Identity Based Key Agreement Protocols

N.P. Smart

Department of Computer Science,
University Of Bristol,
Merchant Venturers Building,
Woodland Road,
Bristol, BS8 1UB.

Joint work with Liqun Chen and Michael Cheng

24th July 2006

Outline

Outline

Types of Pairings

A set of pairing parameters for cryptographic use is a set of three groups $\mathbb{G}_1$, $\mathbb{G}_2$ and $\mathbb{G}_T$.

The DLP in each of these groups should be hard.

The exponent of each group should be divisible by a large prime q

There should be a bilinear map

$$\hat{e} : \mathbb{G}_1 \times \mathbb{G}_2 \longrightarrow \mathbb{G}_T$$

In addition various protocols require certain other properties...

Types of Pairings

Let $\mathcal{G} = E[q]$, the points of order q on an elliptic curve over $\mathbb{F}_p$.

The group $E[q]$ is contained in $E(\mathbb{F}_{p^k})$

- For efficiency we assume that k is even.

$\mathcal{G}$ is a product of two cyclic groups $\mathcal{G}_1$, $\mathcal{G}_2$ of order q.

Let $\mathcal{P}_1 \in E(\mathbb{F}_p)$ be a generator of $\mathcal{G}_1$

Let $\mathcal{P}_2 \in E(\mathbb{F}_{p^k})$ be a generator of $\mathcal{G}_2$.

- $\mathcal{P}_2$ is in the image of the quadratic twist of E over $\mathbb{F}_{p^{k/2}}$.

Types of Pairings

There is a pairing $\hat{e}$ from $\mathcal{G} \times \mathcal{G}$ to the subgroup $\mathcal{G}_T$ of order q of the finite field $\mathbb{F}_{p^k}$.

This pairing is trivial if and only if the two input values are linearly dependent in the vector space $E[q]$.

The trace map

$$\mathrm{Tr}: \left\{ \begin{array}{ccc} E(\mathbb{F}_{p^k}) & \longrightarrow & E(\mathbb{F}_p), \\ P & \longmapsto & \sum_{\sigma \in Gal(\mathbb{F}_{p^k}/\mathbb{F}_p)} P^\sigma, \end{array} \right.$$

defines a group homomorphism on $E[q]$ which has kernel $\mathcal{G}_2$.

Types of Pairings

An important point to note is that Tr and the pairing do not necessarily commute:

$$\hat{e}(\mathrm{Tr}(A), B) = \hat{e}(\mathrm{Tr}(B), A)$$

if and only if A and B lie in the same order q subgroup of $\mathcal{G}$.

In addition it is easy to produce a hash function which hashes onto $\mathcal{G}_1$, $\mathcal{G}_2$ or $\mathcal{G}$

It is not easy to produce a function which hashes onto any other subgroup of order q of $\mathcal{G}$, bar $\mathcal{G}_1$ and $\mathcal{G}_2$.

We shall define four types of cryptographic pairing parameters.

- In all cases $\mathbb{G}_T = \mathcal{G}_T$.

Type 1 Pairings

If we are using a supersingular elliptic curve:

Set $\mathbb{G}_1 = \mathbb{G}_2 = \mathcal{G}_1$.

We let $P_1 = P_2 = \mathcal{P}_1$ denote the generators of $\mathbb{G}_1$ and $\mathbb{G}_2$.

Pairing is defined via a distortion map

There is an efficient algorithm to cryptographically hash arbitrary bit strings into $\mathbb{G}_1$ and $\mathbb{G}_2$

There is a trivial group isomorphism $\psi : \mathbb{G}_2 \to \mathbb{G}_1$ mapping P_2 to P_1.

Type 2 Pairings

If we are using an ordinary elliptic curve:

Set $\mathbb{G}_1 = \mathcal{G}_1$ and $\mathbb{G}_2$ to be a subgroup of $\mathcal{G}$ which is not equal to either $\mathcal{G}_1$ or $\mathcal{G}_2$.

Let $P_1 = \mathcal{P}_1$ and for convenience we set $P_2 = \frac{1}{k}\mathcal{P}_1 + \mathcal{P}_2$.

There is an efficient algorithm to cryptographically hash arbitrary bit strings into $\mathbb{G}_1$, but there is no way to hash bit strings into $\mathbb{G}_2$ (nor to generate random elements of $\mathbb{G}_2$ bar multiplying P_2 by an integer).

There is an efficiently computable group isomorphism $\psi : \mathbb{G}_2 \rightarrow \mathbb{G}_1$ mapping P_2 to P_1, which is simply the trace map restricted to $\mathbb{G}_2$.

Type 3 Pairings

If we are using an ordinary elliptic curve:

Set $\mathbb{G}_1 = \mathcal{G}_1$ and $\mathbb{G}_2 = \mathcal{G}_2$.

Let $P_1 = \mathcal{P}_1$ and $P_2 = \mathcal{P}_2$ be generators of $\mathbb{G}_1$ and $\mathbb{G}_2$.

There is an efficient algorithm to cryptographically hash arbitrary bit strings into $\mathbb{G}_1$, and a slightly less efficient algorithm to hash bit strings into $\mathbb{G}_2$.

There is no known efficiently computable group isomorphism $\psi : \mathbb{G}_2 \to \mathbb{G}_1$ mapping P_2 to P_1.

Type 4 Pairings

If we are using an ordinary elliptic curve:

Set $\mathbb{G}_1 = \mathcal{G}_1$, select $\mathbb{G}_2$ to be the whole group $\mathcal{G}$ which is a group of order q^2.

As in the Type 2 situation we set $P_1 = \mathcal{P}_1$ and $P_2 = \frac{1}{k}\mathcal{P}_1 + \mathcal{P}_2$.

Hashing into $\mathbb{G}_1$ or $\mathbb{G}_2$ can be performed, although maybe not very efficiently into $\mathbb{G}_2$. However, one cannot hash efficiently into the subgroup of $\mathbb{G}_2$ generated by P_2.

There is an efficiently computable homomorphism ψ from $\mathbb{G}_2$ to $\mathbb{G}_1$ such that $\psi(P_2) = P_1$.

Note, that the pairing of a non-zero element in $\mathbb{G}_1$ and a non-zero element in $\mathbb{G}_2$ may be trivial in this situation.

Summary

In all situations we have that

- P_1 is the generator of $\mathbb{G}_1$.
- P_2 is a fixed element of $\mathbb{G}_2$ of prime order q.
  - Such that where there is a computable homomorphism ψ from $\mathbb{G}_2$ to $\mathbb{G}_1$ we have $\psi(P_2) = P_1$.

In Type 3 curves, an isomorphism exists however you cannot compute it.

- We will still refer to ψ in this situation.

Curve Choices

Type 1 curves do not scale very well as one increaes the security parameter, hence from now on we assume we are using ordinary curves.

The most efficient parameters are those ordinary curves with complex multiplication by $D = -3$ and k divisible by six.

- Efficient arithmetic in $\mathbb{G}_2$ via the sextic twist.
- Efficient pairing using the Ate-pairing.
- Reduced bandwidth if k selected sensibly.

Outline

Subgroup Membership Testing

In security proofs of key agreement protocols it is often implicitly assumed that elements transmitted lie in the correct subgroup of a larger group.

In practice one needs then to check for subgroup membership

Often forgotten about

- If you do not do it the security proof does not apply.

For each of our ordinary curve pairing parameters, i.e. Type 2, 3, 4, we need to show how to test for subgroup membership.

Subgroup Membership Testing

Almost always the message flows will be elements of $\mathcal{G}_1$, $\mathcal{G}_2$ or $\mathcal{G}_T$.

Detecting whether an octet string is an element of a finite field, or a point on a curve is easy.

- The question is whether the element/point is in the correct subgroup.

For $\mathcal{G}_T$ standard techniques apply, such as cofactor multiplication.
For $\mathbb{G}_1$, since elements always lie in $E(\mathbb{F}_p)$ and have order q.

- Thus standard cofactor multiplication can be applied.

For $\mathbb{G}_2$, for Type 2,3,4 parameters, elements lie in $E(\mathbb{F}_{p^k})$

- This has order divisible by q^2, so standard techniques need to be adapted.
- Depends on the type of pairing parameters

Subgroup Membership Testing

Type 3 Here $\mathbb{G}_2$ is the image of the quadratic/sextic twist over a the field $\mathbb{F}_{p^k/2}/\mathbb{F}_{p^k/6}$.

- Represent elements of $\mathbb{G}_2$ as on the twist.
- Subgroup testing then done by standard techniques.

Type 2 Here $\mathbb{G}_2$ is generated by $P_2 = \frac{1}{k}\mathcal{P}_1 + \mathcal{P}_2$.
If we wish to test whether $Q \in \langle P_2 \rangle$

- We first check whether it has order q.
- We then know that $Q = a\mathcal{P}_1 + b\mathcal{P}_2$ for unknown a and b.
- We compute $a\mathcal{P}_1 = \frac{1}{k}\mathrm{Tr}(Q)$ and $b\mathcal{P}_2 = Q - a\mathcal{P}_1$
- We need to test whether $a = b/k$, which we do via

$$\hat{e}(\mathrm{Tr}(Q), \mathcal{P}_2) = \hat{e}(ka\mathcal{P}_1, \mathcal{P}_2) = \hat{e}(\mathcal{P}_1, b\mathcal{P}_2) = \hat{e}(\mathcal{P}_1, Q - \frac{1}{k}\mathrm{Tr}(Q)).$$

Subgroup Membership Testing

Type 4 In this situation we also need to test whether a general point

$$Q = a\mathcal{P}_1 + b\mathcal{P}_2$$

is a multiple of another point

$$P = c\mathcal{P}_1 + d\mathcal{P}_2$$

without knowing a, b, c or d.

We first test whether $P, Q \in \mathcal{G}$ as above.

Then we test whether $a = tc$ and $b = td$ for some unknown t by testing whether

$$\hat{e}(\mathrm{Tr}(Q), P - \frac{1}{k}\mathrm{Tr}(P)) = \hat{e}(\mathrm{Tr}(P), Q - \frac{1}{k}\mathrm{Tr}(Q)).$$

Outline

Hard Problems

We require a set of hard problems on which to base our protocols:

Diffie–Hellman (DH)

For $a, b \in_R \mathbb{Z}_q^*$ and some values of $i, j, k \in \{1, 2\}$, given (aP_i, bP_j), computing abP_k is hard.

- Use the notation "DH$_{i,j,k}$ problem".

Bilinear Diffie–Hellman (BDH)

For $a, b, c \in_R \mathbb{Z}_q^*$, given (aP_i, bP_j, cP_k), for some values of $i, j, k \in \{1, 2\}$, computing $\hat{e}(P_1, P_2)^{abc}$ is hard.

Decisional BDH (DBDH)

For $a, b, c, r \in_R \mathbb{Z}_q^*$, differentiating

$$(aP_i, bP_j, cP_k, \hat{e}(P_1, P_2)^{abc}) \text{ and } (aP_i, bP_j, cP_k, \hat{e}(P_1, P_2)^{r}),$$

for some values of $i, j, k \in \{1, 2\}$, is hard.

Hard Problems: Variants

A particular scheme may not require the computable homomorphism to implement it

But the computable homomorphism may be required in the security proof.

Thus for Type 3 curves, where no such isomorphism exists, we are creating a relativised security proof

We denote the corresponding relativised hard problem by a superscript-ψ,

- As in $\mathrm{DH}^{\psi}_{2,2,1}$, $\mathrm{BDH}^{\psi}_{2,1,2}$, etc.

Hard Problems: Variants

Some security proofs make use of some gap assumptions.

They assume that if an algorithm exists to resolve a decisional problem, the corresponding computational problem is still hard.

We let

- GBDH,

stand for

- the gap BDH assumption

Outline

University of BRISTOL

ID-Based Key Agreement

Quite early on in the history of pairing based crypto ID-based key agreement was considered.

In this talk we do not consider whether such a primitive is useful or not.

Just as in standard key agreement various properties can be considered:

- Known Session Key Security.
- Forward Secrecy.
- Key-Compromise Impersonation Resilience.
- Unknown Key-Share Resilience.
- Role Symmetry.

ID-Based Key Agreement

In this talk we restrict to ID-based keys of the SOK/BF format:

- See our paper for a full survey of all current protocols

SOK/BF key extraction comes in two variants:

- Which we refer to as Extract 1 and Extract 1'.

In both cases we have

- The pairing parameters
- An identity string ID_A for a user A
- The master private key $s \in \mathbb{Z}_q^*$,
- The master public key,
  - This is either $R = sP_1 \in \mathbb{G}_1$ or $R' = sP_2 \in \mathbb{G}_2$ or both.

ID-Based Key Agreement

Extract 1

Here we have a hash-function $H_1 : \{0,1\}^* \rightarrow \mathbb{G}_1$,

The algorithm computes

- $Q_A = H_1(ID_A) \in \mathbb{G}_1$
- $d_A = sQ_A \in \mathbb{G}_1$.

Extract 1'

This is the same, except that H_1 is now a hash function with codomain $\mathbb{G}_2$, and hence Q_A and d_A lie in $\mathbb{G}_2$.

In both cases, the values Q_A and d_A will be used as the public and private key pair corresponding to A's identity ID_A.

Smart's Protocol

The first ID-based key agreement protocol was given by Smart.

Use Extract 1 method for key extraction.

Alice and Bob randomly choose x and y from $\mathbb{Z}_q^*$ and perform the protocol as follows:

$$\begin{aligned} A \to B &: \quad E_A = xP_2, \\ B \to A &: \quad E_B = yP_2. \end{aligned}$$

The shared secret key is

$$\hat{e}(xQ_B + yQ_A, P_2)^s.$$

- Alice computes this via $\hat{e}(xQ_B, R') \cdot \hat{e}(d_A, E_B)$.
- Bob computes this via $\hat{e}(yQ_A, R') \cdot \hat{e}(d_B, E_A)$.

Smart’s Protocol

Smart’s protocol is

- Implementable in all pairing parameter types.
- Role symmetric.
- Has all required security properties
  - Bar some strict forms of forward secrecy.
- Provably securely under the GBDH assumption (Kudla/Paterson)

If using the Extract 1’ method:

- Can not be implemented in parameter Type 2.
- Need to replace the message flows by xP_1 and yP_1.
- Obtain shorter message flows in this case.
- Detecting subgroup membership for Type 3 and 4 pairings is then easier.

Smart's Protocol

Would like a protocol which is better than Smart's protocol.

In this talk will then only concentrate on protocols which meet this property.

In particular for this talk we will concentrate on protocols which

- Have a proof of security
- Secure against known key security and key compromise impersonation and unknown key shares.
-

See paper for other protocols.

SYL Protocol

Using the Extract 1' method only can define another protocol due to Shim, Yuan and Li.

$$\begin{aligned} A \rightarrow B &: \quad E_A = xP_2, \\ B \rightarrow A &: \quad E_B = yP_2. \end{aligned}$$

The shared secret key is

$$xyP_2 \| \hat{e}(yP_1 + \psi(Q_B), xP_2 + Q_A)^s$$

- Alice computes this via $xE_B \| \hat{e}(\psi(E_B + Q_B), xR' + d_A)$.
- Bob computes this via $yE_A \| \hat{e}(y\psi(R') + \psi(d_B), E_A + Q_A)$.

SYL Protocol

SYL protocol is

- Implementable only in pairing parameter Type 1 and 4.
  - Requires hashing into $\mathbb{G}_2$ and an isomorphism.
- Not role symmetric in the Type 4 setting.
- Has all required security properties
  - Including all strict forms of forward secrecy.
- Provably securely under the BDH assumption (Chen/Cheng/Smart)
- Has poor bandwidth efficiency in the Type 4 setting.

CK and Wang Protocols

These are from the same family, first proposed by Chen and Kudla.

Key extraction is by the Extract 1' method.

They require the isomorphism to implement the protocol

These two properties mean they only hold in the Type 1 and 4 setting.

Message flows are given by

$$\begin{aligned} A \rightarrow B &: \quad E_A = x\psi(Q_A), \\ B \rightarrow A &: \quad E_B = yQ_B. \end{aligned}$$

CK and Wang Protocols

For the CK protocol the shared secret key is

$$\hat{e}(\psi(Q_A), Q_B)^{s(x+y)}$$

- Alice computes this via $\hat{e}(\psi(d_A), xQ_B + E_B)$
- Bob computes this via $\hat{e}(E_A + y\psi(Q_A), d_B)$

For the Wang protocol the shared secret key is

$$\hat{e}(\psi(Q_A), Q_B)^{s(x+s_A)(y+s_B)}$$

where $s_A = h(x\psi(Q_A), yQ_B)$ and $s_B = h(yQ_B, x\psi(Q_A))$ and h is a one-way function.

- Alice computes this via $\hat{e}((x + s_A)\psi(d_A), s_BQ_B + E_B)$
- Bob computes this via $\hat{e}(s_A\psi(Q_A) + E_A, (y + s_B)d_B)$

CK and Wang Protocols

The CK and Wang protocols are

- Not role symmetric in the Type 4 setting.
- Has all required security properties
  - Bar the strictest form of forward secrecy.
  - Wang protocol is better than CK in this respect.
- CK is secure under the GBDH problem.
- Wang is secure under the DBDH problem.
- Has poor bandwidth efficiency in the Type 4 setting for one party.
- Subgroup membership testing in the Type 4 setting is harder for one party.

SCK Protocol

We really want a protocol which

- Meets all of our security goals
  - Including strong forms of forward secrecy.
- Has a proof of security relative to a standard, i.e. non-gap problem.
- Is role symmetric.
- Is efficient.

Turns out a minor modification of Smart's original protocol meets these requirements.

- Modification proposed by Chen and Kudla.

SCK Protocol

Smart/Chen/Kudla Protocol

Use Extract 1 method for key extraction.

Alice and Bob randomly choose x and y from $\mathbb{Z}_q^*$ and perform the protocol as follows:

$$\begin{aligned} A \to B &: E_A = xP_2, \\ B \to A &: E_B = yP_2. \end{aligned}$$

The shared secret key is

$$xyP_2 \| \hat{e}(xQ_B + yQ_A, P_2)^s.$$

- Alice computes this via $xE_B \| \hat{e}(xQ_B, R') \cdot \hat{e}(d_A, E_B)$.
- Bob computes this via $yE_A \| \hat{e}(yQ_A, R') \cdot \hat{e}(d_B, E_A)$.

We have only added in the Diffie-Hellman secret to the KDF.

SCK Protocol

The SCK protocol is

- Implementable in all pairing parameter types.
- Role symmetric.
- Has all required security properties
  - Including strict forms of forward secrecy.
- Provably securely under the BDH assumption (Chen/Cheng/Smart)

If using the Extract 1' method:

- Can not be implemented in parameter Type 2.
- Need to replace the message flows by xP_1 and yP_1.
- Obtain shorter message flows in this case.
- Detecting subgroup membership for Type 3 and 4 pairings is then easier.

Outline

Conclusion

We have looked at a set of ID-based key agreement protocols.

Whether one can implement a given protocol depends on what type of pairing parameters we are using.

The only protocol which currently meets all security requirements and has a proof of security relative to a standard hard problem is the SCK protocol.

See the full paper for proofs and a more extensive discussion of the protocols in this talk and others in the literature.