Key-Escrow-less M-Pin

Michael Scott

MIRACL Labs
mike.scott@miracl.com

Abstract. We have been tasked to harden the M-Pin protocol against a "key-escrow" attacker, who has the authority to demand and be issued with all of the secrets from all of the distributed trust authorities (D-TAs) and the M-Pin server, and use them to try to create valid credentials in the identities of valid clients in order to impersonate them and gain access to their accounts on a remote server via the normal M-Pin authentication process. As a purely identity-based protocol M-Pin is open to this kind of attack. Our recommended response is to use ideas from so-called Certificateless cryptography, which is a standard and established response to the key-escrow property of pure identity-based schemes.

1 Introduction

The reader should refer to the original M-Pin literature for definitions. M-Pin is a multi-factor client-server authentication protocol, using identity based cryptography.

Many in the cryptographic community have a visceral and negative response to any cryptographic protocol that admits to the key-escrow property, whereby a single component of the system may recreate the keys of any or all users. That component may subsequently be coerced by law enforcement or other agencies to co-operate and give up those keys. For many people only a system whereby each user generates their own secret and does not ever expose it to any external third party, is acceptable.

In M-Pin, as in other identity based systems, the single component at risk is the TA (Trusted Authority). Our solution to the key-escrow issue has been to distribute this functionality to at least three D-TAs (Distributed Trust Authorities). The intention is that the three D-TAs are under separate organisational and jurisdictional control.

From a practical point of view this is probably a completely satisfactory solution. Neverthess for some critics it is not enough. Understandable, since a TA compromise, like a PKI root key compromise, is particularly insidious as it applies no matter how careful clients and servers are about their personal security. The purpose of this note is to suggest ways in which M-Pin might be made key-escrow-immune.

The standard way to do this is to become no longer purely identity based. The client now has, as well as their identity, a public/private key pair. But fortunately using the "certificateless cryptography" paradigm [1], as suggested

by the name, this can be done without having to resort to a PKI (Public Key Infrastructure).

2 The Patch

Original M-Pin looks like this. Recall that Q is a fixed point in the group $G2$, s is the combined master secret of the D-TAs, $(s-\alpha)A$ is the client "token" and α is their PIN.

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

Our suggested patch to fix the key-escrow issue is shown in Table 2. In line with the certificateless approach, there will be a single private key (as before), for each client an identity hashed, say for Alice, to a point A (as before), and a new public key associated with Alice P_A. As before the client accepts from the D-TAs the client secret shares and combines them to create sA (now called a partial client secret), but now it multiplies it by a randomly generated value z to get the new full client secret zsA. The server behaves in a similar fashion, combining its secret shares to create sQ, and again multiplying it by a randomly generated value w to get the full server secret wsQ. It also calculates a public key wQ.

The full client secret can be split up into multiple factors exactly as before. The client then accesses the server public key and calculates its own new public key as $P_A = z^{-1}wQ$ and conveys this to the server for safe keeping. Now only a client who can reconstruct from its multiple factors the value zsA, can authenticate to a server who is in possession of the associated public key P_A and the full server secret wsQ. After creating the full private key and the public key the client should delete z, and similarly the server should delete w.

Now Al Riyami and Paterson [1] warn us that "Of course, we must assume that the (D-TAs) do not mount an attack of this type: armed with the partial private key and the ability to replace public keys, the (D-TAs) could impersonate any entity in generating a private/public key pair and then making the public key available." Our sole new security concern then is to ensure that the new public key is beyond the reach of those in control of the D-TAs, meaning that they cannot substitute this public key with another, with a view to impersonating a client by generating a new set of keys for the impersonator, and substituting the impersonator's public key for that of the original client.

Alice - identity ID_a	**Server**
Generates random $0 < x < q$	Generates random $0 < y < q$
$A = H(ID_a)$	
$U = xA$	
$ID_a, U \quad \rightarrow$	
	$\leftarrow y$
$V = -(x+y)((sz-\alpha)A + \alpha A) \rightarrow$	Retrieve Client Public Key P_A
	$A = H(ID_a)$
	$g = e(V, P_A).e(U + yA, wsQ)$
	if $g \neq 1$, reject the connection

Table 2. Key-Escrow-less M-Pin

The proposed patch does not affect the D-TAs in any way, or the client enrollment with the D-TAs. It requires a minimal modification to the client and server sides of the protocol, and comes at little extra computational cost. Since the D-TAs do not know z or w, they cannot be coerced into revealing the full client private key zsA or the full server secret wsQ. There are no extra secrets to be protected on either the client or server side. The use of Time Permits complicates things, but with a bit more work on the server side they can be incorporated into this new regime.

We emphasise that the public key P_A as created by Alice at the time of her enrollment must be used by the server in all subsequent authentications by Alice.

However, as always in cryptography, its not so simple! This new client public key must be handled correctly, and must be available to the server in its original form whenever that client attempts to authenticate. The standard way would be for the public keys to be placed in a public directory. Or the server might maintain a FIDO-like list of identities and their associated public keys. Recall that we must avoid a public-key-substitution attack, where an impersonator

generates their own private key and gets the server to accept the associated public key in place of the original.

3 Server Enrollment

It would appear that the client, following its enrollment via the D-TAs, now needs to complete its enrollment directly with the server. Assume that the server has already generated a random w, constructed its full secret wsQ and its public key wQ, and deleted w. Here we sketch a simple enrollment process.

1. The client authenticates to the server using regular M-Pin, but without using its multi-factor feature, or requiring a time permit.
2. The server checks that an account in this client identity does not already exist. If there is an existing account the server drops the link.
3. The server creates a new database entry and initialises a new account for the client, and sends the client its public key wQ.
4. The client generates random z and calculates $z^{-1}wQ$ and returns this (its public key) to the server.
5. The server stores the client public key in the newly created client account database, and drops the link.
6. The client generates their full private key zsA, deletes z, chooses their PIN number, and splits the full private key into token and PIN.

On subsequent authentications the server retrieves the client public key from the client database after the first step in the protocol as shown in Table 2. It is assumed that a Key-Escrow attacker, for all of its powers, does not have the ability to directly access the client database (otherwise they could completely bypass the authentication process, avoiding the need to extract the secret keys of any entity).

In fact the all-powerful Key-Escrow attacker who demands and is given all of the secrets of all parties (that is all of the D-TA and server secrets, and read access to client public keys), lacking only the secret possessed by a targeted client, is still unable to authenticate via the normal process in the identity of that client.

However unsurprisingly such an attacker can do a lot of damage: They can for example set up a false server, launch phishing attacks, and reduce the client protection by one low-entropy factor. For example in the standard two-factor setting if they capture a client token they can try every PIN against the false server until the right one is found. The same known issue arises with standard M-Pin if the server secret is lost.

4 Time Permits

As mentioned above using the method described causes a problem with time permits. To continue using them as before, they would have to be multiplied

by z by the client before use to ensure the mathematics still worked correctly. However for security reasons the client has already deleted z. The answer is for the client to forward their time permit sT directly to the server, and not to add it into the value V as is currently done. Recall that since Time Permits are broadcast by the D-TAs and do not need any cryptographic protection, they could in theory already be picked up directly by the server. However we suggest that the Time Permit should continue to be picked up by the client. and simply forwarded to the server as part of the authentication process. The verification equation for the server then becomes the slightly more elaborate test that $e(V, P_A).e(sT, wQ).e(U + y(A + T), wsQ) = 1$. This requires a triple pairing calculation, which will cost just a little more (I would guess 20%) than the current double pairing calculation.

5 Discussion

It is worth briefly revisiting the necessity for all this. In a scenario where an external agency can seize or otherwise force the cooperation of all of the D-TAs, it is hard to imagine a scenario where these counter-measures would practically improve security. For example why should such an agency not just seize the client database directly from behind the server and access client data directly, completely bypassing the authentication mechanism? An objection might be that while the D-TAs are all in a jurisdiction under the control of the external agency, the server may be in a different jurisdiction. But we already defend against this eventuality by ensuring that one D-TA is under the control of the same entity that controls the server, and which is presumably in the same jurisidiction as the server. Indeed we could place a D-TA right next to the server, even running as a seperate thread on another core of the same processor!

Observe that the attack model we are defending against may not be very realistic. It assumes that the server hands over its secrets but does not co-operate beyond that. In particular the client database cannot be accessed directly by the attacker - they are allowed only to go through a normal authentication process while attempting to impersonate a client. In our opinion that is a little artificial. As an aside, if I were an owner of an M-Pin server, and if approached by a Key-Escrow authority and asked for my keys, I would rather cut a deal before compromising all of my clients: I would volunteer to put in a patch that allowed access to certain named accounts from certain named IP addresses that completely bypassed the authentication check.

One other issue worth mentioning. Until now an M-Pin signature has not had the property of being non-repudiatable. A client could always claim in a court of law that their supposed signature was forged by a conspiracy of the D-TAs. However using the method described above this would no longer be the case, and such signatures would be non-repudiatable, just like PKI signatures. Of course this would only be true in a narrow cryptographic sense, it is a matter of law to determine the validity or otherwise of a particular type of digital signature.

The other downside of M-Pin signatures, that only the server in possession of the full server secret can verify them, remains.

6 Conclusion

Note that this proposal applies only to M-Pin and its variants.

References

1. S. Al-Riyami and K. Paterson. Certificateless public key cryptography. Cryptology ePrint Archive, Report 2003/126, 2003. `http://eprint.iacr.org/2003/126`.