SOK it to the IoT

Michael Scott

Chief Cryptographer
MIRACL Labs
Not for Public Distribution
`mike.scott@miracl.com`

Abstract. SOK is the only practical non-interactive authenticated key exchange protocol known to cryptography. It is identity-based – in fact it must be since if interaction is not allowed, then only the identity of the correspondent is known, and clearly this needs to be known in order to initiate a contact. Here we discuss SOK and its application to the Internet of Things, and how it can leverage the MIRACL cloud based infrastructure.

1 Introduction to SOK

First a short description of SOK. Consider Alice and Bob. They are each issued with secrets s.Alice and s.Bob where s is a master secret belonging to a Trusted Authority (TA), who issues these secrets to all those entitled to them. These secrets are such that knowing s.Alice, it is impossible to determine s.

When Alice makes contact with Bob she calculates the encryption key $e(s.\text{Alice},\text{Bob})$. Bob calculates the same key as $e(s.\text{Bob},\text{Alice})$. By the magic of the function $e(.)$ (called a pairing) and its bilinearity property, these keys are the same, and unique to the connection between Alice and Bob. They can then use this key to encrypt their communication, each assured of the identity of the other. No other third party can calculate this key, except for the Trusted Authority. The MIRACL infrastructure solves this problem by distributing the Trusted Authority (DTA).

To apply this to an IoT, secrets are issued from the DTAs to each device, which is typically identified by its serial number.

2 Introduction to IoT

An IoT consists of a lot of Things of differing capability and functionality which nevertheless need to communicate. The Things may be static or mobile. They almost certainly communicate by wireless. The inter-Thing communication may be either peer-to-peer, or client-to-server. The networking topology may be fixed or fluid.

Each Thing needs a safe place to store its unique identifying secret. Each Thing also needs to be able to compute any required calculations in a reasonable amount of time.

3 Applying SOK to the IoT

There are some practical problems which need to be solved. If SOK is to work as described above, the pairing needs to have the symmetric property e(Alice,Bob)=e(Bob,Alice). What is known as a type-1 pairing has this property but a a type-3 pairing does not. However type-3 pairings are much faster to calculate. We would prefer to use a type-3 pairing.

To get this to work as before in a peer-to-peer setting each correspondent can be issued with two secrets, a left-hand secret and a right-hand secret. By convention a caller uses their left-hand secret and a callee uses their right-hand secret. So Alice calculates e(s.Alice-left,Bob-right) and Bob calculates e(Alice-left,s.Bob-right), and since bilinearity still holds, these will be the same.

However we can make this problem into a feature which supports the client-server setting. Now some Things are issued with both secrets, some are issued with only left-hand secrets, and some are issued only with right-hand secrets, depending on their role in the overall IoT. This brings security advantages. For example a Thing with only client capabilities cannot respond to a connection request from another Thing.

The next problem is that even a type-3 pairing is quite a complex calculation. On a Raspberry pi (Version 1, 700MHz) it takes 86ms to calculate. So we must assume that either the Thing's processor has the necessary whooomph to handle this, or that the hardware that stores the secret also has dedicated hardware capable of calculating the pairing (Kealan knows a thing or two about this), or that the Thing can safely off-load the pairing calculation to a more powerful, but not necessarily trusted, entity (we are aware of a way of doing this – the DCU patent applies). See below for an alternate approach.

4 The MIRACL infrastructure

The existing MIRACL cloud-based infrastructure already fulfils the role of a Distributed Trusted Authority, suitable for use as described above. Its powerful revocation capability based on time-permits also applies here. Of course Things don't have PINs so M-Pin or indeed any multi-factor technology is not relevant for Thing-to-Thing communication.

However an IoT has another often overlooked component. It makes no sense for an IoT to exist which does not need at some point to interact with an authenticated real person. We would suggest that real people authenticate to the IoT using our M-Pin technology, which also relies on pairing-based cryptography, and shares the same MIRACL infrastructure.

The software to implement SOK on Things is already fully developed in our AMCL library.

5 Bootstrap into a PAKE

A well known (to cryptographers) cryptographic primitive is the PAKE, or Password Authenticated Key Exchange, like SPAKE2. This is typically based on a simple Diffie-Hellman scheme, and basically it allows the use of a short password or PIN to take care of authentication. Recall that Diffie-Hellman is a near perfect solution to the problem of forward-secure key exchange, its only problem being that it is not authenticated and therefore wide open to a Man-In-The-Middle (MITM) attack. Less well known is that this shortcoming can be fixed using a PAKE if both parties share even a low entropy 4-digit PIN.

Which suggests the following hybrid scheme. When two entities first make contact they share a key by running Diffie-Hellman under the cover of SOK. During this communication they agree a 4-digit PIN to be used with their next connection. These short PINs are cached in secure memory. The next time they meet they use the PAKE scheme, without SOK, and again agree a new PIN for the next time, etc. This technique is sometimes known as a "Diffie-Hellman rachet", whereby the authentication established for one session can be carried forward to the next.

Using this idea the great majority of connnections use the PAKE approach, all key exchanges are now forward secure, and the SOK mechanism is rarely used other than as a bootstrapping mechanism to get over the "first contact" problem. The PAKE protocol takes only a few milliseconds on the Raspberry pi.

A better idea might be to purge the cache at regular intervals and force a fall-back to SOK, so that the time-permit mechanism can take effect. This might be necessary anyway if a pair of entities using the rachet idea were to somehow get out of synchronism.