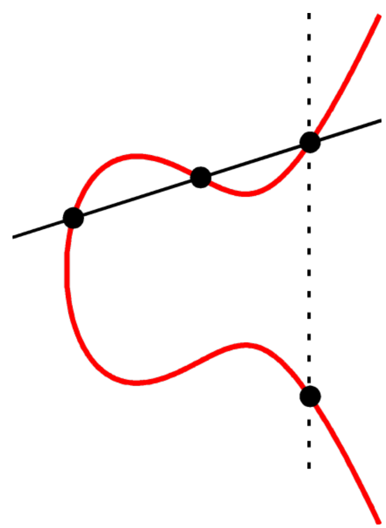
An Introduction to Pairing Based Cryptography



Dustin Moody

October 31, 2008

Definitions

Let G_1 and G_2 be abelian groups, written additively.

Definitions

Let G_1 and G_2 be abelian groups, written additively.

Let n be a prime number such that $[n]P$ for all P in G_1 and G_2 .

Definitions

Let G_1 and G_2 be abelian groups, written additively.

Let n be a prime number such that $[n]P$ for all P in G_1 and G_2 .

Let G_3 be a cyclic group of order n, written multiplicatively.

$$e: \; G_1 \times G_2 \rightarrow G_3$$

Definitions

Let G_1 and G_2 be abelian groups, written additively.

Let n be a prime number such that $[n]P$ for all P in G_1 and G_2 .

Let G_3 be a cyclic group of order n, written multiplicatively.

Then a pairing is a map:

$$e: \; G_1 \times G_2 \to G_3$$

$$e(P + P', Q) = e(P, Q)e(P', Q) \quad \text{for all } P, P' \in G_1, Q \in G_2$$

$$e(P, Q + Q') = e(P, Q)e(P, Q') \quad \text{for all } P \in G_1, \; Q, Q' \in G_2$$

Definitions

Let G_1 and G_2 be abelian groups, written additively.

Let n be a prime number such that $[n]P$ for all P in G_1 and G_2 .

Let G_3 be a cyclic group of order n, written multiplicatively.

Then a pairing is a map:

$$e: \ G_1 \times G_2 \rightarrow G_3$$

(*Bilinearity*)

$$e(P+P',Q) = e(P,Q)e(P',Q) \quad \text{for all } P, P' \in G_1, Q \in G_2$$

$$e(P,Q+Q') = e(P,Q)e(P,Q') \quad \text{for all } P \in G_1, \ Q, Q' \in G_2$$

(*Non-Degeneracy*)

For all non-zero $P \in G_1$, there is a $Q \in G_2$ such that $e(P,Q) \neq 1$.

For all non-zero $Q \in G_2$, there is a $P \in G_1$ such that $e(P,Q) \neq 1$.

Properties of Bilinear Pairings

Properties of Bilinear Pairings

1) $e(P,0) = e(0,Q) = 1$

Properties of Bilinear Pairings

1) $e(P,0) = e(0,Q) = 1$

2) $e(-P,Q) = e(P,Q)^{-1} = e(P,-Q)$

Properties of Bilinear Pairings

1) $e(P,0) = e(0,Q) = 1$

2) $e(-P,Q) = e(P,Q)^{-1} = e(P,-Q)$

3) $e([a]P,Q) = e(P,Q)^{a} = e(P,[a]Q)$ for all $a \in \mathbf{Z}$

Properties of Bilinear Pairings

1) $e(P,0) = e(0,Q) = 1$

2) $e(-P,Q) = e(P,Q)^{-1} = e(P,-Q)$

3) $e([a]P,Q) = e(P,Q)^{a} = e(P,[a]Q)$ for all $a \in \mathbf{Z}$

4) $e([a]P,[b]Q) = e(P,Q)^{ab}$ for all $a,b \in \mathbf{Z}$

Pairings on Elliptic Curves

For our purposes, we will use the following:

Pairings on Elliptic Curves

For our purposes, we will use the following:

Let E be an elliptic curve defined over F_{q}.

Pairings on Elliptic Curves

For our purposes, we will use the following:

Let E be an elliptic curve defined over $F_{q.}$

Let P be a fixed point on E of prime order n.

Pairings on Elliptic Curves

For our purposes, we will use the following:

Let E be an elliptic curve defined over F_q.

Let P be a fixed point on E of prime order n.

Let k be the order of q mod n.

Then k is also the smallest integer such that $n \mid (q^k - 1)$.

Pairings on Elliptic Curves

For our purposes, we will use the following:

Let E be an elliptic curve defined over F_q.

Let P be a fixed point on E of prime order n.

Let k be the order of q mod n.

Then k is also the smallest integer such that $n \mid (q^k - 1)$.

k is called the *embedding degree*.

Pairings will be of the form

$$e : \langle P \rangle \times \langle P \rangle \text{®}\ \mu_n \subseteq F_{q^k}{}^{*}$$

Pairings on Elliptic Curves

For our purposes, we will use the following:

Let E be an elliptic curve defined over $F_{q.}$

Let P be a fixed point on E of prime order n.

Let k be the order of q mod n.

Then k is also the smallest integer such that $n \mid (q^{k} - 1)$.

k is called the *embedding degree*.

Pairings will be of the form

$$e : \langle P \rangle \times \langle P \rangle \text{ ® } \mu_n \subseteq F_{q^k}{}^{*}$$

We'll also require that $e(P,P) \neq 1$, which can be done using distortion maps.

Computability and the Embedding Degree

Computability and the Embedding Degree

The Weil pairing and Tate pairing are examples of pairings on curves.

Computability and the Embedding Degree

The Weil pairing and Tate pairing are examples of pairings on curves.

Both are efficiently computable, provided that the embedding degree k is small.

Computability and the Embedding Degree

The Weil pairing and Tate pairing are examples of pairings on curves.

Both are efficiently computable, provided that the embedding degree k is small.

For a random elliptic curve, $k \approx n$, which is too large.

Computability and the Embedding Degree

The Weil pairing and Tate pairing are examples of pairings on curves.

Both are efficiently computable, provided that the embedding degree k is small.

For a random elliptic curve, $k \approx n$, which is too large.

<u>Theorem</u>: If E is a supersingular elliptic curve, then $k \leq 6$.

(Recall E is supersingular if $\#E(F_{p^r}) \equiv 1 \bmod p$.)

Computability and the Embedding Degree

The Weil pairing and Tate pairing are examples of pairings on curves.

Both are efficiently computable, provided that the embedding degree k is small.

For a random elliptic curve, $k \approx n$, which is too large.

Theorem: If E is a supersingular elliptic curve, then $k \leq 6$.

(Recall E is supersingular if $\#E(F_{p^r}) \equiv 1 \bmod p$.)

There are ordinary curves with low embedding degree

(MNT curves have k = 2,3, or 4.)

Cryptographic Applications

Cryptographic Applications

- MOV attack- Transfers the discrete logarithm problem on *E* to a discrete logarithm in F_{q^k}.

Cryptographic Applications

- MOV attack- Transfers the discrete logarithm problem on *E* to a discrete logarithm in F_{qk}.
- Separating DDH from DH- Pairings can be used to show the Decision Diffie-Hellman problem is easier than the Diffie-Hellman problem on some curves.

Cryptographic Applications

- MOV attack- Transfers the discrete logarithm problem on *E* to a discrete logarithm in F_{qk}.
- Separating DDH from DH- Pairings can be used to show the Decision Diffie-Hellman problem is easier than the Diffie- Hellman problem on some curves.
- Identity based encryption- Public key encryption system where the users public key is based on his own identity, i.e. an email address.

Cryptographic Applications

- MOV attack- Transfers the discrete logarithm problem on *E* to a discrete logarithm in F_{qk}.
- Separating DDH from DH- Pairings can be used to show the Decision Diffie-Hellman problem is easier than the Diffie- Hellman problem on some curves.
- Identity based encryption- Public key encryption system where the users public key is based on his own identity, i.e. an email address.
- Short signatures- Signature schemes with signatures half the length of other signature schemes.

Cryptographic Applications

- MOV attack- Transfers the discrete logarithm problem on *E* to a discrete logarithm in F_{qk}.
- Separating DDH from DH- Pairings can be used to show the Decision Diffie-Hellman problem is easier than the Diffie- Hellman problem on some curves.
- Identity based encryption- Public key encryption system where the users public key is based on his own identity, i.e. an email address.
- Short signatures- Signature schemes with signatures half the length of other signature schemes.
- Key exchange- A tripartite key exchange can be done in one round.

Cryptographic Applications

•MOV attack- Transfers the discrete logarithm problem on *E* to a discrete logarithm in F_{qk}.

•Separating DDH from DH- Pairings can be used to show the Decision Diffie-Hellman problem is easier than the Diffie- Hellman problem on some curves.

•Identity based encryption- Public key encryption system where the users public key is based on his own identity, i.e. an email address.

•Short signatures- Signature schemes with signatures half the length of other signature schemes.

•Key exchange- A tripartite key exchange can be done in one round.

•Group structure of *E*- can be determined efficiently using pairings.

Cryptographic Applications

- MOV attack- Transfers the discrete logarithm problem on *E* to a discrete logarithm in F_{qk}.
- Separating DDH from DH- Pairings can be used to show the Decision Diffie-Hellman problem is easier than the Diffie- Hellman problem on some curves.
- Identity based encryption- Public key encryption system where the users public key is based on his own identity, i.e. an email address.
- Short signatures- Signature schemes with signatures half the length of other signature schemes.
- Key exchange- A tripartite key exchange can be done in one round.
- Group structure of *E*- can be determined efficiently using pairings.
- Identity based signatures, Identity based key exchange……..

The MOV attack

1993 (Menezes, Okamoto, and Vanstone)

The MOV attack

1993 (Menezes, Okamoto, and Vanstone)

Discrete Log (DL) Problem: Given P and $Q = [c]P$ on E, find c.

The MOV attack

1993 (Menezes, Okamoto, and Vanstone)

Discrete Log (DL) Problem: Given P and $Q = [c]P$ on E, find c.

The attack:

The MOV attack

1993 (Menezes, Okamoto, and Vanstone)

Discrete Log (DL) Problem: Given P and $Q = [c]P$ on E, find c.

The attack:

1) Find S of order n on E, such that $e(P,S) \neq 1$.

The MOV attack

1993 (Menezes, Okamoto, and Vanstone)

Discrete Log (DL) Problem: Given P and $Q = [c]P$ on E, find c.

The attack:

1) Find S of order n on E, such that $e(P,S) \neq 1$.
2) Compute $e(P,S) = \zeta$.

The MOV attack

1993 (Menezes, Okamoto, and Vanstone)

Discrete Log (DL) Problem: Given P and $Q = [c]P$ on E, find c.

The attack:

1) Find S of order n on E, such that $e(P,S) \neq 1$.
2) Compute $e(P,S) = \zeta$.
3) Compute $e(Q,S) = e([c]P,S) = e(P,S)^c = \zeta^c$.

The MOV attack

1993 (Menezes, Okamoto, and Vanstone)

Discrete Log (DL) Problem: Given P and $Q = [c]P$ on E, find c.

The attack:

1) Find S of order n on E, such that $e(P,S) \neq 1$.
2) Compute $e(P,S) = \zeta$.
3) Compute $e(Q,S) = e([c]P,S) = e(P,S)^c = \zeta^c$.
4) Solve the DL in F_{qk} with ζ and ζ^c.

Best algorithms for solving DL on elliptic curves is $O(\sqrt{n})$.

The MOV attack

1993 (Menezes, Okamoto, and Vanstone)

Discrete Log (DL) Problem: Given P and $Q = [c]P$ on E, find c.

The attack:

1) Find S of order n on E, such that $e(P,S) \neq 1$.
2) Compute $e(P,S) = \zeta$.
3) Compute $e(Q,S) = e([c]P,S) = e(P,S)^c = \zeta^c$.
4) Solve the DL in F_{q^k} with ζ and ζ^c.

Best algorithms for solving DL on elliptic curves is $O(\sqrt{n})$.

In F_{q^k}, there are subexponential methods (index calculus).

Note, the attack is only efficient for small k.

Key Exchanges

Key Exchanges

Diffie-Hellman Key exchange:

Key Exchanges

Diffie-Hellman Key exchange:

1) Alice selects secret a, sends $[a]P$ to Bob.

Key Exchanges

Diffie-Hellman Key exchange:

1) Alice selects secret a, sends $[a]P$ to Bob.
2) Bob selects secret b, sends $[b]P$ to Alice.

Key Exchanges

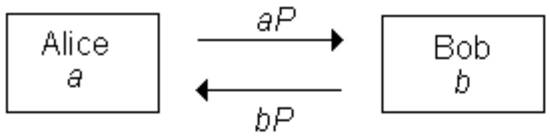
Diffie-Hellman Key exchange:

1) Alice selects secret a, sends $[a]P$ to Bob.
2) Bob selects secret b, sends $[b]P$ to Alice.
3) Each can compute the key $[ab]P$

Key Exchanges

Diffie-Hellman Key exchange:

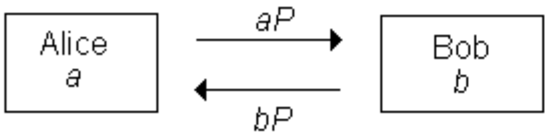
1) Alice selects secret *a*, sends [*a*]*P* to Bob.
2) Bob selects secret *b*, sends [*b*]*P* to Alice.
3) Each can compute the key [*ab*]*P*



Key Exchanges

Diffie-Hellman Key exchange:

1) Alice selects secret a, sends $[a]P$ to Bob.
2) Bob selects secret b, sends $[b]P$ to Alice.
3) Each can compute the key $[ab]P$

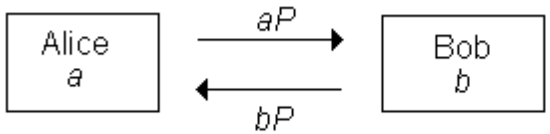


Extend to three parties?

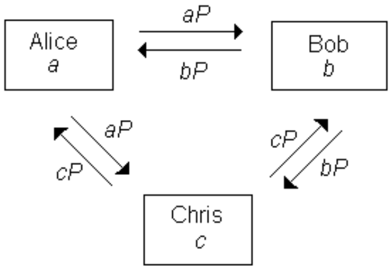
Key Exchanges

Diffie-Hellman Key exchange:

1) Alice selects secret a, sends $[a]P$ to Bob.
2) Bob selects secret b, sends $[b]P$ to Alice.
3) Each can compute the key $[ab]P$



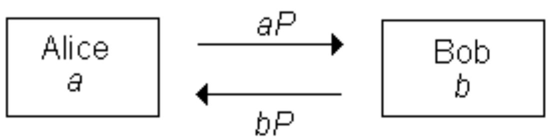
Extend to three parties?



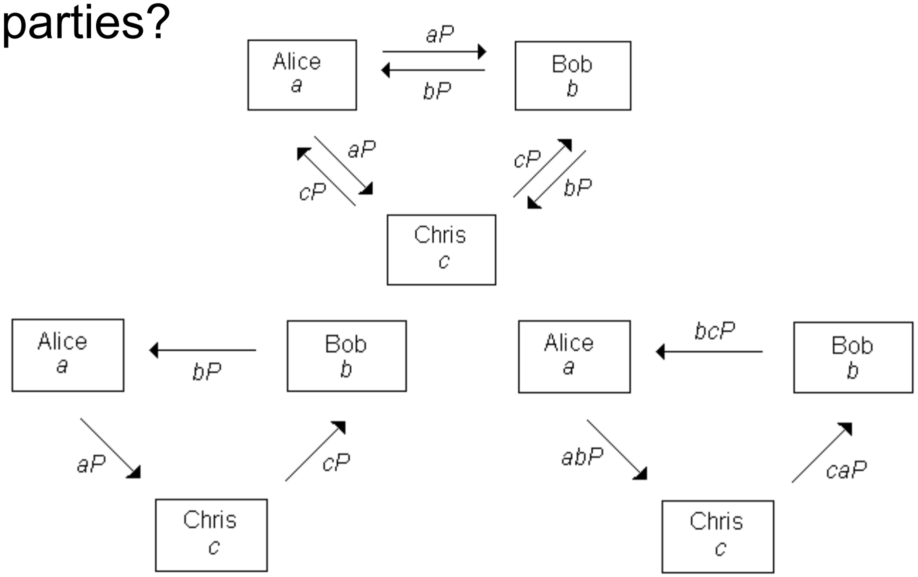
Key Exchanges

Diffie-Hellman Key exchange:

1) Alice selects secret a, sends $[a]P$ to Bob.
2) Bob selects secret b, sends $[b]P$ to Alice.
3) Each can compute the key $[ab]P$



Extend to three parties?



One Round Tripartite Key Exchange

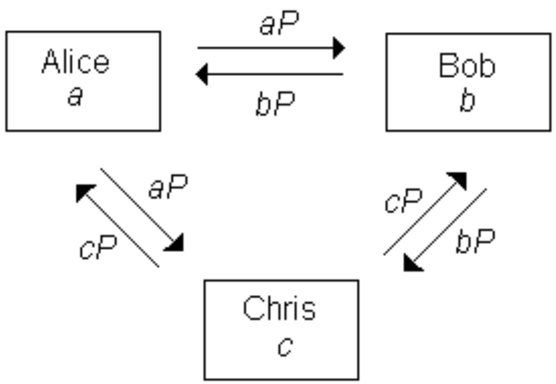
2000 (Joux)

1) Alice sends $[a]P$ to Bob and Chris
2) Bob sends $[b]P$ to Alice and Chris
3) Chris sends $[c]P$ to Alice and Bob
4) All can compute the key $e(P,P)^{abc}$.

(For example, Alice computes $e([b]P,[c]P)^{a}$.)

One Round Tripartite Key Exchange

2000 (Joux)

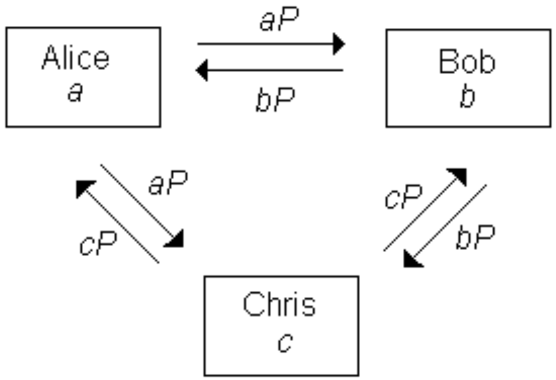


1) Alice sends $[a]P$ to Bob and Chris
2) Bob sends $[b]P$ to Alice and Chris
3) Chris sends $[c]P$ to Alice and Bob
4) All can compute the key $e(P,P)^{abc}$.

(For example, Alice computes $e([b]P,[c]P)^{a}$.)

One Round Tripartite Key Exchange

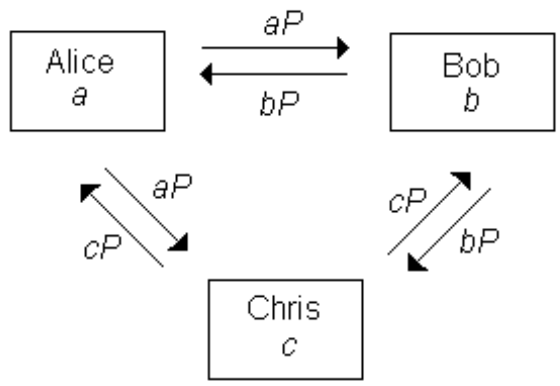
2000 (Joux)



1) Alice sends $[a]P$ to Bob and Chris
2) Bob sends $[b]P$ to Alice and Chris

One Round Tripartite Key Exchange

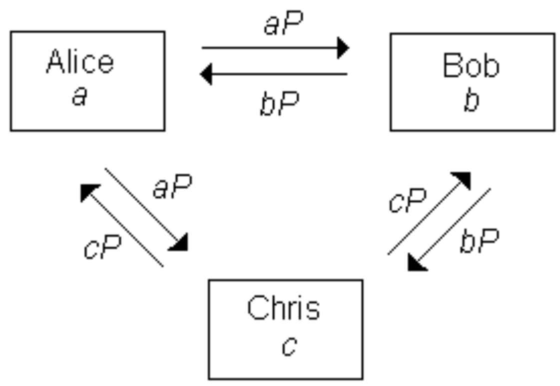
2000 (Joux)



1) Alice sends $[a]P$ to Bob and Chris
2) Bob sends $[b]P$ to Alice and Chris
3) Chris sends $[c]P$ to Alice and Bob

One Round Tripartite Key Exchange

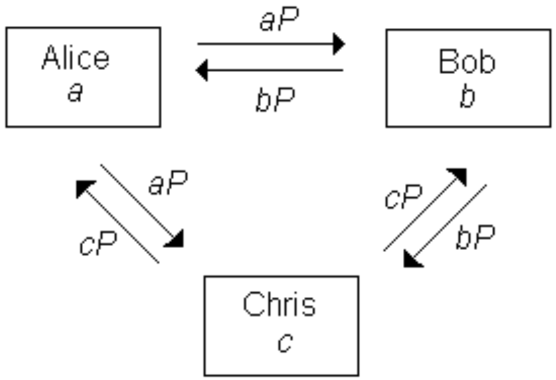
2000 (Joux)



1) Alice sends $[a]P$ to Bob and Chris
2) Bob sends $[b]P$ to Alice and Chris
3) Chris sends $[c]P$ to Alice and Bob
4) All can compute the key $e(P,P)^{abc}$.

One Round Tripartite Key Exchange

2000 (Joux)



1) Alice sends $[a]P$ to Bob and Chris
2) Bob sends $[b]P$ to Alice and Chris
3) Chris sends $[c]P$ to Alice and Bob
4) All can compute the key $e(P,P)^{abc}$.

(For example, Alice computes $e([b]P,[c]P)^a$.)

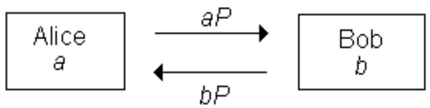
Security

For the Diffie-Hellman key exchange, security is based on the (DH) problem:

Security

For the Diffie-Hellman key exchange, security is based on the (DH) problem:

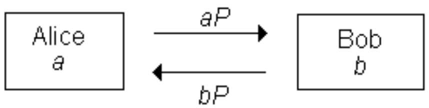
Given P, $[a]P$, and $[b]P$, compute $[ab]P$.



Security

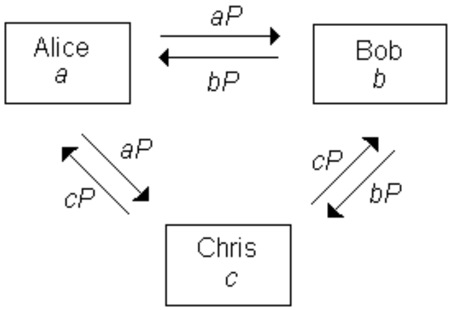
For the Diffie-Hellman key exchange, security is based on the (DH) problem:

Given P, $[a]P$, and $[b]P$, compute $[ab]P$.



For Joux's tripartite exchange, security is based on the Bilinear Diffie-Hellman (BDH) problem:

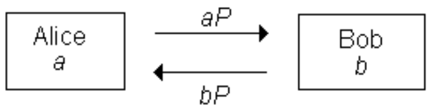
Given P, $[a]P$, $[b]P$, and $[c]P$, compute $e(P,P)^{abc}$.



Security

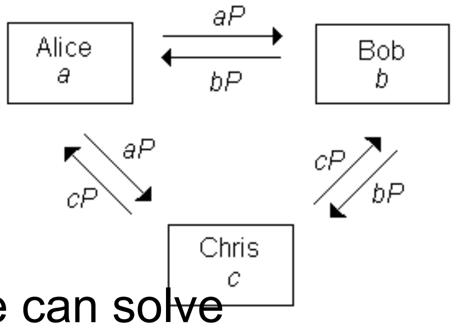
For the Diffie-Hellman key exchange, security is based on the (DH) problem:

Given P, $[a]P$, and $[b]P$, compute $[ab]P$.



For Joux's tripartite exchange, security is based on the Bilinear Diffie-Hellman (BDH) problem:

Given P, $[a]P$, $[b]P$, and $[c]P$, compute $e(P,P)^{abc}$.

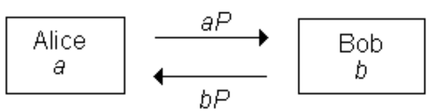


Clearly, if one can solve the discrete log problem, then one can solve the Diffie-Hellman problem.

Security

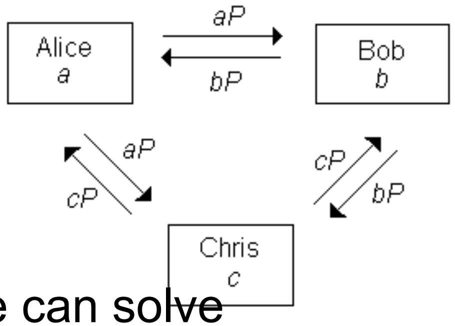
For the Diffie-Hellman key exchange, security is based on the (DH) problem:

Given P, $[a]P$, and $[b]P$, compute $[ab]P$.



For Joux's tripartite exchange, security is based on the Bilinear Diffie-Hellman (BDH) problem:

Given P, $[a]P$, $[b]P$, and $[c]P$, compute $e(P,P)^{abc}$.



Clearly, if one can solve the discrete log problem, then one can solve the Diffie-Hellman problem.

Anyone who can solve the Diffie-Hellman problem can solve the bilinear Diffie-Hellman problem.

Separating DH and DDH

Separating DH and DDH

The decision Diffie-Hellman (DDH) problem is:

Given P, $[a]P$, $b[P]$, and Q, determine if $Q = [ab]P$.

Separating DH and DDH

The decision Diffie-Hellman (DDH) problem is:

Given P, $[a]P$, $b[P]$, and Q, determine if $Q = [ab]P$.

The DDH problem is no harder than the DH problem.

Separating DH and DDH

The decision Diffie-Hellman (DDH) problem is:

Given P, $[a]P$, $b[P]$, and Q, determine if $Q = [ab]P$.

The DDH problem is no harder than the DH problem.

For awhile, there were no examples of groups where the DDH was strictly easier than the DH. Such groups are called “gap Diffie-Hellman groups”.

Separating DH and DDH

The decision Diffie-Hellman (DDH) problem is:

Given P, $[a]P$, $b[P]$, and Q, determine if $Q = [ab]P$.

The DDH problem is no harder than the DH problem.

For awhile, there were no examples of groups where the DDH was strictly easier than the DH. Such groups are called "gap Diffie-Hellman groups".

Pairings make the DDH problem "easy":

Separating DH and DDH

The decision Diffie-Hellman (DDH) problem is:

Given P, $[a]P$, $b[P]$, and Q, determine if $Q = [ab]P$.

The DDH problem is no harder than the DH problem.

For awhile, there were no examples of groups where the DDH was strictly easier than the DH. Such groups are called “gap Diffie-Hellman groups”.

Pairings make the DDH problem “easy”:

1) Compute $e(P,Q)$

Separating DH and DDH

The decision Diffie-Hellman (DDH) problem is:

Given P, $[a]P$, $b[P]$, and Q, determine if $Q = [ab]P$.

The DDH problem is no harder than the DH problem.

For awhile, there were no examples of groups where the DDH was strictly easier than the DH. Such groups are called “gap Diffie-Hellman groups”.

Pairings make the DDH problem “easy”:

1) Compute $e(P,Q)$

2) Compute $e([a]P,[b]P)=e(P,P)^{ab}$

Separating DH and DDH

The decision Diffie-Hellman (DDH) problem is:

Given P, $[a]P$, $b[P]$, and Q, determine if $Q = [ab]P$.

The DDH problem is no harder than the DH problem.

For awhile, there were no examples of groups where the DDH was strictly easier than the DH. Such groups are called “gap Diffie-Hellman groups”.

Pairings make the DDH problem “easy”:

1) Compute $e(P,Q)$

2) Compute $e([a]P,[b]P)=e(P,P)^{ab}$

3) $Q=[ab]P$ if and only if the above two results agree.

Separating DH and DDH

The decision Diffie-Hellman (DDH) problem is:

Given P, $[a]P$, $b[P]$, and Q, determine if $Q = [ab]P$.

The DDH problem is no harder than the DH problem.

For awhile, there were no examples of groups where the DDH was strictly easier than the DH. Such groups are called “gap Diffie-Hellman groups”.

Pairings make the DDH problem “easy”:

1) Compute $e(P,Q)$

2) Compute $e([a]P,[b]P)=e(P,P)^{ab}$

3) $Q=[ab]P$ if and only if the above two results agree.

Thus, elliptic curves with small k are gap Diffie-Hellman groups.

(Actually, the curve needs a *distortion map* so that $e(P,P) \neq 1$.)

Short Signatures

2001 (Boneh, Lynn, and Schacham)

Parameters: E,e,P, and a hash function

Short Signatures

2001 (Boneh, Lynn, and Schacham)

Parameters: *E,e,P*, and a hash function $H_1 : \{0,1\}^m$ ® $\langle P \rangle$

Setup: Private key is a secret integer *r.*

Short Signatures

2001 (Boneh, Lynn, and Schacham)

Parameters: *E,e,P*, and a hash function $H_1 : \{0,1\}^m$ ® $\langle P \rangle$

Setup: Private key is a secret integer *r.*
Public key is $R = [r]P$.

Short Signatures

2001 (Boneh, Lynn, and Schacham)

Parameters: *E,e,P*, and a hash function $H_1 : \{0,1\}^m \rightarrow \langle P \rangle$

Setup: Private key is a secret integer *r.*
Public key is *R* = [*r*]*P*.

To sign a message M, Alice computes $S = [r]H_1(M)$.

To verify the signature, check if $e(P, S) = e(R, H_1(M))$.

Short Signatures

2001 (Boneh, Lynn, and Schacham)

Parameters: *E,e,P*, and a hash function $H_1 : \{0,1\}^m$ ® $\langle P \rangle$

Setup: Private key is a secret integer *r.*
Public key is $R = [r]P$.

To sign a message M, Alice computes $S = [r]H_1(M)$.

To verify the signature, check if $e(P, S) = e(R, H_1(M))$.

$$e(P, S) = e(P, [r]H_1(M)) = e([r]P, H_1(M)) = e(R, H_1(M)).$$

To forge a signature on M, need to be able to find $S = [r]H_1(M)$, given P,R, and $H_1(M)$, which is a Diffie-Hellman problem in $< P >$.

Identity Based Encryption

Identity Based Encryption

Using public key encryption, when Bob wants to send a message to Alice, he encrypts using Alice's public key K_A.

Identity Based Encryption

Using public key encryption, when Bob wants to send a message to Alice, he encrypts using Alice's public key K_A.

Suppose the malicious Mallory substitutes her own public key K_M for Alice's. How does Bob know the key isn't really Alice's?

Identity Based Encryption

Using public key encryption, when Bob wants to send a message to Alice, he encrypts using Alice's public key K_A.

Suppose the malicious Mallory substitutes her own public key K_M for Alice's. How does Bob know the key isn't really Alice's?

One solution is to require a trusted authority (TA) to give certificates for public keys. Such a certificate has Alice's ID and public key on it, signed by the TA. Bob can check the trusted authority's signature on the certificate, and be assured of what Alice's public key is.

Identity Based Encryption

Using public key encryption, when Bob wants to send a message to Alice, he encrypts using Alice's public key K_A.

Suppose the malicious Mallory substitutes her own public key K_M for Alice's. How does Bob know the key isn't really Alice's?

One solution is to require a trusted authority (TA) to give certificates for public keys. Such a certificate has Alice's ID and public key on it, signed by the TA. Bob can check the trusted authority's signature on the certificate, and be assured of what Alice's public key is.

1984- Shamir proposed using Alice's ID information as her public key. (Such a key could be an email address, for example.)

Identity Based Encryption

Using public key encryption, when Bob wants to send a message to Alice, he encrypts using Alice's public key K_A.

Suppose the malicious Mallory substitutes her own public key K_M for Alice's. How does Bob know the key isn't really Alice's?

One solution is to require a trusted authority (TA) to give certificates for public keys. Such a certificate has Alice's ID and public key on it, signed by the TA. Bob can check the trusted authority's signature on the certificate, and be assured of what Alice's public key is.

1984- Shamir proposed using Alice's ID information as her public key. (Such a key could be an email address, for example.)

Bob then knows for sure who he is sending his message to.

Identity Based Encryption

Identity Based Encryption

2001 (Boneh and Franklin)

Parameters: *E*, *P*, *e*, and two hash functions:

$$H_1 : \{0,1\}^m \text{ ® } < P >$$

$$H_2 : \mathrm{F}_{q^k} \text{ ® } \{0,1\}^m$$

Identity Based Encryption

2001 (Boneh and Franklin)

Parameters: *E*, *P*, *e*, and two hash functions:

$$H_1 : \{0,1\}^m \rightarrow \langle P \rangle$$

$$H_2 : \mathbb{F}_{q^k} \rightarrow \{0,1\}^m$$

Setup: Alice's public key is $K_A = H_1(ID_A)$.

The TA has private key *s*, and public key $S=[s]P$.

TA gives Alice her secret decryption key $D_A = [s]K_A$.

Identity Based Encryption

2001 (Boneh and Franklin)

Parameters: *E*, *P*, *e*, and two hash functions: $H_1 : \{0,1\}^m$ ® $< P >$

$H_2 : \mathrm{F}_{q^k}$ ® $\{0,1\}^m$

Setup: Alice's public key is $K_A = H_1(ID_A)$.

The TA has private key *s*, and public key $S=[s]P$.

TA gives Alice her secret decryption key $D_A = [s]K_A$.

Encryption: To send *M*, Bob selects a random *r* and computes $R = [r]P$ and $c = M \oplus H_2(e(K_A, S)^r)$. He sends Alice (R, c).

Identity Based Encryption

2001 (Boneh and Franklin)

Parameters: *E*, *P*, *e*, and two hash functions:

$$H_1 : \{0,1\}^m \text{ ® } < P >$$

$$H_2 : \mathrm{F}_{q^k} \text{ ® } \{0,1\}^m$$

Setup: Alice's public key is $K_A = H_1(ID_A)$.
The TA has private key *s*, and public key $S=[s]P$.
TA gives Alice her secret decryption key $D_A = [s]K_A$.

Encryption: To send *M*, Bob selects a random *r* and computes
$R = [r]P$ and $c = M \oplus H_2(e(K_A, S)^r)$. He sends Alice (R, c).

Decryption: Alice uses her private key D_A to calculate

$$c \oplus H_2(e(D_A,R)) = c \oplus H_2(e([s]K_A, [r]P)) = c \oplus H_2(e(K_A, S)^r) = M.$$

Identity Based Encryption

2001 (Boneh and Franklin)

Parameters: *E*, *P*, *e*, and two hash functions:

$$H_1 : \{0,1\}^m \circledR \; < P >$$

$$H_2 : \mathrm{F}_{q^k} \circledR \; \{0,1\}^m$$

Setup: Alice's public key is $K_A = H_1(ID_A)$.

The TA has private key *s*, and public key $S=[s]P$.

TA gives Alice her secret decryption key $D_A = [s]K_A$.

Encryption: To send *M*, Bob selects a random *r* and computes $R = [r]P$ and $c = M \oplus H_2(e(K_A, S)^r)$. He sends Alice (R, c).

Decryption: Alice uses her private key D_A to calculate

$$c \oplus H_2(e(D_A, R)) = c \oplus H_2(e([s]K_A, [r]P)) = c \oplus H_2(e(K_A, S)^r) = M.$$

Anyone other than Alice wishing to decrypt the message from (R, c) needs to be able to compute $e(K_A, S)^r = e(K_A, P)^{rs}$ given P, K_A, S, and R. This requires solving the bilinear Diffie-Hellman problem.

Computing Pairings

Computing Pairings

Miller's algorithm to evaluate $< P, Q >_n$

Computing Pairings

Miller's algorithm to evaluate $< P,Q >_n$

1. Given P,Q with order n, choose R with order n and $R \neq \infty, P, -Q, P-Q$.

Computing Pairings

Miller's algorithm to evaluate $< P, Q >_n$

1. Given P, Q with order n, choose R with order n and $R \neq \infty, P, -Q, P-Q$.
2. Write n in binary as $n = (n_t, ..., n_1, n_0)$.

Computing Pairings

Miller's algorithm to evaluate $< P,Q >_n$

1. Given P,Q with order n, choose R with order n and $R \neq \infty, P, -Q, P-Q$.
2. Write n in binary as $n = (n_t, ..., n_1, n_0)$.
3. Set $f = 1$, $T = P$ and $i = t$.

Computing Pairings

Miller's algorithm to evaluate $< P,Q >_n$

1. Given P,Q with order n, choose R with order n and $R \neq \infty, P, -Q, P-Q$.
2. Write n in binary as $n = (n_t, ..., n_1, n_0)$.
3. Set $f = 1$, $T = P$ and $i = t$.
4. If $i < 0$ then go to step 5. Else do the following:
   (a) Let l be the tangent line to E through T. Let v be the vertical line through $2T$.
   (b) Set $T = 2T$.
   (c) Set $f = f^2 \dfrac{l(Q+R)v(R)}{v(Q+R)l(R)}$

   (d) If $n_i = 1$ then do the following:
      i. Let l be the line through T and P, and v the vertical line through $T + P$.
      ii. Set $T = T + P$.
      iii. Set $f = f \dfrac{l(Q+R)v(R)}{v(Q+R)l(R)}$

   (e) Set $i = i - 1$ and return to step 4

Computing Pairings

Miller's algorithm to evaluate $< P,Q >_n$

1. Given P,Q with order n, choose R with order n and $R \neq \infty, P, -Q, P-Q$.
2. Write n in binary as $n = (n_t, ..., n_1, n_0)$.
3. Set $f = 1$, $T = P$ and $i = t$.
4. If $i < 0$ then go to step 5. Else do the following:
   - (a) Let l be the tangent line to E through T. Let v be the vertical line through $2T$.
   - (b) Set $T = 2T$.
   - (c) Set $f = f^2 \dfrac{l(Q+R)v(R)}{v(Q+R)l(R)}$
   - (d) If $n_i = 1$ then do the following:
     - i. Let l be the line through T and P, and v the vertical line through $T + P$.
     - ii. Set $T = T + P$.
     - iii. Set $f = f \dfrac{l(Q+R)v(R)}{v(Q+R)l(R)}$
   - (e) Set $i = i - 1$ and return to step 4
5. The desired value is $< P,Q >_n = f$.

Verheul's Theorem

2001 (Verheul)

XTR: Let p be a prime $p \equiv 2 \bmod 3$ and n a prime number such that $n \mid p^2+p+1$. Let g be a generator of μ_n, the group of nth roots of unity in . Let P be a point of order n on a supersingular E defined over with $\#E( ) = p^2+p+1$.

Theorem: If an efficiently computable homomorphism can be found from μ_n to $\langle P \rangle$, then the Diffie-Hellman problem can be efficiently solved in both μ_n and $\langle P \rangle$.

What are the implications?

My dissertation generalizes Verheul's theorem.

Verheul's Theorem

2001 (Verheul)

XTR: Let p be a prime $p \equiv 2 \bmod 3$ and n a prime number such that $n \mid p^2+p+1$. Let g be a generator of μ_n, the group of nth roots of unity in F_{p^6}. Let P be a point of order n on a supersingular E defined over F_{p^2} with $\#E(F_{p^2}) = p^2+p+1$.

Theorem: If an efficiently computable homomorphism can be found from μ_n to $\langle P \rangle$, then the Diffie-Hellman problem can be efficiently solved in both μ_n and $\langle P \rangle$.

What are the implications?

My dissertation generalizes Verheul's theorem.

Verheul's Theorem

2001 (Verheul)

XTR: Let p be a prime $p \equiv 2 \bmod 3$ and n a prime number such that $n \mid p^2+p+1$. Let g be a generator of μ_n, the group of nth roots of unity in F_{p^6}. Let P be a point of order n on a supersingular E defined over F_{p^2} with $\#E(F_{p^2}) = p^2+p+1$.

Theorem: If an efficiently computable homomorphism can be found from μ_n to $<P>$, then the Diffie-Hellman problem can be efficiently solved in both μ_n and $<P>$.

Verheul's Theorem

2001 (Verheul)

XTR: Let p be a prime $p \equiv 2 \bmod 3$ and n a prime number such that $n \mid p^2+p+1$. Let g be a generator of μ_n, the group of nth roots of unity in F_{p^6}. Let P be a point of order n on a supersingular E defined over F_{p^2} with $\#E(F_{p^2}) = p^2+p+1$.

Theorem: If an efficiently computable homomorphism can be found from μ_n to $<P>$, then the Diffie-Hellman problem can be efficiently solved in both μ_n and $<P>$.

What are the implications?

My dissertation generalizes Verheul's theorem.

Conclusion

Conclusion

- Active area of research (see Barreto's Pairing Based Crypto Lounge or eprint.iacr.org)

Conclusion

- Active area of research (see Barreto's Pairing Based Crypto Lounge or eprint.iacr.org)
- Many interesting/simpler protocols

Conclusion

- Active area of research (see Barreto's Pairing Based Crypto Lounge or eprint.iacr.org)
- Many interesting/simpler protocols
- Not quite yet in SAGE

Conclusion

- Active area of research (see Barreto’s Pairing Based Crypto Lounge or eprint.iacr.org)
- Many interesting/simpler protocols
- Not quite yet in SAGE
- Already commercially available -- Voltage

Conclusion

- Active area of research (see Barreto’s Pairing Based Crypto Lounge or eprint.iacr.org)
- Many interesting/simpler protocols
- Not quite yet in SAGE
- Already commercially available -- Voltage
- Security needs to be studied and tested

Conclusion

- Active area of research (see Barreto's Pairing Based Crypto Lounge or eprint.iacr.org)
- Many interesting/simpler protocols
- Not quite yet in SAGE
- Already commercially available -- Voltage
- Security needs to be studied and tested
- Other pairings (Ate, Eta, Eil, etc…)

Conclusion

- Active area of research (see Barreto's Pairing Based Crypto Lounge or eprint.iacr.org)
- Many interesting/simpler protocols
- Not quite yet in SAGE
- Already commercially available -- Voltage
- Security needs to be studied and tested
- Other pairings (Ate, Eta, Eil, etc…)
- Pairings for hyperelliptic curves, abelian varieties

Conclusion

- Active area of research (see Barreto's Pairing Based Crypto Lounge or eprint.iacr.org)
- Many interesting/simpler protocols
- Not quite yet in SAGE
- Already commercially available -- Voltage
- Security needs to be studied and tested
- Other pairings (Ate, Eta, Eil, etc…)
- Pairings for hyperelliptic curves, abelian varieties
- etc.…

Conclusion

- Active area of research (see Barreto's Pairing Based Crypto Lounge or eprint.iacr.org)
- Many interesting/simpler protocols
- Not quite yet in SAGE
- Already commercially available -- Voltage
- Security needs to be studied and tested
- Other pairings (Ate, Eta, Eil, etc…)
- Pairings for hyperelliptic curves, abelian varieties
- etc….

Questions?

Thank You!