Migrating SOK to a type-3 Pairing

Michael Scott

MIRACL Labs
mike.scott@miracl.com
September 16, 2015

Abstract. The non-interactive authenticated key exchange protocol known as SOK after its inventors Sakai, Oghishi and Kasahara, is one of the original pairing-based protocols. Like many such early protocols it was designed to work with a symmetric pairing. However now it is known that symmetric pairings are inefficient. So the issue arises of how to migrate it successfully to the setting of an efficient asymmetric pairing. In this short research note we consider the challenges and opportunities.

1 Introduction

The SOK protocol [7] proposes the only known practical method for non-interactive authenticated key exchange. As originally described it is based on a type-1 pairing [4] on a supersingular elliptic curve. A type-1 pairing operates as $\mathbb{G}_1 \times \mathbb{G}_1 \rightarrow \mathbb{G}_T$, where $\mathbb{G}_1$ is a group of points of prime order q on the curve, and $\mathbb{G}_T$ is a finite extension field of the same order, whose extension is the so-called embedding degree k associated with the curve. The SOK inventors were one of the first to realise that by carefully matching the field size with the embedding degree, that the discrete logarithm problem could remain hard in both $\mathbb{G}_1$ and $\mathbb{G}_T$, and hence that the pairing was a suitable vehicle for cryptography.

A type-1 pairing has the property of symmetry, and it turns out that this property is quite important to the SOK protocol as originally described. However time has not been kind to type-1 pairings over the intervening years. For required levels of security either $\mathbb{G}_1$ or $\mathbb{G}_T$ must be greater than strictly necessary due to the restricted choice of embedding degree possible on supersingular curves, leading to inefficiencies. And for some of the most promising families of supersingular curves, it turns out that the discrete logarithm problem in $\mathbb{G}_T$ is much easier than originally expected [5].

The most efficient pairing is the asymmetric type-3 pairing, which works with non-supersingular pairing-friendly curves. These operate as $\mathbb{G}_1 \times \mathbb{G}_2 \rightarrow \mathbb{G}_T$, where $\mathbb{G}_2$ is a particular group of points, again of the order q, but on a twisted elliptic curve defined over an extension which is a divisor of k. These curves can be constructed to be a near perfect fit at any required level of security [3]. For example the BN curves [1], with an embedding degree of $k = 12$ are an exact fit for the AES-128 equivalent level of security. But the loss of symmetry causes a problem for the SOK protocol.

Pairings are usually written as functions of the form $g = e(A, B)$,where $A \in \mathbb{G}_1$, $g \in \mathbb{G}_T$, and for a type-1 pairing $B \in \mathbb{G}_1$ and for type-3 $B \in \mathbb{G}_2$. There are various ways in which a pairing may be calculated, but that is not an issue here. It is assumed that there is no problem with hashing an arbitrary string to an element in $\mathbb{G}_1$ or $\mathbb{G}_2$, but the details are omitted.

We should also mention the type-2 pairing, which allows a pairing between a pair of elements in $\mathbb{G}_2$. But to preserve the symmetry property for the SOK protocol it would have to be possible to hash arbitrary strings into the same group of order q in $\mathbb{G}_2$, and there is no known way to do this [4].

2 The SOK protocol

The SOK protocol showcases the bilinearity property of the pairing

$$e(xA, B) = e(A, xB) = e(A, B)^x$$

For a type-1 pairing there is also the property of symmetry

$$e(A, B) = e(B, A)$$

We first assume a type-1 pairing is being used. A Trusted Authority generates a master secret s. Alice and Bob seperately visit the trusted authority, and present their identities and prove their right to those identities in some way. Alice is issued with her secret $s.A$ where $A = H_1(\text{“Alice”})$, and the hash function $H_1(.)$ hashes the identity string to a point of order q in $\mathbb{G}_1$. The trusted authority calculation is simply the well known operation of multiplication by the scalar s of a point on an elliptic curve. Similarily Bob is issued with the secret $s.B$ where $B = H_1(\text{“Bob”})$.

Now Alice and Bob can communicate using a shared key, calculated by Alice as $e(sA, B)$ and by Bob as $e(sB, A)$. These keys will be the same due to bilinearity and symmetry. Note that by convention each can put their own secret as the left-hand argument of the pairing, and the hashed identity of the other as the right-hand argument.

2.1 An application

Consider now an application of this protocol to an imagined Internet of Things (IoT) setting [6]. Each Thing is issued with a serial number and its own SOK secret based on that serial number as an identity. These SOK secrets may be embedded at the time of manufacture, by the manufacturer acting as a naturally trusted authority.

When a Thing needs to communicate with another Thing, an action which requires knowing only the identity of the other, both parties can activate SOK to calculate the same key to encrypt their communication.

However in reality this description is probably a massive simplification of a real world IoT deployment. Are the Things capable of protecting their secrets

from an attacker, do they support secure storage? Are they capable of calculating a pairing? Do they communicate on a peer-to-peer basis, or client-server? Are the things mobile or stationary, what is the network topology of their communication links, and are they fixed or fluid? Are all Things created equal or do some have more resources than others? Since we do not have a particular application in mind, we merely suggest that SOK might be a nice fit for at least some of these scenarios.

One question we can help to answer - a pairing on a BN curve at the AES-128 level of security on a Raspberry Pi computer (Version 1, 700MHz), which is often touted as an IoT platform, can be calculated using our own software in just 86ms. While it has to accepted that on a very low power device any pairing-based solution may be impracticable, it may be posssible to offset poor performance by caching keys and/or by using the SOK mechanism to bootstrap into something more efficient.

3 Migrating to a type-3 pairing

Take away the symmetry property, and things get a bit more complicated. One thing we can exploit – in any communication context there is an initiator and a responder. Therefore the obvious solution is to issue each entity with two secrets, one in $\mathbb{G}_1$ and the other in $\mathbb{G}_2$, as proposed by Dupont and Enge [2]. So Alice is issued with sA_1 and sA_2, where $A_1 = H_1(\text{“Alice”})$ and $A_2 = H_2(\text{“Alice”})$. We call these Alice's lefthand and righthand secrets respectively, as this describes where they can appear in the pairing. Similarly Bob is issued with sB_1 and sB_2. Now if Alice initiates and Bob responds, Alice calculates the key as $e(sA_1, B_2)$ and Bob can calculate the same key as $e(A_1, sB_2)$, where by convention the initiator uses their lefthand secret and the responder uses their righthand secret.

That seems an appropriate and workable solution. However maybe we can do better. Consider again the IoT setting. Now Things are divided into two categories, Talkers and Listeners. Some Things might have only one of these attributes, some may have both. But now this division of capabilities can be cryptographically enforced, by issuing lefthand secrets only to talkers and righthand secrets only to listeners. Perhaps a listener-only Thing might be lower powered, and perhaps its secret does not need to be so vigorously defended, as a hacked listener secret may be of less significance.

At first glance it may appear that a listener secret could still be used to talk by exploiting bilinearity – if we cannot calculate $e(sA_1, B_2)$ because we do not possess sA_1, we could instead calculate $e(B_1, sA_2)$. But these are not the same as $A_1 \neq A_2$ and $B_1 \neq B_2$.

Without making any dogmatic claims, we suggest that this attribute of SOK on a type-3 pairing may in fact be considered as a useful feature in many IoT contexts.

References

1. P.S.L.M. Barreto and M. Naehrig. Pairing-friendly elliptic curves of prime order. In *Selected Areas in Cryptography – SAC'2005*, volume 3897 of *Lecture Notes in Computer Science*, pages 319–331. Springer-Verlag, 2006.
2. R. Dupont and A. Enge. Practical non-interactive key distribution based on pairings. Cryptology ePrint Archive, Report 2002/136, 2002. `http://eprint.iacr.org/2002/136`.
3. D. Freeman, M. Scott, and E. Teske. A taxonomy of pairing-friendly elliptic curves. *Journal of Cryptography*, 23:224 – 280, 2010.
4. S. Galbraith, K. Paterson, and N. Smart. Pairings for cryptographers. *Discrete Applied Mathematics*, 156:3113–3121, 2008.
5. R. Granger, T. Kleinjung, and J. Zumbragel. Breaking 128-bit secure supersingular binary curves. In *Advances in Cryptology – Crypto 2014*, volume 8617 of *Lecture Notes in Computer Science*, pages 126–145. Springer-Verlag, 2014.
6. L.B. Oliveira, D.F. Aranha, C.P.L. Gouvea, M. Scott, D.F. Camara, J. Lopez, and R. Dahab. TinyPBC: Pairings for authenticated identity-based non-interactive key distribution in sensor networks. *Computer Communications*, 34:485–493, 2011.
7. R. Sakai, K. Ohgishi, and M. Kasahara. Cryptosystems based on pairing. The 2000 Symposium on Cryptography and Information Security, Okinawa, Japan, 2000.