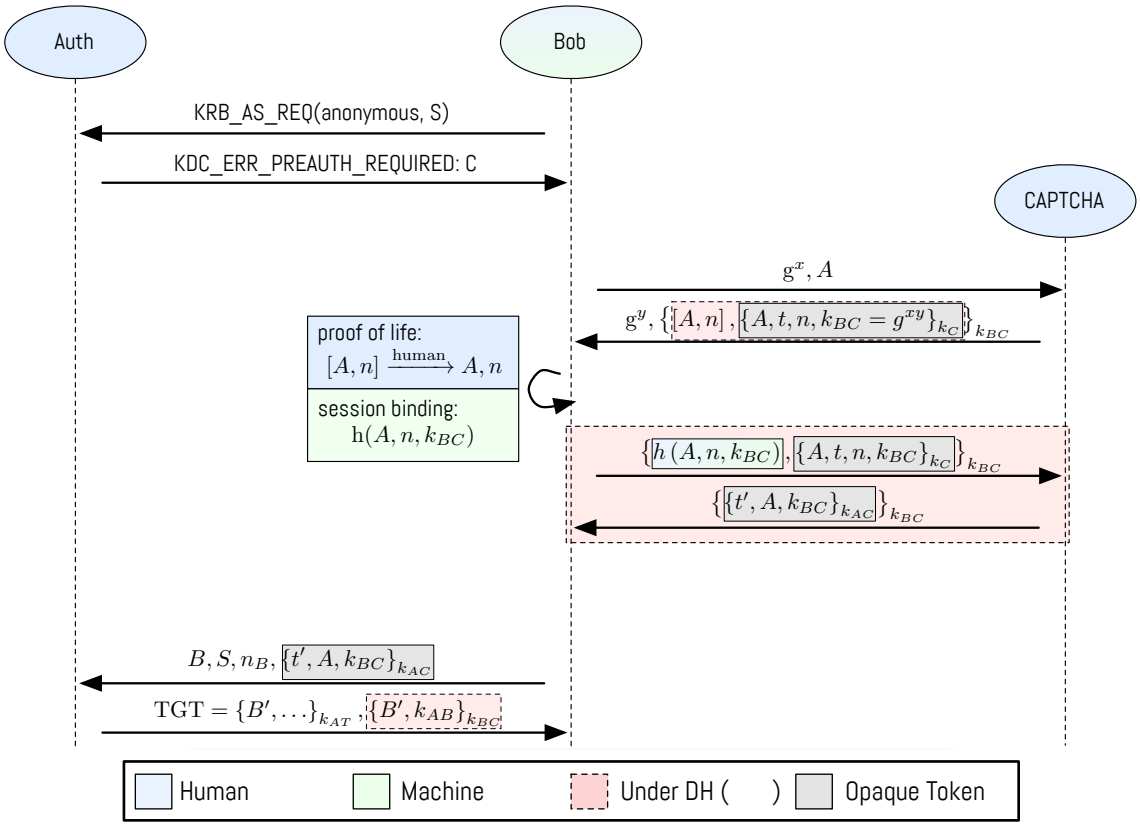

Fig. 3. Kerberos "pre-authentication" via aliveness.

Bob's interaction with the CAPTCHA server consists of two round trips which convey a challenge and response within a Diffie-Hellman–negotiated channel that binds the response to the session key.

Bob begins his interaction with the CAPTCHA server by initiating a Diffie-Hellman exchange (g^x) and providing the name of the realm A to which he would like to authenticate. This name will be included in the token which the CAPTCHA server eventually provides to A in order to prevent one *aliveness token* from being spent at multiple sites.

The CAPTCHA server replies with the second half of the Diffie-Hellman exchange (g^y) in the clear and a user-decodable challenge $[A, n]$, as well as a state cookie. This cookie contains the name A, the time t that the challenge was sent, the nonce n that the challenge was generated from and the negotiated session key k_{BC} being used to protect communications between Bob and the CAPTCHA service. The use of cookies allows the CAPTCHA server to be stateless apart from a replay cache.

The challenge $[A, n]$ sent to the user contains both the name of the Kerberos authentication realm and a dictionary word generated from the random nonce n. An example of such a challenge is depicted in Figure 4,

Fig. 4. A user-decodable CAPTCHA: $[A, n]$.

Bob's response to this challenge consists of C's state cookie and a cryptographic hash of three elements: the name A from the challenge, the word n from the challenge and the session key k_{BC} being used to communicate with the CAPTCHA server. The session key is included in order to bind the response to the channel over which it is given, mitigating the middleperson attack described in Section 4.

Finally, the CAPTCHA server gives Bob a token $\{t', A, k_{BC}\}_{k_{AC}}$ that Bob can present to A as evidence that he solved a CAPTCHA for service A at time t'. This token also contains the session key that Bob negotiated with the CAPTCHA service; A can use this key to bootstrap a new shared secret k_{AB}. The rest of the protocol run is standard pre-authenticated Kerberos with one exception: as well as granting Bob a Ticket-Granting Ticket (TGT) and performing a password update to k_{AB}, A must also assign a new pseudonym B' which can be used to identify this particular "anonymous" user in the future.

At the conclusion of this protocol run, B possesses a Ticket-Granting-Ticket (TGT) in the name of an anonymous user who was believed to represent a real, live person at time t'. Such a user might now be granted limited access to a public FTP server, temporary access to an SMTP server for the purposes of mail delivery (but not mail relay) or rate-limited access to an IRC server or a Wiki. Without explicitly naming a user, our Kerberos realm has established a secure communications channel with "that user". If the user fulfils the traditional Kerberos security assumptions (e.g. not revealing secret keys to anyone), there can be continuity in future communications between the unnamed user and existing Kerberized infrastructure.

4 Denaturing a Middleperson Attack

denature, v. /dɪˈneɪtjʊə(r)/ /diː-/ (Oxford English Dictionary)

1. *trans.* To render unnatural. Obs.
2.
   (a) To alter (anything) so as to change its nature; e.g. to render alcohol or tea unfit for consumption.
   (b) *Biochem.* To modify (a protein) by heat, acid, etc., so that it no longer has its original properties.

Since Bob shares no long-term secrets with the Authentication Server (AS), the AS cannot set up a confidential channel between Bob and the CAPTCHA server. Thus, an active adversary Mallory can launch a middleperson attack between Bob and the CAPTCHA server.

After inserting himself, however, Mallory is unable to exploit his position in the conventional way. We refer to this as a *denatured* attack: the new assumptions implicit in anonymous "authentication" render the original properties of the attack unfit for use. The attack exists, but it no longer has its original properties.

Mallory is able to "steal" Bob's CAPTCHA by not forwarding it, but rather solving it himself. If he does, however, he has effectively carried out a Denial of Service attack against Bob but has not breached Alice's security policy: to Alice, one unknown anonymous user is as good as another. If, on the other hand, Mallory forwards the CAPTCHA to Bob, Bob's solution does not help Mallory to authenticate himself, since it is bound to the session key k_{BM} and Mallory needs a solution bound to k_{MC} in order to acquire a "CAPTCHA solved" token. Thus, Mallory has exactly the same power to authenticate to Alice whether or not he is attacking Bob: this is a *denatured* middleperson attack.

5 Related Work

The Kerberos network authentication protocol [5,6] is based on an earlier authentication protocol by Needham and Schroeder [4]. These protocols allow users to receive authorisation tokens from a trusted authentication and authorisation server. These tokens can be presented to networked services such as file servers and mail servers, proving to those servers that the user has the right to use them without revealing the user's password to them. Kerberos relies on a strong notion of user identity, although primitives have been introduced to deal with some degree of anonymity.

Zhu, Hartman and Leach's RFC 6112 proposes anonymous credentials for Kerberos [10]. This would allow users to be granted tickets that identify them only by realm, not personal identity. These tickets provide a natural expression of the "identity" expressed by the above aliveness extensions: they identify an identity that it believed by a particular AS to have some property, which in our case is to *be alive*.

RFC 6112 would also allow users to obtain tickets from a TGS without any authentication whatsoever that prove nothing about the ticket bearer's identity. Lacking any authentication, this protocol cannot prevent automatic ticket harvesting; a

Sybil-prevention feature such as that afforded by our liveness extensions would be a natural fit for this part of the RFC.

Holt and Seamons' Nym is a pseudonymous credential system that allows users to convert pseudoname tokens into TLS client certificates [3]. For instance, a user could request a pseudonymous token from a server that enforces a "one token per IP address" policy. This token can then be exchanged for a client certificate that is accepted by a service such as a wiki. The aliveness extensions that we propose applying to Kerberos could fit equally well into the Nym framework: the token service is agnostic towards the policy that it enforces. Indeed, the authors of Nym observed that tokens might be earned by solving CAPTCHAs or puzzles; what our extensions add is the specific cryptographic protocol that should be used to earn the token.

Tsang *et al.*'s Nymble is a larger system for providing credentials that are unlinkable by default but linkable when the credentials are misused [7]. Nymble includes a pseudonymous credential acquisition phase, but the focus of the work is elsewhere: blacklisting pseudonyms. As with Nym, Nymble could benefit from the inclusion of our aliveness protocol in its credential acquisition phase.

6 Conclusion

The Kerberos protocol and its existing extensions provide a useful service: given a trusted third party, clients and servers can authenticate each other and set up secure communication channels. The protocol is a natural fit for services such as mail relays that only provide service to users who are known to the system administrator according to a pre-arranged unique name. It is a less natural fit for services that are offered to anonymous users: in Kerberos, authorisation requires authentication.

We have extended Kerberos, using an existing "pre-authentication" framework and plugin architecture, for scenarios which the existing model does not fit well: services such as chat systems or Wikis that do not necessarily require a globally-unique username but do need to know whether or not the user is alive. The protocol uses a technique and notation called *unboxing*: an analogue to encryption that is trivial for humans but difficult for machines.

The protocol is secure against eavesdropping by a passive adversary and indifferent to active attacks: there are no pre-existing shared secrets to steal and a middleperson cannot impersonate the authenticating user. A middleperson can authenticate themselves to the server instead of the client he is attacking, but the attack is *denatured*: rendered unfit for use and without its original properties. The server is indifferent to which anonymous user is currently connected; the only property it is required to verify is that the connected anonymous user is alive.

Our protocol allows network services to be provided to anonymous-but-alive users. User authorization is not done on the basis of *who* one is, but *whether* one is.

Acknowledgements

We would like to thank Paul Syverson, Matt Blaze and Ross Anderson for helpful comments and questions during the workshop at which this protocol was presented.

References

1. Ahn, L. V., Blum, M., Hopper, N. J., and Langford, J. CAPTCHA: using hard AI problems for security. In *EUROCRYPT 2003: Proceedings of the 22nd Annual International Conference on the Theory and Applications of Cryptographic Techniques* (May 2003), Springer-Verlag, pp. 294–311.
2. Hartman, S., and Zhu, L. A Generalized Framework for Kerberos Pre-Authentication. RFC 6113, Apr. 2011.
3. Holt, J. E., and Seamons, K. E. Nym: Practical Pseudonymity for Anonymous Networks. Tech. Rep. 2006-4, Internet Security Research Lab (ISRL), Brigham Young University, June 2006.
4. Needham, R. M., and Schroeder, M. D. Using encryption for authentication in large networks of computers. *Communications of the ACM 21*, 12 (Dec. 1978).
5. Neuman, B., and Ts'o, T. Kerberos: an authentication service for computer networks. *IEEE Communications Magazine 32*, 9 (1994), 33–38.
6. Neuman, C., Yu, T., Hartman, S., and Raeburn, K. The Kerberos Network Authentication Service (V5). RFC 4120, July 2005.
7. Tsang, P., Kapadia, A., Cornelius, C., and Smith, S. Nymble: Blocking Misbehaving Users in Anonymizing Networks. *IEEE Transactions on Dependable and Secure Computing 8*, 2 (2011), 256–269.
8. Turing, A. M. Computing Machinery and Intelligence. *Mind 59* (Oct. 1950), 433–460.
9. von Ahn, L. Human Computation. Tech. Rep. CMU-CS-05-193, Carnegie Mellon University, Pittsburgh, PA, 2005.
10. Zhu, L., Hartman, S., and Leach, P. Anonymity Support for Kerberos. RFC 6112, Apr. 2011.