Applications of Multiple Trust Authorities in Pairing Based Cryptosystems

Liqun Chen, Keith Harrison, David Soldera, Nigel Smart[1]
Trusted Systems Laboratory
HP Laboratories Bristol
HPL-2003-17
February 6th , 2003*

E-mail: {liqun_chen, keith_harrison, david_soldera}@hpl.hp.com, nigel@cs.bris.ac.uk

multiple trusted authorities, multiple identifiers, identity based cryptography, pairings

We investigate a number of issues related to the use of multiple trust authorities and multiple identities in the type of identifier based cryptography enabled by the Weil and Tate pairings. An example of such a system is the Boneh and Franklin encryption scheme. We present various applications of multiple trust authorities. In particular we focus on how one can equate a trust authority with a way to add contextual information to an identity.

* Internal Accession Date Only Approved for External Publication
[1] Computer Science Dept., Woodland Road, University of Bristol, Bristol, BS8 1UB, U.K.

Applications of Multiple Trust Authorities in Pairing Based Cryptosystems

L. Chen[1], K. Harrison[1], D. Soldera[1], and N.P. Smart[2]

[1] Hewlett-Packard Laboratories,
Filton Road,
Stoke Gifford,
Bristol, BS34 8QZ,
United Kingdom.
{liqun_chen, keith_harrison, david_soldera}@hpl.hp.com

[2] Computer Science Department,
Woodland Road,
University of Bristol,
Bristol, BS8 1UB,
United Kingdom.
nigel@cs.bris.ac.uk

Abstract. We investigate a number of issues related to the use of multiple trust authorities and multiple identities in the type of identifier based cryptography enabled by the Weil and Tate pairings. An example of such a system is the Boneh and Franklin encryption scheme. We present various applications of multiple trust authorities. In particular we focus on how one can equate a trust authority with a way to add contextual information to an identity.

1 Introduction

In 1984 Shamir [11] introduced the concept of identity based cryptography and proposed a signature scheme based on the RSA assumption. Over the years a number of researchers tried to propose secure and efficient identity based encryption algorithms, but with little success. This state of affairs changed in 2001 when two identity based encryption algorithms were proposed, one by Cocks [6] based on the quadratic residuosity assumption and another by Boneh and Franklin based on the Weil pairing, although using a variant based on the Tate pairing is more efficient. A number of identity based signature algorithms based on the Tate pairing also exist, e.g. [5], [9] and [10].

In this paper we investigate the applications of the Boneh and Franklin scheme in more detail. In particular we examine how the key structure can lead to interesting applications. Due to our later applications we prefer to talk about *identifier* based cryptography rather than *identity* based cryptography. This is because the word identity seems to imply the name of someone (such as their email address), whilst the schemes actually associate keys with strings.

Each string being able to represent anything, such as legal terms and conditions, and not just a persons identity.

We shall use the additive property of keys of the pairing based systems to define a way of expressing groupings in a clear and concise manner. We then use this to define policies which are then enforced via the means of possession of certain keys. This results in a kind of "key calculus".

The "key calculus" we present is different from, but inspired by, the ideas of Desmedt [7] and Boyd [3] [4] who presented the notion of group oriented cryptography. Usually, one sees group oriented cryptography in the context of group signatures, we shall be concerned in this paper with applications to encryption. Much of our discussion will, however, also apply to identifier based signature schemes as well.

The paper is organised as follows, first we recap on the the Boneh-Franklin encryption scheme. Then we go through a number of applications which are enabled due to the additive property of the keys in the scheme. Finally we give the semantics of the resulting key calculus which will describe precisely what sets of keys can be singled out in any given application.

2 Notation and Pairing Based Schemes

2.1 The Tate Pairing

Let G_1 and G_2 denote two groups of prime order q in which the discrete logarithm problem is believed to be hard and for which there exists a computable bilinear map

$$t : G_1 \times G_1 \longrightarrow G_2.$$

We shall write G_1 with an additive notation and G_2 with a multiplicative notation, since in real life G_1 will be the group of points on an elliptic curve and G_2 will denote a subgroup of the multiplicative group of a finite field.

Since the mapping is bilinear, we can move exponents/multipliers around at will. For example if $a, b, c \in \mathrm{F}_q$ and $P, Q \in G_1$ then we have

$$\begin{aligned} t(aP, bQ)^c &= t(aP, cQ)^b = t(bP, cQ)^a = t(bP, aQ)^c = t(cP, aQ)^b = t(cP, bQ)^a \\ &= t(abP, Q)^c = t(abP, cQ) = t(P, abQ)^c = t(cP, abQ) \\ &= \ldots \\ &= t(abcP, Q) = t(P, abcQ) = t(P, Q)^{abc} \end{aligned}$$

These tricks will be used repeatedly throughout this document.

We define the following cryptographic hash functions

$$\begin{aligned} H_1 &: \{0,1\}^* \longrightarrow G_1, \\ H_2 &: G_2 \longrightarrow \{0,1\}^*. \end{aligned}$$

2.2 Types of Public/Private Key Pairs

We require the following two types of keys:

- A standard public/private key pair is a pair (R, s) where $R \in G_1$ and $s \in \mathrm{F}_q$ with
$$R = sP$$
for some given fixed point $P \in G_1$.
- An identifier based key pair is a pair $(Q_{\mathtt{ID}}, S_{\mathtt{ID}})$ where $Q_{\mathtt{ID}}, S_{\mathtt{ID}} \in G_1$ and there is some trust authority (TA) with a standard public/private key pair given by $(R_{\mathtt{TA}}, s)$, such that the key pair of the trust authority and the key pair of the identifier are linked via
$$S_{\mathtt{ID}} = sQ_{\mathtt{ID}} \text{ and } Q_{\mathtt{ID}} = H_1(\mathtt{ID}),$$
where $\mathtt{ID}$ is the identifier string.

2.3 Identifier Based Encryption

We recap on the Boneh and Franklin encryption algorithm. Although much of what we will discuss also applies to other pairing based schemes such as the short signature scheme of Boneh, Lynn and Shacham [2] or the identity based signature schemes to be found in [5], [9] and [10].

The scheme of Boneh and Franklin [1], allows the holder of the private part $S_{\mathtt{ID}}$ of an identifier based key pair to decrypt a message sent to her under the public part $Q_{\mathtt{ID}}$. We present only the simple scheme which is only ID-OWE, for a ID-CCA scheme one applies the Fujisaki-Okamoto transformation [8].

Let m denote the message to be encrypted

- **Encryption :**
Compute $U = rP$ where r is a random element of F_q. Then compute
$$V = m \oplus H_2(t(R_{\mathtt{TA}}, rQ_{\mathtt{ID}}))$$
Output the ciphertext (U, V).
- **Decryption :**
Decryption is performed by computing
$$\begin{aligned} V \oplus H_2(t(U, S_{\mathtt{ID}})) &= V \oplus H_2(t(rP, sQ_{\mathtt{ID}})) \\ &= V \oplus H_2(t(P, Q_{\mathtt{ID}})^{rs}) \\ &= V \oplus H_2(t(sP, rQ_{\mathtt{ID}})) \\ &= V \oplus H_2(t(R_{\mathtt{TA}}, rQ_{\mathtt{ID}})) \\ &= m. \end{aligned}$$

3 Applications of the Addition of Keys

The main observation of this section is that we can add trust authorities public keys or users identifier based public keys together. We in this way obtain corresponding "virtual" secret keys to obtain an exponential number of keys. We shall go on to show a number of applications of this ability to add keys, a property which does not hold in standard cryptographic systems. We consider only the simplest case, other cases are possible but lead to more complicated protocols. In a later section we shall extend the notion and explain precisely which virtual keys one can obtain.

We first present the case where there are n trust authorities each with their own standard public/private key pair

$$R_{\mathtt{TA}_i} = s_i P.$$

We assume that all the public keys $R_{\mathtt{TA}_i}$ are trusted by all parties in the system. In addition, suppose we have a fixed identifier $\mathtt{ID}$ and we obtain the n private keys corresponding to this identifier from the relevant trust authorities, i.e. we have

$$S_{\mathtt{ID}_i} = s_i Q_{\mathtt{ID}}$$

where

$$Q_{\mathtt{ID}} = H_1(\mathtt{ID}).$$

Given an n bit string $\mathrm{b} = (b_1, \ldots, b_n)$ we can then form the private key

$$S_{\mathtt{ID},\mathrm{b}} = \sum_{i=1}^{n} b_i S_{\mathtt{ID}_i}$$

corresponding to the "virtual" trust authority with public key

$$R_{\mathtt{TA},\mathrm{b}} = \sum_{i=1}^{n} b_i R_{\mathtt{TA}_i}$$

and private key

$$s_{\mathrm{b}} = \sum_{i=1}^{n} b_i s_i.$$

In this way we can produce 2^n different public/private key pairs given only n trust authorities.

As a second case assume we have one fixed trust authority with its standard public/private key pair

$$R_{\mathtt{TA}} = sP.$$

Now assume we have n identifier's $\mathtt{ID}_i$, with private keys given by

$$S_{\mathtt{ID}_i} = sQ_{\mathtt{ID}_i}$$

where

$$Q_{\mathtt{ID}_i} = H_1(\mathtt{ID}_i).$$

Given an n bit string $\mathrm{b} = (b_1, \ldots, b_n)$ we can then form the private key

$$S_{\mathtt{ID},\mathrm{b}} = \sum_{i=1}^{n} b_i S_{\mathtt{ID}_i}$$

corresponding to the "virtual" identifier

$$Q_{\mathtt{ID},\mathrm{b}} = \sum_{i=1}^{n} b_i Q_{\mathtt{ID}_i}.$$

This two situations open up a large number of possible application areas which we shall now explore.

3.1 Escrowing of Keys

The first, obvious, application of key addition is that one does not need to have a single point of failure with a single trust authority. With identifier based cryptography the trust authority is always able to either decrypt messages (with an encryption scheme) or to sign messages (with a signature scheme). By using the key addition property one can distribute this key amongst a number of trust authorities, such that no single trust authority ever obtains more than their own portion. Indeed users, both encryptors and decryptors, can perform this operation themselves on the fly by selecting which subsets of trust authorities they will use in any given message. This property of identifier based systems in that the encryptor can choose, at the time of encryption, which trust authorities it is willing to trust we feel gives a significant advantage to using multiple trust authorities.

3.2 Contexts for Identities

A major problem when using an identity as a means to deduce a public key is that "names" for identities are not unique. There may be many "Ron Rivest"s in the world, so which one do I mean when I send an encrypted message to the person with identity "Ron Rivest"? We seem to have the same global name hierarchy problems that traditional PKI technologies come with.

Using multiple trust authorities we can add some context to each identity. For example suppose the International Association of Cryptologic Research (IACR) was a trust authority with public/private key pair

$$R_{\mathtt{IACR}} = rP.$$

They could issue all cryptographers with the name "Ron Rivest" the same public/private key pair

$$S_{\mathtt{Ron\ Rivest},\mathtt{IACR}} = rQ_{\mathtt{Ron\ Rivest}},$$

where

$$Q_{\texttt{Ron Rivest}} = H_1(\texttt{Ron Rivest}).$$

Now, suppose there was also a trust authority for the US government with public/private key pair given by

$$R_{\texttt{US}} = sP.$$

This trust authority would issue all US citizens with a public/private key pair corresponding to their name, i.e. Ron Rivest would be given the key pair

$$S_{\texttt{Ron Rivest,US}} = sQ_{\texttt{Ron Rivest}}.$$

In addition suppose there is a trust authority for the Massachusetts Institute of Technology with public/private key pair

$$R_{\texttt{MIT}} = tP.$$

They would issue all employees with the public/private key corresponding to their name as

$$S_{\texttt{Ron Rivest,MIT}} = tQ_{\texttt{Ron Rivest}}.$$

Suppose some third party wished to send an encrypted message to Ron Rivest. They may know that Ron Rivest is an US national and a cryptographer who works at MIT. They therefore create the "virtual" trust authority public key corresponding to what context they wish to put the identity of Ron Rivest in, i.e. they can create

$$R_{\texttt{virtual}} = R_{\texttt{IACR}} + R_{\texttt{US}} + R_{\texttt{MIT}}.$$

Note, the corresponding secret key for the virtual trust authority is

$$r + s + t,$$

but no party knows this key (and no two colluding parties can determine this key). The third party now encrypts the data for Ron Rivest using the virtual trust authorities public key and the public key

$$Q_{\texttt{Ron Rivest}}.$$

The encrypted data is then sent, along with some information to encode which contexts the identity is being considered in.

Ron Rivest can then decrypt the information using the private key

$$S_{\texttt{Ron Rivest,IACR}} + S_{\texttt{Ron Rivest,US}} + S_{\texttt{Ron Rivest,MIT}}.$$

Clearly this use of contexts to narrow down the possibilities for sending data to the wrong Ron Rivest also comes with the added benefit that no single trust authority can decrypt the encrypted message.

A similar technique can also be applied to any form of group membership, for example in access control systems where the rights to do some task may depend on whether the principal is a member of a number of different groups.

3.3 Legal Hoop Jumping

One can use a similar idea to create legal hoops through which people must jump before a certain action can be performed. We give an example, which may be peculiar to the United Kingdom, although others can be easily created.

In the United Kingdom every car needs to display a tax disk. This is purchased each year for a nominal fee, and essentially proves that at a given point in the year the owner of the car had car insurance and a certificate of road worthiness for the car. We describe a possible online car tax disk dispenser.

We note the three obvious trust authorities

- The ownership of the car is recorded by the Driver and Vehicle Licensing Agency (DVLA).
- The insurance certificate is produced by an insurance company, say AXA.
- The certificate of road worthiness is produced by an accredited garage, say Joes Garage.

The three trust authority public keys we denote by

$$R_{\texttt{DVLA}},\ R_{\texttt{AXA}},\ R_{\texttt{Joes}}.$$

Suppose the owner of the car with registration number `X 675 AHO` wished to obtain a new tax disk from the government. They could then log into some web site and claim that they owned the car, that they had insured it through AXA and that Joes Garage had issued them with a certificate of road worthiness. The government could then email the user an encrypted version of the tax disk, upon payment of some fee, where the encryption is under the virtual trust authority

$$R_{\texttt{DVLA}} + R_{\texttt{AXA}} + R_{\texttt{Joes}},$$

and the identifier is

$$Q_{\texttt{X 675 AHO}}.$$

The owner would need to obtain from each trust authority the corresponding private key (clearly date/year information needs to be added but we ignore that issue here for simplicity),

$$S_{\texttt{X 675 AHO,DVLA}},\ S_{\texttt{X 675 AHO,AXA}},\ S_{\texttt{X 675 AHO,Joes}}.$$

The owner now adds these private keys together to form another private key,

$$S_{\texttt{X 675 AHO,virtual}},$$

with which they can decrypt the electronic form of the tax disk and print it on their printer.

3.4 Concurrency Issues

One can imagine a situation where there is a trust authority which reveals the secret key corresponding to the identifier given by the time and date, at given time intervals. The secret key is broadcast to all, anyone who then uses the secret key in combination with others is therefore proving that they did so after a certain time. One should think of this trust authority acting like a Greenwich Pips signal. In this way issues of concurrency can be brought into applications.

One application would be that of press releases. Suppose a press release related to a companies accounts needs to be released simultaneously at 12.00 on June 1st to a group of financial journalists. To do this one takes the two public keys

$$Q_{\texttt{journalists}} = H_1(\texttt{journalists}),$$
$$Q_{\texttt{1200:01:06}} = H_1(\texttt{1200:01:06}),$$

and forms the sum

$$Q = Q_{\texttt{journalists}} + Q_{\texttt{1200:01:06}}.$$

If the trust authority for both identifies is the same $R_{\texttt{TA}}$, then using $R_{\texttt{TA}}$ and Q one can encrypt the press release and distribute it before the event, knowing that only after 12.00 on June 1st a given group of people (namely journalists) will be able to read the press release. Later we shall see a more general solution to this problem, which does not require the trust authorities to be the same.

3.5 Advantages over Traditional Solutions

Whilst in standard discrete logarithm based schemes one can still add keys together to form a "virtual" private key, the difference with identifier based schemes is that the public keys need to exist before the addition takes place. This is because the only way traditional keys are trusted is via a digital certificate. Hence, the trust authority (or CA for traditional schemes) needs to certify the public key before it is added to another one.

With identifier based schemes one knows the public key by simply knowing the identifier. Hence, one can add the public keys or TA keys together and encrypt the message before the entity singled out by the identifier has gone to the trust authority to obtain their private key.

4 A "Key Calculus"

In this section we formally define our key calculus which formed the basis of the examples above. The following formalism is richer than the examples above. We assume a set of "atomic" identifies I, these are identifiers which correspond to true identifier based public keys,

$$Q_{\texttt{ID}_i} = H_1(\texttt{ID}_i),$$

where $\mathtt{ID}_i \in I$.

We also assume a set of "atomic" trust authorities T. These are trust authorities which possess public/private key pairs,

$$R_{\mathtt{TA}_i} = s_i P,$$

where $\mathtt{TA}_i \in T$, for which the value of s_i is explicitly known by at least one party (i.e. the trust authority). We shall assume all parties in the system have trusted copies of the TA's public keys $R_{\mathtt{TA}_i}$.

To each pair $(\mathtt{ID}, \mathtt{TA})$ of atomic identifier/trust authority keys, called an atomic pair, there is a corresponding secret key denoted by

$$S_{\mathtt{ID},\mathtt{TA}} = s_{\mathtt{TA}} Q_{\mathtt{ID}},$$

although it may be the case that no one in the system, bar the trust authority, knows the value of $S_{\mathtt{ID},\mathtt{TA}}$.

The goal is to encrypt a message m to a set of people who know a given subset of keys. We would like to do this in as short a message as possible and be able to describe completely exactly which set of keys are needed to decrypt the message.

4.1 Naive Conjunction of Sets

We let S denote a set of atomic pairs $(\mathtt{ID}_i, \mathtt{TA}_j)$. We first wish to encrypt a message so that it can be decrypted only by people who possess, for every pair in S, the equivalent atomic secret key. This can be trivially done using an onion form of encryption where each encryption is applied in turn to obtain the ciphertext. Not only is this trivial technique wasteful of computing resources it also implies that the decryptor needs to apply the necessary decryptions in the same order that the encryptor applied the encryptions.

We will try and be more efficient, but this will come at the price of not being able to deal with all possible sets S. First we define an operation of sets of atomic pairs S_1, S_2

$$\mathrm{S}_1 \otimes \mathrm{S}_2 = \{(\mathtt{ID}_i, \mathtt{TA}_j) : (\mathtt{ID}_i, \mathtt{TA}') \in \mathrm{S}_1 \cup \mathrm{S}_2, (\mathtt{ID}', \mathtt{TA}_j) \in \mathrm{S}_1 \cup \mathrm{S}_2\}.$$

A set S will be called admissible if

- $\mathrm{S} = \{(\mathtt{ID}, \mathtt{TA})\}$ consists of a single atomic pair.
- or S is built up from two admissible sets S_1 and S_2 using the operation

$$\mathrm{S} = \mathrm{S}_1 \otimes \mathrm{S}_2.$$

We let $\mathtt{ID}(\mathrm{S})$ denote the set of identifiers and $\mathtt{TA}(\mathrm{S})$ denote the set of trust authorities contained in an admissible set S. The "virtual" trust authority corresponding to S is given by

$$R_{\mathrm{S}} = \sum_{\mathtt{TA} \in \mathtt{TA}(\mathrm{S})} R_{\mathtt{TA}},$$

and the "virtual" identifier is given by

$$Q_{\mathrm{S}} = \sum_{\mathtt{ID} \in \mathtt{ID}(\mathrm{S})} Q_{\mathtt{ID}}.$$

Encrypting using these two public keys, namely R_{S} and Q_{S}, means that only recipients who possess **every** private key corresponding to every pair in S will be able to decrypt the message. Equivalently, the decryptor needs to know

$$S_{\mathrm{S}} = \sum_{\mathtt{ID} \in \mathtt{ID}(\mathrm{S})} \left(\sum_{\mathtt{TA} \in \mathtt{TA}(\mathrm{S})} S_{\mathtt{ID},\mathtt{TA}} \right).$$

Hence, we call this secret key the virtual secret key corresponding to the admissible set S.

Example To see the use of the special conjunction operation $\otimes$ consider the case of two trust authorities and two identifiers. We form the admissible atomic sets

$$\mathrm{S}_1 = \{(\mathtt{ID}_1, \mathtt{TA}_1)\}, \quad \mathrm{S}_2 = \{(\mathtt{ID}_2, \mathtt{TA}_2)\},$$

where $R_{\mathtt{TA}_j} = s_j P$. Then

$$\mathrm{S} = \mathrm{S}_1 \otimes \mathrm{S}_2 = \{(\mathtt{ID}_1, \mathtt{TA}_1), (\mathtt{ID}_2, \mathtt{TA}_1), (\mathtt{ID}_1, \mathtt{TA}_2), (\mathtt{ID}_2, \mathtt{TA}_2)\}.$$

Suppose a cryptogram (U, V) is sent, in the Boneh-Franklin scheme, to the virtual trust authority and identifier corresponding to the admissible set S. This cryptogram corresponds to the public key of the trust authority

$$R = R_{\mathtt{TA}_1} + R_{\mathtt{TA}_2},$$

and the "identifier" given by

$$Q = Q_{\mathtt{ID}_1} + Q_{\mathtt{ID}_2}.$$

To decrypt the cryptogram we need to compute

$$\begin{aligned} t(R, rQ) &= \prod_{i,j=1}^{2} t(R_{\mathtt{TA}_j}, rQ_{\mathtt{ID}_i}) = \prod_{i,j=1}^{2} t(s_j P, rQ_{\mathtt{ID}_i}), \\ &= \prod_{i,j=1}^{2} t(rP, s_j Q_{\mathtt{ID}_i}) = \prod_{i,j=1}^{2} t(U, S_{\mathtt{ID}_i,\mathtt{TA}_j}), \\ &= t(U, \sum_{i,j=1}^{2} S_{\mathtt{ID}_i,\mathtt{TA}_j}). \end{aligned}$$

Hence, we need to know the four secret keys $S_{\mathtt{ID}_i,\mathtt{TA}_j}$ for $1 \leq i, j \leq 2$, or in other words we need to know the "virtual" secret key

$$S_{\mathrm{S}} = \sum_{i,j=1}^{2} S_{\mathtt{ID}_i,\mathtt{TA}_j}.$$

We have seen in Section 3 examples where either $\mathtt{ID}_i = \mathtt{ID}_{i'}$ for all i, i' or $\mathtt{TA}_j = \mathtt{ID}_{j'}$ for all j, j'. We leave it to the reader to check all our previous examples correspond to admissible sets.

4.2 General Conjunction of Sets

We have seen that encryption to the virtual trust authorities and virtual identifier represented by an admissible set S is equivalent to encryption to an entity which possesses the virtual secret key S_{S}. From now on we will drop the usage of the word "virtual" when dealing with the trust authority and identifier based keys represented by an admissible set S.

The above form of conjunction of keys is too restrictive in that it does not allow us to encrypt to someone who holds the secret key corresponding to the pairs $(\mathtt{ID}_1, \mathtt{TA}_1)$ and $(\mathtt{ID}_2, \mathtt{TA}_2)$, without knowing also knowing the secret keys corresponding to $(\mathtt{ID}_2, \mathtt{TA}_1)$ and $(\mathtt{ID}_2, \mathtt{TA}_1)$.

To enable this we change the encryption algorithm slightly. Suppose we have admissible sets S_i, corresponding to the pairs $(\mathtt{ID}_i, \mathtt{TA}_i)$. To encrypt to the conjunction $\mathrm{S}_1 \vee \mathrm{S}_2 \vee \ldots \vee \mathrm{S}_n$ we transmit $U = rP$ and

$$V = m \oplus H_2 \left(\prod_{i=1}^{n} t(R_{\mathtt{TA}_i}, rQ_{\mathtt{ID}_i}) \right).$$

Decryption is performed by computing

$$m = V \oplus H_2 \left(t(U, \sum_{i=1}^{n} S_{\mathtt{ID}_i, \mathtt{TA}_i}) \right).$$

So for decryption we still add private keys together, but for encryption we need to multiply ephemeral keys together, after they have been passed through the pairing.

We can now return to our press release example. Using an authority for time, which simply issues the private key according to the current time at given time intervals, and another (different) authority to issue private keys to all journalists, we can encrypt a press release so that the intended recipients can only read it at some point in the future. Notice, how this is simple using identifier based systems since the encryptor does not need to know in advance what the time authority will do. The encryptor need only know that the authority will issue the private key corresponding to a given identifier.

With a traditional public key system one could obtain the same effect but only by knowing the public key corresponding to the private key at the encryption stage, but this would involve the encryptor asking the time authority for precisely which public key should be used.

4.3 Naive Disjunction of Sets

The use of variable identifiers has little use unless one is able to define a disjunction operation. For example one may wish to send an encrypted email to either

the president or CEO of the ACME company. Hence, using the trust authority ACME one would want to encrypt to either

$$(\texttt{PRESIDENT}, \texttt{ACME}) \text{ or } (\texttt{CEO}, \texttt{ACME}).$$

We now show how to modify the Boneh-Franklin encryption scheme so that we can encrypt to a disjunction of admissible sets, in a way which is akin to the naive conjunction above, in the next subsection we shall deal with a general form of disjunction which only works for two admissible sets. The following idea of a naive disjunction will work for an arbitrary collection of trust authorities and/or identifiers but we present, for ease of exposition, the case where we have two trust authorities and two identifiers.

Suppose the two trust authorities have public keys

$$R_{T_1} = s_1 P \text{ and } R_{T_2} = s_2 P.$$

Let the two identifiers be

$$Q_A \text{ and } Q_B.$$

Hence, there are a total of four secret keys

$$\begin{aligned} S_{A,1} &= s_1 Q_A, \quad & S_{A,2} &= s_2 Q_A, \\ S_{B,1} &= s_1 Q_B, \quad & S_{B,2} &= s_2 Q_B. \end{aligned}$$

We would like to encrypt a message so that a party who has any one of these secret keys is able to decrypt the message. We make the assumption that there are four fixed integers

$$x_1, x_2, y_1, y_2 \in \mathrm{F}_q^*$$

known to all parties, such that $x_1 \neq x_2$ and $y_1 \neq y_2$. We then form the virtual public keys given by

$$R_V = R_{T_1} + y_1 R_{T_2} \text{ and } Q_V = Q_A + x_1 Q_B.$$

We also form the "auxiliary" keys,

$$T_R = R_{T_1} + y_2 R_{T_2} \text{ and } T_Q = Q_A + x_2 Q_B.$$

The cryptogram is then formed by computing, for some random $r \in \mathrm{F}_q^*$,

$$\begin{aligned} U_1 &= rP, \quad & U_2 &= rT_Q, \quad & U_3 &= rT_R, \\ V &= m \oplus H_2(t(R_V, rQ_V)). \end{aligned}$$

To decrypt the message (U_1, U_2, U_3, V) we will make use of the identities,

$$\begin{aligned} R_V &= \alpha R_{T_1} + \beta T_R \text{ or } R_V = \gamma R_{T_2} + T_R, \\ Q_V &= \delta Q_A + \epsilon T_Q \text{ or } Q_V = \zeta Q_B + T_Q, \end{aligned}$$

where

$$\begin{aligned} \alpha &= (y_2 - y_1)/y_2, \quad & \beta &= y_1/y_2, \quad & \gamma &= y_1 - y_2, \\ \delta &= (x_2 - x_1)/x_2, \quad & \epsilon &= x_1/x_2, \quad & \zeta &= x_1 - x_2. \end{aligned}$$

The precise identities one uses will depend on which of the secret keys the decryptor has access to.

Decryptor knows $S_{A,1}$ We first show how to decrypt assuming the recipient has the private key $S_{A,1}$. The decryptor needs to compute $t(R_V, rQ_V)$, which with knowledge of $S_{A,1}$ they can do via:

$$\begin{aligned} t(R_V, rQ_V) &= t\left(\alpha R_{T_1} + \beta T_R, r(\delta Q_A + \epsilon T_Q)\right), \\ &= t(\alpha R_{T_1}, r\delta Q_A) \cdot t(\alpha R_{T_1} + \beta T_R, r\epsilon T_Q) \cdot t(\beta T_R, r\delta Q_A), \\ &= t(rs_1\alpha P, \delta Q_A) \cdot t(\alpha R_{T_1} + \beta T_R, \epsilon U_2) \cdot t(r\beta T_R, \delta Q_A), \\ &= t(r\alpha P, s_1\delta Q_A) \cdot t(\alpha R_{T_1} + \beta T_R, \epsilon U_2) \cdot t(\beta U_3, \delta Q_A), \\ &= t(\alpha U_1, \delta S_{A,1}) \cdot t(\alpha R_{T_1} + \beta T_R, \epsilon U_2) \cdot t(\beta U_3, \delta Q_A). \end{aligned}$$

Decryptor knows $S_{A,2}$ We first show how to decrypt assuming the recipient has the private key $S_{A,2}$. The decryptor needs to compute $t(R_V, rQ_V)$, which with knowledge of $S_{A,2}$ they can do via:

$$\begin{aligned} t(R_V, rQ_V) &= t\left(\gamma R_{T_2} + T_R, r(\delta Q_A + \epsilon T_Q)\right), \\ &= t\left(\gamma R_{T_2}, r(\delta Q_A + \epsilon T_Q)\right) \cdot t(rT_R, \delta Q_A + \epsilon T_Q), \\ &= t(\gamma r s_2 P, \delta Q_A) \cdot t(\gamma R_{T_2}, r\epsilon T_Q) \cdot t(U_3, \delta Q_A + \epsilon T_Q), \\ &= t(\gamma U_1, \delta S_{A,2}) \cdot t(\gamma R_{T_2}, \epsilon U_2) \cdot t(U_3, \delta Q_A + \epsilon T_Q). \end{aligned}$$

The other two cases we leave for the reader, since they are computed in a similar way.

4.4 General Disjunction of Sets

Just as with our earlier naive conjunction, which led to the construction of the admissible sets, the above naive disjunction has a number of problems. The main being that when encrypting to the disjunction of the two pairs $(\mathtt{ID}_1, \mathtt{TA}_1)$ and $(\mathtt{ID}_2, \mathtt{TA}_2)$, the above construction will allow anyone with the secret key corresponding to the pair $(\mathtt{ID}_1, \mathtt{TA}_2)$ to decrypt. This may not be the effect we are after.

In this subsection we show how a proper disjunction can be created between two identifier/authority pairs, such that only the desired recipients can decrypt the resulting ciphertext. Once again we are looking for a more efficient solution than simply encrypting the message twice to different possible recipients. Our solution however does not extend to forming a disjunction of more than two identifier/authority pair, unlike the naive form of disjunction above.

Suppose I have the two pairs $(\mathtt{ID}_1, \mathtt{TA}_1)$ and $(\mathtt{ID}_2, \mathtt{TA}_2)$. To encrypt to either of these pairs one forms the virtual identifier/trust authority keys given by

$$R_V = Q_B + x_1 R_{T_1} \text{ and } Q_V = Q_A + y_1 R_{T_2},$$

and the auxiliary keys given by

$$T_R = Q_B + x_2 R_{T_1} \text{ and } T_Q = Q_A + y_2 R_{T_2},$$

again assuming some globally fixed integers $x_1, x_2, y_1, y_2 \in \mathrm{F}_q^*$.

The cryptogram is then formed by computing, for some random $r \in \mathrm{F}_q^*$,

$$\begin{aligned} U_1 = rP, \quad U_2 = rT_Q, \quad U_3 = rT_R, \\ V = m \oplus H_2(t(R_V, rQ_V)). \end{aligned}$$

To decrypt the message (U_1, U_2, U_3, V) we will make use of the identities,

$$\begin{aligned} R_V = \alpha R_{T_1} + T_R \ \text{ or } \ R_V = \beta Q_B + \gamma T_R, \\ Q_V = \delta Q_A + \epsilon T_Q \ \text{ or } \ Q_V = \zeta R_{T_2} + T_Q, \end{aligned}$$

where

$$\begin{array}{lll} \alpha = x_1 - x_2, & \beta = (x_2 - x_1)/x2, & \gamma = x_1/x_2, \\ \delta = (y_2 - y_1)/y_2, & \epsilon = y_1/y_2, & \zeta = y_1 - y_2. \end{array}$$

The precise identities one uses will depend on which of the secret keys the decryptor has access to.

If the decryptor has knowledge of $S_{A,1}$ they use the identities

$$R_V = \alpha R_{T_1} + T_R \text{ and } Q_V = \delta Q_A + \epsilon T_Q$$

to decrypt using

$$\begin{aligned} t(R_V, rQ_V) &= t(\alpha R_{T_1} + T_R, r\delta Q_A + r\epsilon T_Q), \\ &= t(\alpha rP, \delta S_{A,1}) \cdot t(\alpha R_{T_1}, r\epsilon T_Q) \cdot t(rT_R, Q_V), \\ &= t(\alpha U_1, \delta S_{A,1}) \cdot t(\alpha R_{T_1}, \epsilon U_2) \cdot t(U_3, Q_V). \end{aligned}$$

If the decryptor has knowledge of $S_{B,2}$ they use the identities

$$R_V = \beta Q_B + \gamma T_R \text{ and } Q_V = \zeta R_{T_2} + T_Q$$

to decrypt using

$$\begin{aligned} t(R_V, rQ_V) &= t(\beta Q_B + \gamma T_R, r\zeta R_{T_2} + rT_Q), \\ &= t(\beta S_{B,2}, \zeta rP) \cdot t(\beta Q_B, rT_Q) \cdot t(\gamma rT_R, Q_V) \\ &= t(\beta S_{B,2}, \zeta U_1) \cdot t(\beta Q_B, U_2) \cdot t(\gamma U_3, Q_V). \end{aligned}$$

Finally notice if the decryptor uses the other two combinations of the identities then they recover no information since, for example, using

$$R_V = \beta Q_B + \gamma T_R \text{ and } Q_V = \delta Q_A + \epsilon T_Q$$

we obtain

$$\begin{aligned} t(R_V, rQ_V) &= t(\beta Q_B + \gamma T_R, \delta rQ_A + \epsilon rT_Q), \\ &= t(\beta Q_B, \delta rQ_A) \cdot t(\gamma U_3, \delta Q_A) \cdot t(R_V, \epsilon U_2) \end{aligned}$$

Hence, to decrypt, one would need knowledge of either rQ_A or rQ_B neither of which is available to the decryptor.

5 Conclusion

We have seen how multiple trust authorities combined with conjunctions of keys can be used in a variety of applications. We have explained how various conjunctions of keys (or essentially access rights) can be created in an efficient manner. The use of identifier based systems with multiple trust authorities and multiple identifiers we believe opens up simplified ways of specifying access rights, a number of possible real life examples we have given in the text.

We have shown how to build up so called 'admissible sets' of identifier/trust authority pairs, via a form of specialised conjunction. These admissible sets can then be used to form general conjunctions, and or a special form of naive disjunction. The general form of disjunction is only possible for two such sets. A general form of disjunction for arbitrary numbers of admissible sets we leave as an open research problem.

References

1. D. Boneh and M. Franklin. Identity based encryption from the Weil pairing. *Advanced in Cryptology - CRYPTO 2001*, Springer-Verlag LNCS 2139, 213–229, 2001.
2. D. Boneh, B. Lynn and H. Shacham. Short signatures from the Weil pairing. *Advances in Cryptology - ASIACRYPT 2001*, Springer-Verlag LNCS 2248, 514–532, 2001.
3. C. Boyd. Some applications of multiple key ciphers. *Advances in Cryptology - EUROCRYPT '88*, Springer-Verlag LNCS 330, 455–467, 1988.
4. C. Boyd. A new multiple key cipher and an improved voting scheme. *Advances in Cryptology - EUROCRYPT '89*, Springer-Verlag LNCS 434, 617–625, 1989.
5. J. C. Cha and J. H. Cheon. An Identity-Based Signature from Gap Diffie-Hellman Groups. Preprint 2002.
6. C. Cocks. An identity based encryption scheme based on quadratic residues. *Cryptography and Coding*, Springer-Verlag LNCS 2260, 360–363, 2001.
7. Y. Desmedt. Society and group oriented cryptography: A new concept. *Advances in Cryptology - CRYPTO '87*, Springer-Verlag LNCS 293, 120–127, 1987.
8. E. Fujisaki and T. Okamoto. Secure integration of asymmetric and symmetric encryption schemes. *Advances in Cryptology - CRYPTO '99*, Springer-Verlag LNCS 1666, 537–554, 1999.
9. F. Hess. Efficient identity based signature schemes based on pairings. To appear *Selected Areas in Cryptography – 2002*.
10. K. Paterson. ID-based Signatures from Pairings on Elliptic Curves. Preprint 2002.
11. A. Shamir. Identity based cryptosystems and signature schemes. *Advanced in Cryptology - CRYPTO '84*, Springer-Verlag LNCS 196, 47–53, 1985.