Replacing Username/Password with Software-Only Two-Factor Authentication

Michael Scott

Certivox Labs
mike.scott@certivox.com

Abstract. It is basically a solved problem for a server to authenticate itself to a client using standard methods of Public Key cryptography. The Public Key Infrastructure (PKI) supports the SSL protocol which in turn enables this functionality. The single-point-of-failure in PKI, and hence the focus of attacks, is the Certification Authority. However this entity is commonly off-line, well defended, and not easily attacked. For a client to authenticate itself to the server is much more problematical. The simplest and most common mechanism is Username/Password. Although not at all satisfactory, the only onus on the client is to generate and remember a password – and the reality is that we cannot expect a client to be sufficiently sophisticated or well organised to protect larger secrets. However Username/Password as a mechanism is breaking down. So-called zero-day attacks on servers commonly recover files containing information related to passwords, and unless the passwords are of sufficiently high entropy they will be found. The commonly applied patch is to insist that clients adopt long, complex, hard-to-remember passwords. This is essentially a second line of defence imposed on the client to protect them in the (increasingly likely) event that the authentication server will be successfully hacked. Note that in an ideal world a client should be able to use a low entropy password, as a server can limit the number of attempts the client can make to authenticate itself. The often proposed alternative is the adoption of multifactor authentication. In the simplest case the client must demonstrate possession of both a token and a password. The banks have been to the forefront of adopting such methods, but the token is invariably a physical device of some kind. Cryptography's embarrassing secret is that to date no completely satisfactory means has been discovered to implement two-factor authentication entirely in software. In this paper we propose such a scheme.

1 Introduction

In 2003 a scheme was proposed by Kim et al. [14], which promised an "ID-based password Authentication Scheme using Smart Cards and Fingerprints". Scott [21] showed how it could be comprehensively broken by an attacker who passively eavesdropped a single transaction.

A much more recent 2012 paper by Martinez-Pelaez and Rico-Novella [16] successfully cryptanalyses a scheme due to Sood, Sarje and Singh, described in

their paper "An Improvement of Liao at al's Authentication Scheme using Smart Cards" [24]. This scheme is essentially a two-factor authentication proposal. From the abstract of [16] ".. we show that Sood at al.'s scheme is still vulnerable to malicious user attacks, man-in-the-middle attack, stolen smart-card attack, off-line ID guessing attack, impersonation attack and server spoofing attack..". Chasing down though the cited papers one finds a wasteland of similarly broken protocols. Many other schemes have fallen into the break-fix-break-fix cycle which generates a lot of literature, but rather less confidence. The cryptographic establishment seems to have largely abandoned this area of research as a kind of wild-west of cryptography, where no-one survives very long.

This problem was recognised by Hao and Clarke [13] who explain it by stating that "The past thirty years of research in the area of authenticated key exchange has proved that it is incredibly difficult to get even a single factor based KE scheme right. Designing a multi-factor AKE protocol can only be harder."

However the reaction of the casual reader might be this: How come at this stage in the development of cryptographic protocols there appears to be no satisfactory and widely accepted solution to the problem of easy-to-use and secure two-factor authentication?

1.1 Password Authenticated Key Exchange

This is a well researched area, and many proven methods for PAKE have been proposed [33]. However most require the maintainance of a "password file", or "password verifier file", which is exactly what an attacker looks for in a hacked server, and which allows them to completely unlock the server's security. Most commonly this file consists of tuples of $\langle Username, Salt, H(Salt|Password)\rangle$ where Salt is a random number, $H(.)$ is a one way hash function and $|$ indicates concatenation. As is well appreciated, unless the password is chosen to have high entropy, the attacker will succeed with an off-line dictionary attack in finding the password by searching through a dictionary at computer speeds. Only passwords that are outside of the dictionary will survive this attack. This leads to the tautology that a high entropy password must be agreed with the server in advance. Recall that the whole purpose of a PAKE is to mutually authenticate and agree a high entropy cryptographic key.

One alternative is to issue each client with a smart card, and to store password related information here rather than on the server. This has the added benefit of adding a second factor to the authentication process - the smart card itself. However for this to work it seems that the smart card component must maintain full smart card functionality. If its security if broken then the system fails. See Yang et al. [34] for a successful implementation of this approach. However smart cards are expensive. Also such schemes invariably require the server to maintain a long term secret s. And again if the server is hacked s might be revealed, unlocking the security of the entire system.

1.2 Multi-factor Authentication

Multi-factor authentication commonly consists of (a) something we have, (b) something we know, and (c) something we are. The form in which it is most familiar to us would be as the two-factor ATM bank card and a 4 digit PIN number. Here most of our discussion will be in the context of two-factor authentication, but with some consideration for support for the third biometric factor.

The "something we have" factor can be a physical token storing static data, perhaps in the form factor of data recorded on a magnetic strip, or in a QR code, or the familiar USB stick. Or it can be a smart-card. Note that a smart-card will have extra functionality in that it can have its own protected secrets and computing ability, and it is unclonable. However a smart card is much more expensive, and losing it requires expensive replacement.

The "something we know" factor is a password. However passwords come in two distinct flavours. The high-entropy password that is now more commonly demanded of us, for example the password which must have eight or more characters and involve both upper and lower case letters, and at least one numeral. Then there is the low entropy password, for example the 4-digit PIN. In the sequel we will use the word password exclusively for high entropy passwords, and the word PIN for the low entropy password. An alternative way to distinguish them is to observe that the latter is easily found by an off-line dictionary attack, whereas the former, hopefully, is not. An off-line dictionary attack is where an attacker can (at computer speeds) run through a dictionary of possible passwords/PINs and easily detect when they have found the right one.

The "something we are" factor is captured as a biometric, perhaps a fingerprint or an iris scan. Biometrics can also be high or low entropy. However they are commonly inexact, and a certain range of values for a biometric measurement might be regarded as acceptable. High entropy biometrics tend to be expensive, whereas low entropy fingerprint scanners for example can be quite cheap.

One reason for the multiplicity of proposed schemes is that there are many different ways of instantiating these three factors. The entropy status of the password (high or low) must be decided, but is sometimes not made clear [29]. Many schemes propose a smart-card as a token. However they often also consider the case where the security of the smart-card is breached and its secrets revealed, in which case it is no better than a passive token [32], [29]. We restrict ourselves to the simplest, easiest and cheapest scenario (and hence the most practical). We will assume

1. a static clonable token,
2. a low entropy 4-digit PIN number
3. (Optionally) a low entropy biometric.

Note that since our token is static, we are in a position to provide a software-only solution. For the client our proposed scheme is in fact closely analagous to the tried and trusted ATM card (with static data recorded on a magnetic strip), and associated PIN number. But one that works over the much more hostile domain that is the internet.

1.3 Desirable features

Many authors have composed helpful lists of desirable features of such protocols. Tsai et al. [26] provide a list of 9 security requirements and 10 goals for such schemes. However their review of schemes available at their time of writing reveals that all are disappointing. Liao et al. [15] came up with a list of 10 properties, which Yang et al. [34] reduced to 5. Recently Wang [32] came up with a list of 8 attacks that such a scheme should resist, and identified 3 categories of potential attacker.

Motivated by this prior art, here we give our own list of desirable conditions that our ideal scheme will meet.

1. The protocol should result in the client and the server being mutually authenticated, and deriving a mutual cryptographic key.
2. No PIN related data is stored on the server. Nevertheless a server should be able to assist a client in recovering their forgotten PIN.
3. The underlying Authenticated Key Exchange is immune to "Key Compromise Impersonation". That is the server cannot impersonate a client, and a client cannot impersonate the server.
4. The client should be able to change their PIN locally without involving the server.
5. An attacker who gains possession of the authentication server's secrets should only be able to (a) set up a false server, and (b) given a client token determine their PIN. Note that this is basically the best that can be hoped for, for any such scheme.
6. The server should be able to identify the extent of any small error ϵ in the client's secret. This will facilitate the inclusion of an inprecise biometric measurement as a factor.
7. The scheme should support the property of "forward secrecy".
8. The scheme should be truly "multi-factor". If there are n factors involved the loss of $n-1$ factors should not be sufficient to permit the final factor to be found. Of course this must also be true for insider attacks: For example a client equipped with a valid token and PIN and who captures the token of another client, should be unable to determine their PIN number.
9. PIN guessing attacks can only be carried out on-line (and therefore can be monitored and prevented by the server).
10. The overall system should not have a single-point-of-failure.

Our literature review would suggest that no such scheme currently exists. Note that condition 3 is clearly impossible to meet for any scheme that allows the server to manage client registration. We adopt in its entirety the adversarial model of [34], which assumes that an adversary who can completely control communications between the servers and its clients, and who may have possession of any number of tokens and associated PINs of users other than the individual under attack.

1.4 Existing proposals

Next we look a little closer at the proposals to date. Most are based in a setting where there are just two parties involved, the clients and the server. The former enter into some initial interaction with the server (a Registration phase) and are issued with some credentials. Invariably this requires the server to have in their possession some master secret which will be involved in the authentication process. Such schemes we would suggest are unlikely to improve on the current situation, as if (when) the server is hacked this secret may be discovered, and the security of the whole scheme will unravel.

A close inspection of many of the simpler proposals reveals at their heart the simple idea of using the one way hash of the clients identity concatenated to a server master-secret, as the basis for client authentication and the establishment of an authenticated secret key. So for client identity ID and server master secret s, the protocol would be built on the high entropy mutual secret $H(ID|s)$, using as a hash function for example the standard SHA256 algorithm. This secret would be stored by the client on their token (masked by a password), and generated on the fly by the server as needed. Such schemes often also include a Diffie-Hellman component to provide forward secrecy. Multiple combinations of these simple ideas have been proposed [35], [15], [34], [29] but most have not survived prolonged scrutiny. The ingenious idea of Yang et al. [34] is to in effect use $H(ID|s)$ as the password in a PAKE. But as already pointed out this requires smart-card functionality for a token, and a long term server secret s and as they state "the secrecy of s is of utmost important because the security of the entire system essentially relies on the security of s". But a hack of the server may reveal s which would be even more lethal than stealing a password file.

A striking feature of most proposals is that often little effort is made to provide a proof of security, which certainly helps to explain why so many schemes are so quickly broken. However those proposals which do provide some kind of proof of security tend to do poorly when measured against our 10 desirable properties. And of course a proof of security is only as good as its assumptions. For example in 2010 Stebila et al. [25] proposed a multi-factor password-authenticated Key exchange. However it does not meet our condition 2, and so a successful hack of the server reveals all static non-high-entropy passwords. The proposal by Yang et al. [34] does not meet condition 3. A recent proposal by Wang [32] does not consider the possibility of an insider attack (our condition 8) and therefore not surprisingly falls to it. The scheme of Pointcheval and Zimmer [17] does not satisfy our condition 2, and recently Hao and Clarke [13] have discovered problems with it based on deficiencies of their formal model.

The lack of a satisfactory solution to date perhaps mirrors the situation with respect to Identity-Based Encryption (IBE). Although this concept was well known since first proposed by Shamir in 1984 [23], using standard PKI constructs no satisfactory scheme for IBE was ever found that didn't violate one important condition or another. It was only in 2001 through the use of the then novel construct of the cryptographic pairing that practical protocols became

possible [5]. So perhaps it is unsurprising that our proposed solution exploits the properties of cryptographic pairings.

The possibility of a better solution based on pairings is hinted at by Yang et al. [34] who identify the paper [20] as describing "a protocol which can be extended to provide explicit mutual authentication and satisfy all the properties given in [15]".

Our solution amounts to the first software-only method for two factor authentication. It satisfies all 10 of the desirable properties that we have identified above and in its simplest form requires only a software token and an easily memorised PIN number.

2 A poor man's protocol

First we describe a scheme in which the server not only handles day-to-day logins by clients, but also registers its own clients. As we will see it is a much better idea to separate the role of server from the responsibility of registration. Clearly any scheme that combines these roles cannot satisfy our condition 3 – an attacker who captures all of the servers secrets can clearly register themselves as any client. Indeed it is not hard to see that a security breach of the server allows the successful attacker a wide range of damaging exploits. Nevertheless such a simple protocol may find uses in lower security applications.

The scheme described here, based on an idea from Wang and Ma [28], requires a standard elliptic curve, with a fixed generator point G of prime order q. The server generates a large random master secret s, and a public key $Y = sG$. Each registering client is issued by the server with the server's identity ID_s, the server's public key Y and an individual cryptographically strong secret $H(s, ID_i, t_i)$, where $H(.)$ is a hash function, ID_i is the client identity, and t_i is their time of registration. The Server stores a table of $< ID_i, t_i >$ for all registered users.

The client immediately splits their secret into two parts, a token $T = H(s, ID_i, t_i) - \alpha$ and α a 4-digit PIN number.

To log onto the server, the protocol proceeds as shown in Table 1.

The protocol should finish with a standard key confirmation exchange so that both parties are assured that they are using the same key before communication proper commences.

The trick here is to combine a simple hash-based idea (the distribution of the mutual secret $H(s, ID)$), with a static Diffie-Hellman protocol based on the server's public key, and an ephemeral Diffie-Hellman to provide forward secrecy. All three ideas are subtly combined so that each cannot be attacked individually. The main threat is an attacker who has gained access to the token T and its contents and who can attack a valid communication and then come away with enough information to launch an off-line dictionary attack on the PIN. But it appears that such an attack won't work here.

A hacker who accesses s can clearly completely break the system. A defence might be to place the calculation of $H(s, ID_i, t_i)$ inside a secure module which

Alice - identity ID_a	**Server - identity** ID_s
Generate random $u < q$	Generates random $v < q$
Calculate $P = uG$, $Q = uY$	
Alice supplies token T and PIN α	
Calculate $K = H(Q, T + \alpha)$	
$ID_a, P, K \rightarrow$	If ID_a not found in table, drop connection
	Calculate $Q = sP$
	If $K \neq H(Q, H(s, ID_a, t_a))$, drop connection
	Calculate $S = vG$, $R = vP$
	$key = H(ID_s, ID_a, Q, R, K, S)$
Calculate $R = uS$	$\leftarrow S$
$key = H(ID_s, ID_a, Q, R, K, S)$	

Table 1. Simple Two-Factor Authentication Protocol

generates and protects s. However a lunchtime attack that gains access to the registration apparatus without necessarily discovering s, can do nearly as much damage.

3 Exploiting Standard Methods of PKI

We would ideally like a method not dissimilar to PKI, where both the clients and the server are enrolled in the scheme by a largely off-line and well protected third party trusted authority. By separating out the roles of the enrollment or registration from the role of the server the loss of server secrets, while clearly damaging, should hopefully not be as catastrophic.

Several proven methods have been reported for one-factor AKE using the methods of PKI. The basic idea is to issue users with a digital signature of their identity which they can then use in the key exchange.

Consider the provably secure Identity-Based Diffie-Hellman scheme of Fiore and Gennaro [7]. Here a trusted authority choses a random group generator g, and a random master secret x and issues a public key $y = g^x$. Then it issues a Schnorr signature [19] on a proffered identity as, (r_{ID}, s_{ID}) by choosing a random k and calculating $r_{ID} = g^k$ and $s_{ID} = k + xH(ID, r_{ID})$. The key exchange proceeds as in Table 2.

Both participants end up with the same key $k = g^{ab}$.

Now let us try to adapt this scheme to the client-server setting and to include a PIN number as a second factor. We immediately face two problems. First this protocol is intrinsically peer-to-peer rather than client-server. Therefore no

Alice - identity ID_a	**Bob - identity** ID_b
Generates random $a < q$	Generates random $b < q$
$u_A = g^a$	$u_B = g^b$
$ID_a, r_A, u_A \rightarrow$	$\leftarrow ID_b, r_B, u_B$
$k = (u_B r_B y^{H(ID_b, r_B)})^{a+s_A}$	$k = (u_A r_A y^{H(ID_a, r_A)})^{b+s_B}$

Table 2. Identity-Based Diffie-Hellman – Fiore and Gennaro [7]

matter how we go about embedding a PIN number such a scheme must fall to an insider off-line dictionary attack. Basically anyone who has possession of a token and PIN can set themselves up as a "server". Now if they steal another's token they can off-line perform key exchanges with their "server" until they find the PIN that works. Furthermore even non-insiders can break the system. Basically they steal a token and try every possible PIN until the token value is revealed as a valid signature. The validity of signatures can be easily verified as the certification authority's public key is available to all.

As the authors themselves say "The user can verify the correctness of its secret key by using the public key y and checking the equation $g^{s_{ID}} = r_{ID}.y^{H(ID,r_{ID})}$". Although implied as a feature of the scheme, for us it represents the problem: The existence of such an equation means that a PIN cannot safely be used in conjunction with the secret signature, because this equation would allow an off-line dictionary attack to recover it. In the scheme that we will suggest, no such equation exists. The only way to verify the correctness of the secret key should be to use it to complete a key agreement with a genuine server.

These problems with exploiting standard PKI methods for two-factor authentication are also pointed out by Wang [32].

4 Pairings and PINs

With the advent of pairings new solutions became possible. A pairing works on a special pairing-friendly elliptic curve [2], [8], with three groups of the same prime order q, normally denoted $\mathbb{G}_1 \times \mathbb{G}_2 \rightarrow \mathbb{G}_T$

Here we assume the type-3 pairing [10] where $\mathbb{G}_1$ and $\mathbb{G}_2$ are distinct, and where $C = e(A, B)$, $A \in \mathbb{G}_1$, $B \in \mathbb{G}_2$, and $C \in \mathbb{G}_T$. Note that elements from these groups cannot be mixed. This is very important for us as we intend to place clients in $\mathbb{G}_1$ and servers in $\mathbb{G}_2$.

The main significant property of pairings is that of bilinearity

$$e(aA, bB) = e(bA, aB) = e(A, B)^{ab}$$

Assume the existence of an independent Trusted Authority (TA) with its own master secret that is not required on-line – it is only responsible for off-line enrollment/registration and issuing of client and server ID-based secrets. This

provides an extra layer of security and limits the damage caused by the loss of client or server long-term secrets.

Assume ID_a and ID_s are Alice's identity and the server's identity respectively. $H_1(.)$ is a hash function that hashes to a point of order q in the group $\mathbb{G}_1$, $H_2(.)$ is a hash function that hashes to a point of order q in the group $\mathbb{G}_2$. Then both the client Alice and the server are issued with secrets sA and sS respectively, where $A = H_1(ID_a)$, $S = H_2(ID_s)$ and s is the TA's master secret for use with a particular server. Of course knowing A and sA does not reveal s, as it is fully protected by a difficult discrete logarithm problem.

Consider now the simple SOK non-interactive key exchange algorithm [18], by which Alice and the server can simultanously mutually authenticate and derive a common key. Alice calculates it as $k = e(sA, S)$, and the server calculates its as $k = e(A, sS)$, By bilinearity both are clearly the same. But now Alice extracts a PIN α of her choosing from her secret, to divide it into the token part $((s-\alpha)A$, and the PIN part αA. Clearly, when these are added together the full secret can be reconstituted. This simple idea (originally suggested in [20]) forms the basis for our proposed two-factor authentication scheme.

The XDH assumption was first informally implied in [20], and is now widely used. The XDH assumption formally states that

1. The discrete logarithm problem (DLP), the computational Diffie-Hellman problem (CDH), and the computational co-Diffie-Hellman problem are all intractable in $\mathbb{G}_1$ and $\mathbb{G}_2$.
2. There exists an efficiently computable bilinear map (pairing) $\mathbb{G}_1 \times \mathbb{G}_2 \rightarrow \mathbb{G}_T$.
3. The decisional Diffie-Hellman problem (DDH) is intractable in $\mathbb{G}_1$.

The intuition is that despite the fact of an efficiently computable type-3 pairing, $\mathbb{G}_1$ acts as a "normal" Diffie-Hellman group. The DDH problem can be described thus: Given a generator G, aG, bG and cG determine if $c = ab$.

Consider now an insider attacker Bob with identity ID_b who is issued with sB where $B = H_1(ID_b)$ and who gains possession of $(s-\alpha)A$. If the hash function has performed its job correctly A and B are independent generators of order q in the cyclic group $\mathbb{G}_1$, where $B = wA$ for some unknown w. The only way to exploit his inside information is to keep trying a guess g and adding gA to $(s-\alpha)A$ until he can use his inside knowledge of sB to identify when $(s-\alpha)A + gA = sA$. Observe that on a type-1 pairing where $\mathbb{G}_1 = \mathbb{G}_2$ this is easy as he could use the fact that $e(A, sB) = e((s-\alpha)A + gA, B)$ when $g = \alpha$. However on a type-3 pairing this is not possible according to the XDH assumption, as it implies the ability to solve the Decisional Diffie-Hellman (DDH) problem in $\mathbb{G}_1$.

Proof: Substitute $A = wB$ and assume an Oracle which if given A, wA, xA and swA can determine if $x = s$. Then such an Oracle can be used to solve the DDH problem. Simply input G, aG, bG and cG and our Oracle will determine if $b = c/a$ or $c = ab$.

So in practice Alice can extract a PIN α of her choosing from her secret, to divide it into the token part $((s-\alpha)A$, and the PIN part αA. Clearly, when these are added together the full secret can be reconstituted. A captured token is compatible with any possible PIN, and therefore useless without it.

Howevever we must be very careful if we are to retain this vital property. PIN extraction cannot be used with many pairing-based protocols, for example, Boneh and Franklin's IBE scheme [5], or the Chen and Kudla authenticated Key exchange [6] (Protocol 1), or any scheme which requires as part of its public parameters a generator point P and $P_{pub} = sP$ in G_2 (or any scheme implemented on a type-1 pairing, for example the PSCAb scheme of [32]). In such a case if Charlie were to capture Alice's token containing $(s-\alpha)A$, Charlie could quickly find her PIN by testing all g until

$$e((s-\alpha)A + gA, P) = e(A, sP)$$

This is another example of the off-line dictionary attack, and it is quite deadly.

If the server secret sS is ever leaked (that is revealing S and sS), or indeed **any multiple of a known point by s in G_2 is leaked**, then those values can be used to determine the PIN associated with a stolen token. We will call this a Master-Secret Server-side Re-use attack (MSSR) It is of course only common sense that this should be possible. If the server secret is discovered, the discoverer can set up their own false server, and try every possible PIN against a stolen token, and thus discover the PIN.

Note that all clients who access any server validated by the same TA using the same s, are at risk if just one of those servers in compromised. For this reason each individual server should ideally be associated with a different master secret s.

Interestingly we can deliberately create a circumstance where the server can launch an off-line dictionary attack on the PIN, and exploit it as a useful feature.

A server offers a simple PIN-recovery service to the client (and so fully satisfies our condition 2). The client A sends $X = H(e((s-\alpha)A + gA, S))$ to the server S via a secure channel, which requires only the token and their incorrect guess of the PIN g. Then the client presumably goes to some lengths to prove their identity (mother's maiden name etc.). Once that is done to the server's satisfaction, the server goes off-line and calculates $Y = H(e(A, sS - iS))$ for all possible i until he gets a match $X = Y$. Then $i = \alpha - g$. This difference between the guess g and the correct PIN α can then be sent back to the client by email, or some other method. Note that this does not reveal the PIN α to the server.

It may be observed that the Trusted Authority's master secret represents a single point of failure in the overall system. However this master secret can easily be secret-shared across a number of independent authorities [5]. In the simplest case a pair of TAs might each independently generate secrets s_1 and s_2, and issue to a client s_1A and s_2A which can be added by the client to create $sA = s_1A + s_2A$. Note that a Hardware Security Module (HSM) – as commonly used to protect PKI private keys – could be used for one or both of these calculations, to provide an extra level of security. Now the master secret s is not known to any single entity. In this way we can formally satisfy our condition 10.

5 A SOK-based protocol

It is important to observe that no Trusted Authority Public Key is required in this protocol. Combined with the idea of putting clients exclusively in $\mathbb{G}_1$ and servers exclusively in $\mathbb{G}_2$, we overcome both of the problems associated with the attempted PKI-based solution of section 3.

Consider an attacker who steals a token and attempts to log on to the server without knowing the PIN. They can derive the key from $e((s-\alpha)A, S)$, whereas the server will derive it correctly from $e(A, sS)$. If the server were then to send something encrypted with the correct key to the attacker, the attacker can find the PIN via an offline dictionary attack by trying to decrypt with every possible key derived from $e((s-\alpha)A+gA, S)$ until they find the value g which provides a valid decryption. To prevent this we force the client to first submit an authenticator derived from their calculated key. Only when the server is convinced that the client has derived the correct key (using the correct PIN) will the protocol be continued.

We are now ready to present our simple SOK based solution. Assuming the protocol succeeds, both sides proceed to use the key K. See Table 3.

Alice - identity ID_a	**Server - identity** ID_s
$ID_a \rightarrow$	$\leftarrow ID_s$
$S = H_2(ID_s)$, $A = H_1(ID_a)$	$A = H_1(ID_a)$, $S = H_2(ID_s)$
$k = e((s-\alpha)A + \alpha A, S)$	$k = e(A, sS)$
$K = H(k)$	$K = H(k)$
$M = H(ID_a \vert ID_s \vert K)$	$N = H(ID_a \vert ID_s \vert K)$
$M \rightarrow$	if $M \neq N$, drop the connection

Table 3. A Simple SOK-based Protocol

Next we test this simple protocol against our 10 properties. As already indicated we have succeeded in fulfilling properties 2 and 10. We also satisfy properties 1 and 9. Clearly a client can change their PIN locally, by simply adding/subtracting a multiple of A to/from their token, so 4 is satisfied as well. Condition 8 is more than satisfied – as pointed out by Boneh and Franklin [5] the simple form of the client secret (as a multiple of their identity) means that it can be securely split up in a variety of ways, using a secret-sharing scheme.

If the client's reconstituted secret is off by a small amount, a mutual key can still be calculated by the server, who can determine the extent of the error and compensate for it. For example if the client uses $sA + \epsilon A$ as their secret, the server can compensate by using $sS + \epsilon S$ as its secret. Furthermore the server can afford to spend some time searching for the right ϵ. Therefore this scheme

supports a key correction capability as required by our property 6 to support a possible biometric factor. The same behavior can be exploited in other ways. If the client were to enter the wrong PIN, the server can easily determine the extent of the error δ using this key correction capability. So for example if the client entered 1224 instead of 1234, then the server could figure out that the PIN was "out" by 10. This behaviour can also be exploited to intelligently respond to the wrong PIN being entered – if it is out by a lot, do not allow another attempt, if it is out by only one digit, then allow a further attempt, etc. A "coercion" convention could be agreed, for example if the PIN was just out by 1 in the last digit, the server might interpret this as a client entering a PIN while under physical threat, and respond appropriately.

That leaves us with properties 3, 5 and 7 still unsatisfied. Clearly the server can impersonate any client to log in, as they can derive the key for any client C as $e(C, sS)$, which violates 3 and 5. There are multiple other problems as well. The same pair of client and server will always derive the same key k, so we are open to replay attacks. An attacker who captures Alice's token and eavesdrops an encrypted conversation, can easily derive the associated PIN.

The challenge is to block these attacks and satisfy the remaining properties while not losing the properties we have achieved already. The main idea is to replace the SOK construction with another somewhat more elaborate construction.

6 Wang's protocol.

A powerful property of any authenticated key exchange protocol is resistance to a Key Compromise Impersonation (KCI) attack [3]. A Key Compromise Impersonation attack is where an attacker Charlie who captures Alice's credentials (token plus PIN) is in a position not only to log in to a server pretending to be Alice (which is only to be expected), but can also pretend to be the server to the real Alice. Our SOK based protocol is wide open to this kind of attack as while the server can calculate the key as $e(A, sS)$, Charlie who has stolen Alice's credentials can calculate the same value as $e(sA, S)$, without knowing sS. The problem is the ability to exploit bilinearity to move the component involving the master secret s from one side of the pairing to the other. The solution is to "overload" each side of the pairing with other necessary calculations and hence to block this possibility.

Observe that in the client-server setting (rather than the peer-to-peer setting in which key exchange is usually described) KCI can be broken down into server-to-client KCI, where the server can masquerade as a client, or client-to-server KCI where the client can masquerade as a server, or mutual KCI where both cases are possible. The point being that there could be a scenario where one or other is possible, but not both. Here we prefer to block both possibilities.

In [31], Wang suggests an efficient Identity-Based Authenticated Key Agreement protocol. It is described in a peer-to-peer setting, but we will adapt it to a client-server protocol by simply implementing it on a type-3 pairing. The original

protocol has a proof of security based on the DBDH assumption. It provides us with a drop-in replacement for the SOK protocol, while now providing support for our properties 3 and 5. Wang's protocol forms part of the P1363.3 proposed IEEE standard [1].

We start with a much simplified version of Wang's protocol – Table 4.

Alice - identity ID_a	**Server - identity** ID_s
Generates random $x < q$	Generates random $y < q$
$ID_a \rightarrow$	$\leftarrow ID_s$
$S = H_2(ID_s),\ A = H_1(ID_a)$	$A = H_1(ID_a),\ S = H_2(ID_s)$
$P_a = xA$	$P_s = yS$
$P_a \rightarrow$	$\leftarrow P_s$
$k = e(x.((s-\alpha)A + \alpha A), P_s)$	$k = e(P_a, y.sS)$
$K = H(k)$	$K = H(k)$
$M = H(ID_a \vert ID_s \vert P_a \vert P_s \vert K)$	$N = H(ID_a \vert ID_s \vert P_a \vert P_s \vert K)$
$M \rightarrow$	if $M \neq N$, drop the connection

Table 4. Simplified Protocol based on [31]

For both parties the agreed key $k = e(A, S)^{sxy}$. Wang's protocol is not vulnerable to a KCI attack. However the Trusted Authority who has eavesdropped on P_a and P_s can easily calculate the key as $e(P_a, P_s)^s$. This is as a direct result of the protocol not having the property (our number 7) of Perfect Forward Secrecy (PFS).

To include the property of PFS, Wang suggests this modification which includes an in-tandem Diffie-Hellman into the key exchange – Table 5. See also [6].

Both keys are the same $K = H(e(A, S)^{sxy} \vert xwA)$. In Wang's original paper [31] it is suggested to use $w = y$. However this re-enables the Key Compromise Impersonation attack, based on the observation that $e(xsZ, yS) = e(xyZ, sS)$, so if sS is captured, Charlie can log in to S with any identity Z.

7 A Two-Factor Client-Server Protocol

As we have described it above Wang's protocol has a problem. The identities of A (and S) are not directly included in the key calculation. So Bob can claim the identity Alice, but still log on using his own credentials. This is clearly not satisfactory. Wang uses a rather complex method to fix this, as we will see.

Assume $H_q(.)$ is a hash function that hashes to a number in the range 1 to q (although according to Wang its OK to reduce this range to a size half the

Alice - identity ID_a	**Server - identity** ID_s
Generates random $x < q$	Generates random $y, w < q$
$ID_a \rightarrow$	$\leftarrow ID_s$
$S = H_2(ID_s)$, $A = H_1(ID_a)$	$A = H_1(ID_a)$, $S = H_2(ID_s)$
$P_a = xA$	$P_s = yS, P_g = wA$
$P_a \rightarrow$	$\leftarrow P_s, P_g$
$k = e(x.((s-\alpha)A + \alpha A), P_s)$	$k = e(P_a, y.sS)$
$K = H(k\|xP_g)$	$K = H(k\|wP_a)$
$M = H(ID_a\|ID_s\|P_a\|P_s\|P_g\|K)$	$N = H(ID_a\|ID_s\|P_a\|P_s\|P_g\|K)$
$M \rightarrow$	if $M \neq N$, drop the connection

Table 5. Simplified Protocol with Perfect Forward Secrecy

number of bits in q, with some performance gains). Our final scheme is given in Table 6.

Alice - identity ID_a	**Server - identity** ID_s
Generates random $x < q$	Generates random $y, w < q$
$ID_a \rightarrow$	$\leftarrow ID_s$
$S = H_2(ID_s)$, $A = H_1(ID_a)$	$A = H_1(ID_a)$, $S = H_2(ID_s)$
$P_a = xA \quad \rightarrow$	$\leftarrow P_s = yS, P_g = wA$
$\pi_a = H_q(P_a\|P_s\|P_g)$, $\pi_s = H_q(P_s\|P_a\|P_g)$	$\pi_s = H_q(P_s\|P_a\|P_g)$, $\pi_a = H_q(P_a\|P_s\|P_g)$
$k = e((x+\pi_a)((s-\alpha)A + \alpha A), \pi_s S + P_s)$	$k = e(\pi_a A + P_a, (y+\pi_s)sS)$
$K = H(k\|xP_g)$	$K = H(k\|wP_a)$
$M = H(ID_a\|ID_s\|K)$	$N = H(ID_a\|ID_s\|K)$
$M \rightarrow$	if $M \neq N$, drop the connection

Table 6. Two-Factor Authentication Protocol

For both parties observe that $k = e(A, S)^{s(x+\pi_a)(y+\pi_s)}$. Observe that Alice's token and PIN are recombined locally before any value calculated from them is transmitted. If the wrong PIN is entered, the server drops the connection (and only allows a few more attempts before taking more drastic action with respect to the purported "Alice").

8 Revocation

A potential problem with any such system is the provision of a mechanism for the revocation of keys issued by the Trusted Authority. One realisation of the simple idea suggested by Boneh and Franklin [5] is to re-issue private keys to each client and server every month, and to establish the convention that the current month should always be appended to the identity. In this way a revoked user would simply not be issued with a private key for the next calendar month.

An alternative idea was suggested by Boldyreva et al. [4] and made explicit by Tseng and Tsai [27], and that is to publicly issue a "permit" to each client, effectively giving them permission to continue to operate for the next time period.

In the protocol above for the client Alice we instead calculate $A = H_1(ID_a) + H_T(ID_a|T_i)$ where T_i is a textual description of the i-th time period. For the protocol to work correctly Alice must be issued by the Trusted Authority with a permit $s.H_T(ID_a|T_i)$ which gets added to her combined PIN-plus-token secret $s.H_1(ID_a)$ to create sA.

Observe that the permit is of no use to any other party, and hence can be issued publicly, or simply sent to Alice by email. A proof of security for this idea in the context of Boneh and Franklin IBE can be found in [27]. Note however that we cannot deploy time-permits on the server side, as this would leave us open to an MSSR attack (see above).

9 Implementation

There is a wide-spread concern that pairing-based protocols may be intrinsically too slow for real world application. However recent progress in efficient implementation has clearly answered the doubters (for a review of recent progress see [22]).

We implemented the protocol using a BN curve [2] at the standard AES-128 bit level of security, on a 64-bit Intel i5 520M processor, clocked at 2.4GHz, using a mixture of C and some automatically generated assembly language.

As well as using optimal techniques for the pairing itself [22], we fully exploit the Fuentes-Castañeda et al. method for fast hashing to $\mathbb{G}_2$ [9], the well-known GLV method for point multiplication in $\mathbb{G}_1$ [12], and the Galbraith-Scott technique for fast point multiplication in $\mathbb{G}_2$ [11].

Even without any precomputation [22] the above protocol is fully practical. On our test hardware the server side of the calculation required 4.48 milliseconds and the client side required 4.10 milliseconds.

10 Conclusions

Note that Wang's protocol is not the only choice here. There is for example an alternative protocol due to S. Wang et al. [30] which also supports a PIN and key correction. It is rather more elegant than Wang's, and claims to be faster.

It also has a nice and simple proof of security. Another alternative is Protocol 2 from Chen and Kudla [6].

Consider the implications of a successful hack on the server which reveals its secret sS. Note that there are no client related secrets (token, PIN or biometric) stored on the server. However this attack obviously allows a false server to be set up, onto which clients might be convinced to log on. If the successful hacker then captures a token, they can easily find the associated PIN. However they cannot use the captured server secret to log onto the genuine server, and access its data. They cannot enroll any other clients. So we fully achieve our property 5 and although loss of the server secret is serious, the effects are to a maximum extent mitigated.

By removing the role of client registration from the server and entrusting it instead to an independent third party, we have enabled a PKI/SSL-like solution to the problem of two-factor authentication. We maintain that our proposed scheme is a very suitable alternative for any Web-based application that currently uses Username/Password, being simultaneously more user-friendly and more secure.

References

1. IEEE P1363 home page. `http://grouper.ieee.org/groups/1363/`.
2. P.S.L.M. Barreto and M. Naehrig. Pairing-friendly elliptic curves of prime order. In *Selected Areas in Cryptology – SAC 2005*, volume 3897 of *Lecture Notes in Computer Science*, pages 319–331. Springer-Verlag, 2006.
3. S. Blake-Wilson, D. Johnson, and A. Menezes. Key agreement protocols and their security analysis. *Cryptography and Coding*, 1355:30–Ũ45, 1997.
4. A. Boldyreva, V. Goyal, and V. Kumar. Identity-based encryption with efficient revocation. In *Proceedings of the 14th ACM Conference on Computer and Communications Security – CCS 2008*, pages 417–426. ACM Press, 2008.
5. D. Boneh and M. Franklin. Identity-based encryption from the Weil pairing. *SIAM Journal of Computing*, 32(3):586–615, 2003.
6. L. Chen and C. Kudla. Identity based key agreement protocols from pairings. In *Proc. of the 16-th IEEE Computer Security Foundations Workshop*, pages 219–233. IEEE Computer Society, 2003.
7. D. Fiore and R. Gennaro. Making the Diffie-Hellman protocol identity-based. In *Topics in Cryptology - CT-RSA 2010*, volume 5985 of *Lecture Notes in Computer Science*, pages 165–178. Springer, 2010.
8. D. Freeman, M. Scott, and E. Teske. A taxonomy of pairing friendly elliptic curves. *Journal of Cryptography*, 23:224–280, 2010.
9. L. Fuentes-Castaneda, E. Knapp, and R. Rodríguez-Henríquez. Faster hashing to $\mathbb{G}_2$. In *Selected Areas in Cryptography – SAC 2011*, volume 7118 of *Lecture Notes in Computer Science*, pages 412–430. Springer-Verlag, 2011.
10. S. Galbraith, K. Paterson, and N. Smart. Pairings for cryptographers. *Discrete Applied Mathematics*, 156:3113–3121, 2008.
11. S. Galbraith and M. Scott. Exponentiation in pairing-friendly groups using homomorphisms. In *Pairing 2008*, volume 5209 of *Lecture Notes in Computer Science*, pages 211–224. Springer-Verlag, 2008.
12. R.P. Gallant, R.J. Lambert, and S.A. Vanstone. Faster point multiplication on elliptic curves with efficient endomorphisms. In *Advances in Cryptology – Crypto 2001*, volume 2139 of *Lecture Notes in Computer Science*, pages 190–200. Springer-Verlag, 2001.
13. F. Hao and D. Clarke. Security analysis of a multi-factor authenticated key exchange protocol. Cryptology ePrint Archive, Report 2012/039, 2012. `http://eprint.iacr.org/2012/039`.
14. H. S. Kim, S. W. Lee, and K. Y. Yoo. ID-based password authentication scheme using smart cards and fingerprints. *ACM Operating Systems Review*, 37(4):32–41, 2003.
15. I. Liao, C. Lee, and M. Hwang. A password authentication scheme over insecure networks. *Journal of Computer and System Sciences*, 72:727–740, 2006.
16. R. Martinez-Pelaez and F. Rico-Novella. Cryptanalysis of Sood at al.'s authentication scheme using smart cards. Cryptology ePrint Archive, Report 2012/386, 2012. `http://eprint.iacr.org/2012/386`.
17. D. Pointcheval and S. Zimmer. Multi-factor authenticated key exchange. In *ACNS'08 Proceedings of the 6th international conference on Applied cryptography and network security*, pages 277–295. Springer-Verlag, 2008.
18. R. Sakai, K. Ohgishi, and M. Kasahara. Cryptosystems based on pairing. The 2000 Symposium on Cryptography and Information Security, Okinawa, Japan, 2000.

19. C. P. Schnorr. Efficient identification and signatures for smart cards. In *Crypto '89: Advances in Cryptology*, volume 435 of *Lecture Notes in Computer Science*, pages 239–252, 1989.
20. M. Scott. Authenticated ID-based key exchange and remote log-in with simple token and PIN number. Cryptology ePrint Archive, Report 2002/164, 2002. `http://eprint.iacr.org/2002/164`.
21. M. Scott. Cryptanalysis of an ID-based password authentication scheme using smart cards and fingerprints. Cryptology ePrint Archive, Report 2004/017, 2004. `http://eprint.iacr.org/2004/017`.
22. M. Scott. On the efficient implementation of pairing-based protocols. In *Cryptography and Coding 2011*, volume 7089 of *Lecture Notes in Computer Science*, pages 296–308. Springer-Verlag, 2011.
23. A. Shamir. Identity-based cryptosystems and signature schemes. In *Advances in Cryptology: Proceedings of CRYPTO 84*, volume 196 of *Lecture Notes in Computer Science*, pages 47–53, 1984.
24. S. Sood, A. Sarje, and K. Singh. An improvement of Liao at al's authentication scheme using smart cards. *International Journal of Computer Applications*, 1(8):16–23, 2010.
25. D. Stebila, P. Poornaprajna, and S. Chang. Multi-factor password-authenticated key exchange. In *Australasian Information Security Conference, CPRIT volume 105*, pages 56–66. Austalian Computer Society, 2010.
26. C. Tsai, C. Lee, and M. Hwang. Password authentication schemes: Current status and key issues. *International Journal of Network Security*, 3(2):101–115, 2006.
27. Y. Tseng and T. Tsai. Efficient revocable ID-based encryption with a public channel. *The Computer Journal*, 55(4):475–486, 2012.
28. D. Wang and C. Ma. Robust smart card based password authentication scheme against smart card security breach. Cryptology ePrint Archive, Report 2012/439, 2012. `http://eprint.iacr.org/2012/439`.
29. D. Wang, C. Ma, and P. Wu. Secure password-based remote user authentication scheme with non-tamper resistant smart cards. Cryptology ePrint Archive, Report 2012/227, 2012. `http://eprint.iacr.org/2012/227`.
30. Shengbao Wang, Zhenfu Cao, Zhaohui Cheng, and Kim-Kwang Raymond Choo. Perfect forward secure identity-based authenticated key agreement protocol in the escrow mode. *Science in China Series F Information Sciences*, 52(8):1358–1370, 2009.
31. Y. Wang. Efficient identity-based and authenticated key agreement protocol. Cryptology ePrint Archive, Report 2005/108, 2005. `http://eprint.iacr.org/2005/108`.
32. Y. Wang. Password protected smart card and memory stick authentication against off-line dictionary attacks. Cryptology ePrint Archive, Report 2012/120, 2012. `http://eprint.iacr.org/2012/120`.
33. T. Wu. The secure remote password protocol. In *Proceedings of the 1998 Internet Society Network and Distributed System Security Symposium*, pages 97–111, 1998.
34. Guomin Yang, Duncan S. Wong, Huaxiong Wang, and Xiaotie Deng. Formal analysis and systematic construction of two-factor authentication scheme. In *Proceedings of the 8th international conference on Information and Communications Security*, ICICS'06, pages 82–91. Springer-Verlag, 2006.
35. E. Yoon and K. Yoo. New authentication scheme based on a one-way hash function and Diffie-Hellman key exchange. In *CANS'05 Proceedings of the 4th international conference on Cryptology and Network Security*, volume 3810 of *Lecture Notes in Computer Science*, pages 147–160. Springer-Verlag, 2005.