Technical Report OUCS-2004-16a

Attacks on Homage Anonymous Group Authentication Protocol

Author:
Sonil Gohil

Department of Computer Science, University of Otago

Status: Report of work done under Departmental summer bursary project.

Attacks on *Homage* Anonymous Group Authentication Protocol

Sonil Gohil

06 February 2004

Abstract. This report describes an implementation of the attacks proposed by *Jaulmes* and *Poupard* [3] on *Homage* [2]. All attacks are shown to be valid. It also gives a modified version of the protocol which nullifies the attacks by adding more constraints on some parameters and an extra phase where the authority must prove the knowledge of his secret key. Finally, it proposes a format where this protocol could be used within a biometric system to ensure it is no longer vulnerable to the revelation of the private key by a user. It does this without losing the efficiency of *Homage* and still maintaining anonymity at verification.

1 Introduction

The *Homage* protocol [1] allows a group member to be authenticated anonymously. The identity of a group member remains unknown to the group authority when the user is being authenticated.

This is important as it prevents the leaking of personal information and stop an authority from tracking down private and confidential information. It allows organizations to only access the information they need without letting the organization get any other information that might compromise confidentiality of the user.

It satisfies the security properties needed for anonymous group authentication. Non-members of a group are rejected and members are successfully authenticated without their identities being exposed.

2 Original *Homage* Protocol

2.1 Outline of *Homage*

The protocol has a registration phase where a user gives his public key to the authority and the authority gives the user a certificate for future authentication. Later in authentication, the user, (via some zero knowledge scheme) proves the knowledge of his private key and thus gets authenticated.

2.2 Definitions

Authority is someone who controls a group and can issue certificates to users and verify them.
User(s) are members of the group.
H(x) is a publicly known secure hash function.
p publicly known prime modulus.
g publicly known generator used in creating public key and certificates for users.
u publicly known constant coprime to $p-1$.
w, z are the private secret keys of Authority.
x is the private key of a user.
$v \equiv u^w \pmod{p-1}$ is publicly known.
$y \equiv g^x \pmod{p}$ is public key of a user.
a is a random number coprime to $p-1$ chosen by the authority when registering a user. It is different for each user.
b, c are random numbers chosen by a user at authentication. c should be coprime to $p-1$.
$d \equiv u^b \pmod{p-1}$ is calculated at authentication by user and is used in sending verification information.

2.3 Registration of a User

The registering user sends his public key y to the authority. The authority issues the user his/hers certificate:

$\alpha_1 \equiv (gy^z)^a \pmod{p}$.
$\alpha_2 \equiv a^w \pmod{p-1}$.

2.4 Authentication of a User

A user wanting to authenticate calculates the following values and sends them to the authority. The values, in no way, determine the identity of the user.

$\beta_1 \equiv \alpha_1^{cd} \pmod{p}.$
$\beta_2 \equiv \alpha_2 v^b \pmod{p-1}.$
$\beta_3 \equiv g^c \pmod{p}.$

The authority then calculates the following values:

$\gamma_1 \equiv \beta_2^{1/w} \pmod{p-1}.$
$\gamma_2 \equiv \beta_1^{1/\gamma_1} \pmod{p}.$
$\gamma_3 \equiv (\frac{\gamma_2}{\beta_3})^{1/z} \pmod{p-1}.$

The authority then sends $H(\gamma_3)$ to user who verifies that it is indeed equal to $H(y^c \pmod{p})$.

Then, through some discreet log algorithm such as **A.1**[1], the user proves the knowledge of his private key in γ_3.

2.5 How is a User Authenticated?

The user calculates and sends the following β values to the authority at authentication.

- $\beta_1 \equiv \alpha_1^{cd} \equiv (gy^z)^{acd} \pmod{p}$
- $\beta_2 \equiv \alpha_2 v^b \equiv (ad)^w \pmod{p-1}$
- $\beta_3 \equiv g^c \pmod{p}$

The authority calculates the following γ values.

- $\gamma_1 \equiv \beta_2^{1/w} \equiv ((ad)^w)^{1/w} \equiv ad \pmod{p-1}$
- $\gamma_2 \equiv \beta_1^{1/\gamma_1} \equiv ((gy^z)^{acd})^{\frac{1}{ad}} \equiv (gy^z)^c \pmod{p}$
- $\gamma_3 \equiv (\frac{\gamma_2}{\beta_3})^{\frac{1}{z}} \equiv y^c \equiv \beta_3^x \pmod{p}$

Now, as both parties have $\gamma_3 = \beta_3^x$ and β_3, the user can now prove the knowledge of his private key x using some log algorithm like the one mentioned in [**A.1**][1]. The user can only be authenticated if he is able to prove this.

As there was some confusion at the presentation of this paper [3] and some indication of contrary findings from the original author [8], it was decided to implement the attacks in order to verify their success against the original protocol.

3 Attacks on Homage

The following attacks are described in Jaulmes and Poupard [3].

3.1 Attack 1

The attack works by the authority placing members in subgroups at registration and choosing the parameter a for their certificates based on the subgroup the user has been put in. Then, when a user tries to authenticate, the authority calculates the multiplicative order of β_1 in modulus p which reveals the subgroup the user belongs to. More specifically, if there is a modulus p such that $\frac{p-1}{2}$ has many factors (i.e. $p = 1 + 2 * \prod_{i=1}^{\eta} q_i$), then the authority can have 2^{η} subgroups. The authority calculates $a = a' * \prod_{i=1}^{\eta} q_i^{v_i} \pmod{p-1}$ where a' is a random number and v is a binary vector of form $(v_1, ..., v_\eta)$. The multiplicative order of β_1 in p is the product of the q_i's not used in a.

Lets consider the following system:

$p = 521 = 1 + 2 * (2 * 2 * 5 * 13)$.
$g = 23, z = 3, w = 13, u = 9$.

A user(Peggy) has private key $x = 19$. When Peggy registers, Victor(Authority) places her in subgroup 2 $(0, 0, 0, 1)$ by using $q_4 = 13$. He chooses $a' = 3$ and calculates a to be 39. Consequently, Peggys certificates are:

$\alpha_1 = 377$ and $\alpha_2 = 39$.

Now, Peggy wants to authenticate. She chooses $b = 15$ and $c = 7$ making sure that c is relatively prime to $p - 1$. She sends her verification information to Victor where $\beta_1 = 388$.

Victor now calculates the multiplicative order of β_1 in p which is 20 and is equal to $(2*2*5)$. Victor now knows that these 3 primes were not used in a and that therefore, the vector must be $(0, 0, 0, 1)$ corresponding to subgroup 2.

As suggested in [3], this attack can be easily avoided by having a safe prime and by verifying that certificate α_2 is coprime to $p-1$.

3.2 Attack 2

This is the most serious of the proposed attacks. It allows the authority at verification to determine the identity of the authenticating user with time complexity linear in the number of users.

For this attack to be valid, the parameter $\frac{q-1}{2}$ must have more than one prime factor. Say $q = 2r_1r_2 + 1$ where r_1 and r_2 are two large primes. The authority chooses parameter u of order r_1. Consequently, $v = u^w \pmod{p-1}$ is also of order r_1 with high likelihood. The authority has a list of all α_2's issued and the user it corresponds to. When a user authenticates, the authority calculates $\zeta \equiv \frac{\beta_2}{\alpha_2} \pmod{p-1}$ for all α_2's he has. The authenticating user will have ζ of order r_1 while other users will have ζ of order r_1r_2 or $2r_1r_2$ with high probabilty.

Say we have $r1 = 19$ and $r_2 = 29$ and so $q = 1103$ and $p = 2207$. The parameter $u = 17$ is of order r_1. Other parameters are $w = 10, g = 23$ and $z = 15$.

Say we have the following users:

Name	x	a	α_1	α_2
Peggy	19	13	1544	1553
Anna	29	17	633	479
Alice	31	21	637	1109
Sue	17	15	1110	813
Marie	37	9	1800	655
Linda	41	25	1849	1063

For the following table, each row corresponds to the authenticating user. The first column represents the users secret values b and c, and the next six columns show the value $\zeta \equiv \frac{\beta_2}{\alpha_2} \pmod{p-1}$ and its order in modulus $p-1$ for every user.

Name	*(b, c)*	*Linda*	*Marie*	*Sue*	*Alice*	*Anna*	*Peggy*
Peggy	(6, 11)	(1809, 551)	(1363, 551)	(1885, 551)	(73, 551)	(2025, 551)	**(501, 19)**
Anna	(14, 27)	(77, 551)	(2975, 551)	(429, 551)	(1325, 551)	**(613, 19)**	(753, 551)
Alice	(18, 7)	(1939, 551)	(311, 551)	(1979, 551)	**(677, 19)**	(195, 551)	(1715, 551)
Sue	(12, 33)	(83, 551)	(1721, 551)	**(1723, 19)**	(1285, 551)	(1377, 551)	(1929, 551)
Marie	(26, 49)	(195, 551)	**(1731, 19)**	(141, 551)	(2009, 551)	(1667, 551)	(1831, 551)
Linda	(14, 35)	**(613, 19)**	(1335, 551)	(579, 551)	(693, 551)	(1213, 551)	(2127, 29)

The ζ value is of order r_1 for the authenticating user each time as seen above. For example, when Alice authenticates, her ζ value is of order r_1 in modulus $p-1$ while the ζ values for other users is of order $r_1r_2 = 551$. But for non-authenticating users, its not neccesarily of order r_1r_2 as seen in the case when Linda tries to authenticate, the ζ value for Peggy is of order r_2. None the less, this attack is very successful and easy to carry out.

Lets consider when $p = 839$ where $r_1 = 11$ and $r_2 = 19$. The other parameters are $w = 7, g = 19$ and $z = 15$.

Say we have the following registered users:

Name	x	a	α_1	α_2
Peggy	37	5	86	191
Anna	19	3	449	511
Alice	29	7	732	637

Then at authentication:

Name	b	c	β_2	*Peggy*	*Alice*	*Anna*
Peggy	9	11	773	**(753, 11)**	**(69, 11)**	(575, 209)
Anna	6	9	777	(377, 209)	(739, 209)	**(13, 11)**
Alice	8	5	121	**(571, 11)**	**(59, 11)**	(771, 209)

The last 3 columns of this table represent ζ and its order in modulus $p-1$ for that user.

As seen above, there is some ambiguity here. This method in this case is giving false positives in some cases. This may be due to the size of parameters b and c. For example, when Anna authenticates, the authority successfully determines her identity. When Peggy and Alice authenticate, there is ambiguity as to whether Peggy or Alice is authenticating. This example only has 3 users. Its very likely that there could be more ambiguity in a system with more users.

Several attacks of this nature where $\frac{q-1}{2}$ has several factors can be successfully implemented. There is a simple fix to this. Parameters u and v are publicly known. If q is of such form, then u and v can be verified to be of order $\frac{q-1}{2}$. Having u of this order would mean that v is also of that order with high probablity. If this can be implemented(i.e. u and v are verified to be of order $\frac{q-1}{2}$), then the authority will not be able to be successful in this attack.

3.3 Attack 3

This attack is almost exactly the same as attack 1 described in section [**3.1**] but uses different z values to distinguish subgroups. Then during authentication, the authority determines what z was used and thus getting the subgroup of the user.

Lets consider the following system:

$p = 2207, g = 19, w = 5, u = 77.$

A user(Peggy) has private key $x = 53$. When Peggy registers, Victor(authority) chooses $a = 13$ and places her in subgroup corresponding to $z = 17$. Consequently, her certificates are $\alpha_1 = 827$ and $\alpha_2 = 685$.

When Peggy wants to authenticate, she chooses $b = 5$ and $c = 13$ making sure that c is relatively prime to $p - 1$. She sends her verification information to Victor where $\beta_1 = 1788, \beta_2 = 965, \beta_3 = 1820$. Victor calculates γ_1 and γ_2 to be 1261 and 827. Now he has to calculate γ_3. He calculates $(\frac{827}{1820})^{1/z} \equiv 72^{1/z} \pmod{p}$. He tries different z values corresponding to each subgroup. Of course, he can get a different z than the one corresponding to Peggy but then when he sends H(γ_3) to Peggy, their values will not agree. In this case, when he tries the z value corresponding to Peggy, he gets $\gamma_3 = 2039$. When the user confirms that their two hash values for γ_3 agree, Victor then knows the subgroup Peggy belongs to. There is some ambiguity as to which method to use to find the z value. There may be more definite ways of determining the correct z value than the one used here.

4 Proposed Protocol by Jaulmes and Poupard

This alternative protocol removes the verification phase from the original protocol and provides a proof with each certificate that it has been correctly computed at registration. A commitment scheme is used to commit the private key z of the authority. This is committed with a public key $Y = g^z h^j \pmod{p}$ where j is random and the discreet logarithm of h (which is of order q) in basis g is unknown to the authority. Opening a commitment would involve revealing the commited value and the random value used in the commitment.

But because the user is authenticated at the same time as registration, the authority still has the public key of that user and therefore knows who the user is. Furthermore, once the certificates are computed, the authority chooses t randomly and sends to the user the following:

$$\begin{aligned}
&C_1 = commit(t)\\
&C_2 = commit(t * a \pmod{q})\\
&C_3 = \alpha_1^t \pmod{p}\\
&C_4 = (ta)^w \pmod{q}
\end{aligned}$$

A challenge based system is used and based on the answer to that challenge, either C_1 or C_2 are opened and the user and the authority prove their knowledge of their respective private keys. As h must be unknown to the authority and is used in these commitments, a third party must be involved when t is committed or when commitment of t is opened. This seems highly unreasonable as the above challenge based process must be repeated n times to ensure the chances of cheating are one in 2^n. As two values are commited each time (C_1 or C_2) and one commitment is opened, the third party must be involved $3n$ times a user tries to authenticate. Although the complexity is still linear, the involvement of the third party to this degree unnecessarily complicates the operation of the protocol in practice.

5 Modified Protocol

5.1 Outline

This protocol is virtually identical to the original *Homage* but with more constraints on its parameters. It also has an additional phase where the authority must prove the knowledge of his secret key *z*. This is proved at verification *before* the user sends any verifying information. The user requests the authority for verification and the authority proves the knowledge of secret key z. Then the user sends verification information. We assume that all communication is via secure channels and can not be traced to the user.

5.2 Definitions

p publicly known prime modulus. $p = 2q + 1$.
q publicly known prime. $q = 2r + 1$.
r publicly known large prime.
g publicly known generator used in creating public key and certificates for users.
g_2 publicly known constant of order q in modulus p.
H(x) is a publicly known secure Hash function.
u publicly known constant coprime to $p - 1$.
w, z are the private secret keys of Authority.
x is the private key of a user.
$v \equiv u^w \pmod{p-1}$ is publicly known.
$y \equiv g^x \pmod{p}$ is public key of a user.
a is a random number coprime to $p - 1$ chosen by the authority when registering a user. It is different for each user.
b, c are random numbers chosen by a user at authentication. c should be coprime to $p - 1$.
$d \equiv u^b \pmod{p-1}$ is calculated at authentication by user and is used in sending verification information.

5.3 Registration

The registering user sends his public key y to the authority. The authority issues the user his/hers certificate by sending the following values:

$g_1 \equiv y^a \pmod{p}$.
$E_1 \equiv g^a \pmod{p}$.
$I_1 \equiv g_1^z \pmod{p}$.
$\alpha_1 \equiv (gy^z)^a \pmod{p}$.
$\alpha_2 \equiv a^w \pmod{p-1}$.

NOTE: $\alpha_1 \equiv E_1 \times I_1 \pmod{p}$ and so the user can verify this or calculate α_1 himself. α_1 has to be divided into g_1, E_1 and I_1 as it allows the authority to prove his knowledge of the secret key z when a user authenticates. The protocol is still the same as in [2] except that now, the first part of the certificate α_1 has been broken into g_1, E_1 and I_1 to allow the authority to prove the knowledge of his secret key.

5.4 Authentication

5.4.1 Proof of Secret Key z

This proof is described in [3].

User chooses r_1 such that $g_2 \equiv {g_1}^{r_1} \pmod{p}$ and calculates $I_2 \equiv I_1^{r_1} \pmod{p} \equiv (g_1^{r_1})^z \pmod{p}$.

Authority chooses random r_2 in [1, q] and sends to user:

$x_1 \equiv {g_2}^{r_2} \pmod{p}$.

User randomly chooses r_3 and sends it to authority.

Authority Computes and sends:

$s \equiv r_2 + (r_3 \times z) \pmod{q}$.

User checks:

${g_2}^s \pmod{p} \stackrel{?}{=} x_1 \times {I_2}^{r_3} \pmod{p}$.

If this is valid, proof of z is successful. As this proof does not reveal any information on secret key z to the user, it is zero knowledge.

Security Analysis. This subprotocol is sound and complete. An authority who is accepted with probabilty larger that $1/B$, where B is the range from which r_2 is selected, can be used by an extractor which computes the discreet logarithm of I_2 in basis g_2. Both x_1 and s do not reveal any information on the secret key z. As g_2 is publicly known and the user never shares I_2,the identity of the user remains anonymous.

If the authority cheated and identified subgroups by using different z's, then he would have to guess correctly the parameter z every time a user authenticates so the chances of that are $1/n$ where n is the number of subgroups. If there are less subgroups, then the chances of the authority guessing correctly are higher but then there must be more members in each subgroup. And for this to be valid, the authority would have to guess correctly every time any user tries to authenticate. This subprotocol can be repeated several times to ensure additional security especially if the range from which r_2 is selected is not too large.

5.4.2 User Authentication

This part of the protocol remains the same as described in section [**2.4**].

5.5 Validity of Attacks on Modified Protocol

Attack 1. As prime modulus p is of the form $p = 2q + 1$ with $q = 2r + 1$, $\frac{p-1}{2}$ has only one factor and therefore makes this attack useless. If p were not of this form, a user can prevent such an attack by making sure that certificate α_2 is coprime to $p - 1$.

Attack 2. Variations of attack 2 are implemented on this modified version of the protocol to see if the protocol sustains those attacks.

Lets consider when $r = 41, q = 83$ and $p = 167$. The parameter $u = 17$ is of order r in $p - 1$. The other parameters are $w = 10, z = 15$ and $g = 23$.

Say we have the following registered users:

Name	x	a	α_1	α_2
Peggy	19	13	141	77
Anna	29	17	22	161
Alice	31	21	127	9

Then at authentication:

Name	b	c	β_2	*Peggy*	*Alice*	*Anna*
Peggy	6	11	123	(21, 41)	(69, 41)	(75, 41)
Anna	14	27	153	(99, 41)	(17, 41)	(69, 41)
Alice	18	7	17	(11, 41)	(131, 41)	(63, 41)

The name in each row is of the authenticating user and the name in each column represents the check against that user. The values under the column names are $\zeta \equiv \frac{\beta_2}{\alpha_2} \pmod{p-1}$ and its order in modulus $p-1$. This attack is not valid when u is chosen of order r as seen above.

Let us now consider when $u = 37$ is of order $\frac{q-1}{2}$ with other parameters the same as above. The registration will be the same as above.

At authentication:

Name	b	c	β_2	*Peggy*	*Alice*	*Anna*
Peggy	6	11	87	(27, 41)	(65, 41)	(49, 41)
Anna	14	27	21	(121, 41)	(113, 41)	(29, 41)
Alice	18	7	25	(65, 41)	(95, 41)	(161, 41)

The attack is again invalid.

Attack 3. For this attack to be valid, the authority must know which secret key z was used for the authenticating user when he proves the knowledge of z. As he has no way of knowing which subgroup the authenticating user belongs to prior to this proof, the only success he can have of cheating is by guessing which z was used. If he guesses the incorrect one, his prove of the knowldge of z would be unsuccessful. So if he divides the group into n subgroups, the chances of him guessing correctly which subgroup the user belongs to is $1/n$. If there are less subgroups, then the chances of the authority guessing correctly are higher. However there must be more members in each subgroup then. For this to be valid, the authority must guess correctly every time any user tries to authenticate. The chances of the authority successfully cheating everytime any user authenticates is pretty slim.

6 Biometric Systems

6.1 Requirements

This protocol can be used within a Biometric system. But how? It must still satisfy the requirements of *Homage*. This is particularly difficult to implement as any biometric data from a user identifies that user directly and thus compromises the requirements of *Homage*.

We need a system such that:

(a) Identity of User is not compromised.

(b) Must ensure that the bio-data(encrypted) given, belongs to a valid user.

(c) Must ensure that bio-data(encrypted) given, corresponds to the private key proved by the user.

6.2 Setup

B - Something that identifies biometric data of user. It is unique and can not be faked. It reveals identity of user.

B' - B encrypted with public key y.

T - B encrypted with private key x. Does not reveal identity.

T' - T encrypted with key β_3^x at verification. Does not reveal identity.

Encryption should be such that:

$Encrypt(Encrypt(B, x), y) \neq Encrypt(Encrypt(B, y), x)$ where $Encrypt(A, b)$ means encrypting A with key b.

When users are issued with their private keys, the users submit T and B' and the key authority, who knows private and public keys x and y, verifies that T and B' are encrypted with x and y. This must be done in the presence of the key authority and the user to ensure the biometric data that was encrypted belongs to that user. The key authority sends a list of the public keys of the users and their corresponding B''s to the authority. The key authority also sends a separate list of T's to the authority which can not be traced to the users in the authority's userlist or to the list of public keys.

During registration, the user can send his public key as well as B' to ensure that he is a valid user. The authority then sends the user his certificates.

When a user wants to authenticate, he sends his verification information and T to the authority. He also sends T and the parameter c to a black box. In practical terms, this can be done by ensuring that the black box is according to some standard such as FIPS 140-2 [7]. Anything happening inside the black box is invisible to the authority and everyone else. The authority then checks to see if the T he received is in the list of T's in his record list. If it is not, then this authenticating person is either not a user, or has used a private key belonging to some other user to encrypt his biometric data.

The authority sends β_3 and T to the black box. Inside the black box, there is a list of public keys y and corresponding T for that public key. A check is made inside the black box to ensure that the two T's received are the same. The value β_3 is then compared against g^c. These two values should also be the same. After this check, $\beta_3^x \equiv y^c \pmod{p}$ is calculated using the y value for the given T and the c value from the user. Then T is encrypted with β_3^x to T' as this can now be calculated. Then T' is sent to the authority. The authority himself has a copy of $\gamma_3 = \beta_3^x$ and uses this to encrypt the T, given to him by user, to T'. The authority compares the two values of T'. If there are the same, it means the private key proved by the user corresponds to that particular user and ensures that that particular user was present when authenticating. All communication is via secure channels. The system must be such that any biometric data or B, T or T' can not be stored and these parameters are only calculated at the time of verification.

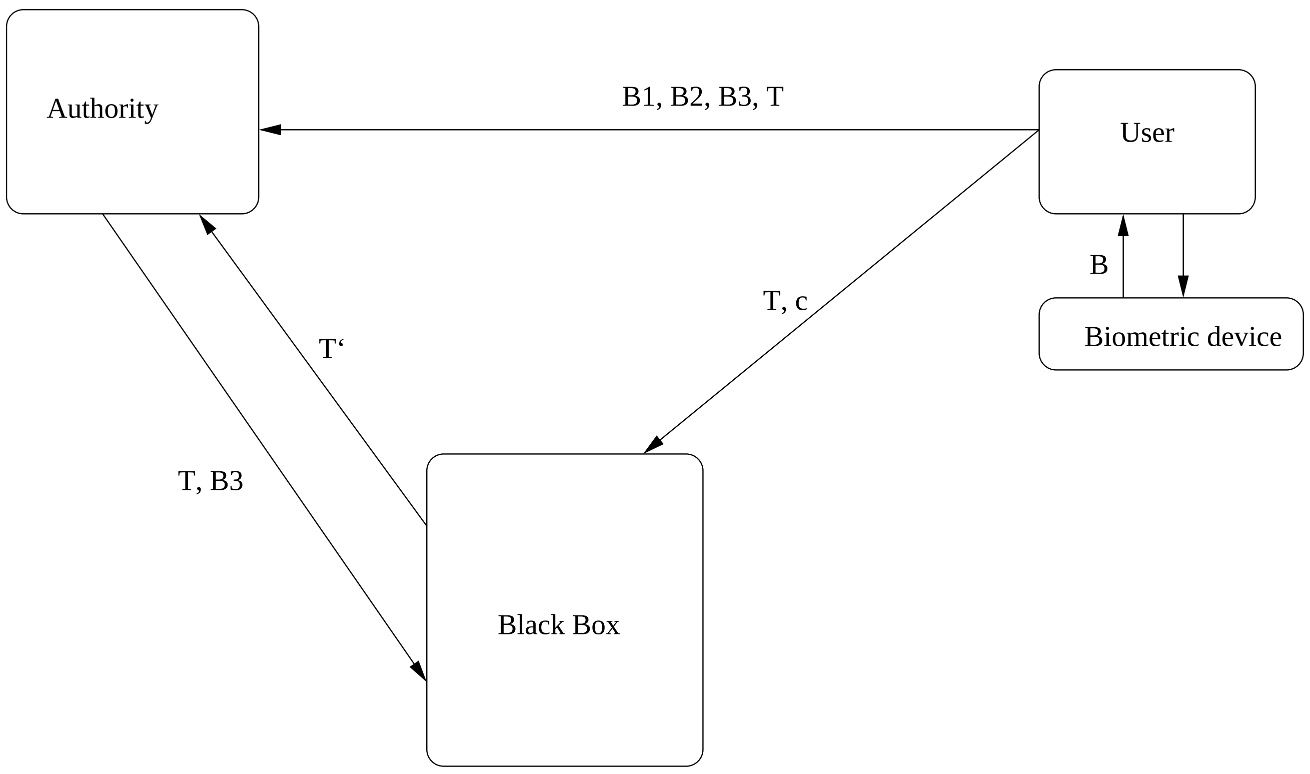

Figure 1: Biometric Layout of Homage

References

1. B. Handley. Resource-Efficient Anonymous Group Identification. In *Prepoceedings of Financial Cryptography* 2000.
2. B. Handley. Resource-Efficient Anonymous Group Identification. In *Financial Cryptography 2000*, LNCS. Springer-Verlag, 2001. *(Personal communication with author in December 2003).*
3. E. Jaulmes, G. Poupard. On The security of *Homage* Group Authentication Protocol.
4. T. P. Pederson. Non-Interactive and Information-Theoretic secure Verifiable Secret Sharing. In *Crypto '91.* Springer-Verlag, 1992.
5. S. Halevi, S. Micali. Practical and Provably-Secure Commitment Schemes from Collsion-Free Hashing.
6. A. Khalili. Notes on Verifiable Secret Sharing Schemes (VSS). *November 2002.*
7. Information Technology Laboratory, National Institute of Standards and Technology, 2001. Security Requirements for Cryptographic Modules. Federal Information Standards Publication FIPS PUB 140-2. National Institute of Standards and Technology, Gaithersburg, MD.
8. B. Handley. (Personal communication)

A. Zero Knowledge Proof

A.1 Knowledge of a Discreet Log Algorithm

NOTE: All arithmetic in this log algorithm is modulus p.

User wants to convince Authority that he knows x in $\gamma_3 \equiv \beta_3^x$. To do this, user chooses some t and sends authority β_3^t. The authority sends the user a 0 or 1.

- If it is a 0, user tells authority what t is, and the authority verifies that the value sent before is β_3^t.
- If it is a 1, user tells authority $k \equiv \frac{x}{t} \pmod{p-1}$. Authority then verifies that $\gamma_3 \equiv \beta_3^{tk}$.

As there is a 50 percent chance of the user cheating each time, the above log algorithm must be repeated n times so the chances of cheating are 1 in 2^n.