# M-Pin Full Technology (Version 3.0)

Michael Scott

Chief Cryptographer
Certivox Labs
mike.scott@certivox.com

**Abstract.** M-Pin is a two-factor authentication protocol which has been proposed as an alternative to Username/Password, which works in conjunction with SSL/TLS. Here we derive a more complex MPin derivative called M-Pin-Full which also supplants the functionality of SSL/TLS.

## 1 Introduction

M-Pin is a zero-knowledge authentication protocol which authenticates a client to a server. Its unique feature is that it allows a short PIN number to be extracted from the client secret to create a token+PIN combination, facilitating two factor authentication. The idea can easily be extended to support multifactor authentication.

A strong client-server protocol should (a) authenticate the client to the server, (b) authenticate the server to the client, and (c) should result in a negotiated encryption key with which subsequent communications can be encrypted. The standard method of implementation uses a Username/Password mechanism to authenticate the client to the server, and the well known TLS/SSL protocol to authenticate the server to the client and to establish the encryption key. The weakest link here is the Username/Password mechanism which is widely regarded as being broken. SSL itself, to a lesser extent, has been weakened by intensive scrutiny which has revealed some exploitable vulnerabilities.

To replace Username/Password, multi-factor authentication is the most often touted solution. Of all the possible form-factors the simple ATM-like combination of a token and a PIN number is the most user-familiar and user-friendly.

Here we extend the M-Pin technology solution to also replace the SSL functionality. Recall that M-Pin makes use of a Trusted Authority to issue client and server secrets. Using M-Pin, no client secrets, or values derived from client secrets, are stored on the server. The reader is encouraged to read the M-Pin paper before continuing with this white paper.

## 2 M-Pin

Here we recall the original M-Pin protocol. Alice is proving to a server that she is in possession of a valid secret, while revealing nothing about it, using a Zero-Knowledge Proof protocol.

A Trusted Authority (TA) possesses a unique secret value $s$ associated with its support for a particular server. That server is issued with a secret $sQ$, which represents a fixed point $Q$ on a special elliptic curve multiplied by the TA secret $s$. Alice, whose identity string is $ID_a$, has this identity hashed and mapped to a point $A$ on the same curve (albeit a different group of the same order $q$), and is issued with the secret $sA$. Alice chooses a PIN number $\alpha$, and extracts this from her secret to create her token $(s-\alpha)A$. The protocol then proceeds as follows.

| **Alice - identity** $ID_a$ | **Server** |
|---|---|
| Generates random $0 < x < q$ | Generates random $0 < y < q$ |
| $A = H(ID_a)$ | |
| $U = xA$ | |
| $ID_a$, $U \quad \rightarrow$ | |
| | $\leftarrow y$ |
| $V = -(x+y)((s-\alpha)A + \alpha A) \rightarrow$ | |
| | $A = H(ID_a)$ |
| | $g = e(V, Q).e(U + yA, sQ)$ |
| | if $g \neq 1$, reject the connection |

**Table 1.** M-Pin

This all works thanks to the pairing function $e(.,.)$ and its remarkable bilinearity property $e(aP, Q) = e(P, aQ) = e(P, Q)^{ab}$.

## 3 M-Pin-Full

This more elaborate protocol not only replaces Username/Password, but replaces the functionality of SSL as well. Our starting point is the M-Pin protocol as described above. The idea is to run it first (to authenticate the client to the server), and then proceed to authenticate the server to the client via an authenticated key exchange, which also establishes the agreed key.

The first thing to note is that both the client and the server can already calculate a mutual authenticated encryption key! The client Alice can calculate it as $e(sA, Q)$, and the server as $e(A, sQ)$. Note that for a client this is a fixed value that can be precomputed. However first we extract the PIN and actually pre-calculate $g_1 = e((s-\alpha)A, Q)$ and $g_2 = e(A, Q)$, and store these on the client along with the token. The full secret can then be reconstructed when the PIN is available as $g_1.{g_2}^{\alpha}$, which only requires a small amount of work as $\alpha$ is small.

However we must be careful to (a) protect the PIN from an active or passive attacker who has perhaps captured the token, (b) prevent a Key Compromise Impersonation (KCI) attack, and (c) achieve the property of Perfect Forward Secrecy (PFS). To support the property of PFS, the standard approach adopted here is to introduce a Diffie-Hellman component into the protocol.

This protocol requires another general hash function $H_g(.)$ which serializes, and hashes its input to a 256-bit value. Both sides can then extract an AES key from this value $K$.

It is left as a simple exercise for the reader to confirm that both client and server end up with the same key. Note that since the first part of the protocol is just the original M-Pin protocol, all of its features and extensions still apply. In particular Time Permits can be used as a revocation mechanism.

| **Alice - identity** $ID_a$ | **Server** |
|---|---|
| Generates random $0 < x, r < q$ | Generates random $0 < y, w < q$ |
| $A = H(ID_a)$ | |
| $U = xA$ | |
| $ID_a, U \quad \rightarrow$ | |
| | $\leftarrow y$ |
| $V = -(x + y)((s - \alpha)A + \alpha A) \rightarrow$ | |
| | $A = H(ID_a)$ |
| | $g = e(V, Q).e(U + yA, sQ)$ |
| | if $g \neq 1$, reject the connection |
| $R = rA \rightarrow$ | $\leftarrow W = wA$ |
| $K = H_g((g_1.{g_2}^{\alpha})^r \vert xW)$ | $K = H_g(e(R, sQ) \vert wU)$ |

**Table 2.** M-Pin-Full

Note that the transmission of $R$ from the client to the server can be done at the same time as $V$ is transmitted, and the transmission of $W$ from the server to the client can be done at the same time as $y$ is transmitted, to avoid introducing any extra flows into the protocol.

## 4 Security

Our main concern is with an attacker who has obtained a client token and is in a position to launch an active attack on the client's attempted authentication in order to determine their PIN.

For example if a client were simply to go ahead and start encrypting using the shared key $e(sA, Q)$, then an attacker who knew the token $(s - \alpha)A$ could exhaustively try adding to the token every possible multiple of $A$ to create $X$ until they hit on the right PIN, in which case $X = sA$ and the key $e(X, Q)$ would decrypt the ciphertext to something sensible. To prevent this we actually use as the key $e(rsA, Q)$, and now an attackers knowledge of the token cannot be used to guess the key without knowing $r$.

A more subtle attack is also possible. An attacker who has captured Alice's credentials can pretend to be a valid server to Alice by simply ignoring the initial M-Pin protocol and then also calculating the mutual key as $e(sA, Q)$ (rather than as $e(A, sQ)$ as a valid server would). However the presence of $r$ in the calculation of the key also prevents this Key Compromise Impersonation attack.

Another type of KCI attack would arise if an attacker who had captured the server secret $sQ$ were able to use it to authenticate as a valid client. Fortunately this is not possible, as $sQ$ is in the wrong group.

An active attacker might allow Alice to complete the first part of the protocol and then attempt to hijack the link before the calculation of the key. But observe how the value of $x$ is re-used for the calculation of the Diffie-Hellman component of the key. This binds both parts of the protocol together and effectively blocks any hijacking attempt.

## 5 Discussion

It should be pointed out that M-Pin-Full is not entirely equivalent to the SSL+M-Pin combination. The client identity is transmitted in the clear in M-Pin-Full, whereas with SSL the entire M-Pin protocol runs under cover of SSL, which therefore provides an anonymity feature. Of course it is always possible to run the M-Pin-Full protocol in conjunction with SSL.

One important advantage compared to the SSL+M-Pin combination is that any so-called phishing attack will be ineffective against this protocol, as the phishing website will not be able to establish the mutual key $K$.