An Elliptic Curve Threshold Key Establishment Scheme

Talia Ringer

April 21, 2012

Abstract

Threshold schemes provide a means through which two groups can communicate if and only if a certain number of members participate. Existing threshold schemes focus on generating a group public and secret key pair which can be used to encrypt or sign. The scheme described in this paper uses a bilinear pairing to establish a secret key that can be used by two groups to communicate. The key can only be generated if at least t_A members of group A and at least t_B members of group B participate. Furthermore, individual shares and group secret keys can be reused in subsequent sessions.

Introduction

This paper describes an elliptic curve threshold key establishment scheme. It is motivated by the following problem: Two groups A and B would like to communicate securely, but only if at least t_A members of A and at least t_B members of B are present.

Other elliptic curve threshold schemes have been proposed. Distributed key generation and virtual secret sharing schemes like those mentioned in [3] and [4] provide means through which to generate a public and private key pair according to a given threshold and to use this key pair for encryption or for signature verification.

Unlike existing schemes, the scheme described in this paper generates a secret key that can be used for both groups to communicate. It is partially based on [1], in which Chen and Hsu describe a (t, n)-threshold scheme for encryption and digital signatures: At least t members of a group consisting of n total members must be present to generate the group secret key. It is unclear how this scheme accomplishes its goal. In the signature step, any individual in A can easily compute the group secret key, compromising it in all further sessions.

The scheme described in this paper establishes a shared secret key for groups A and B using group thresholds t_A and t_B. Long-term group and individual secret keys can be reused in subsequent sessions without an apparent sacrifice in security. It utilizes an elliptic curve not only for the boost in efficiency it provides, but also to allow for the use of a bilinear pairing for secret key generation. Finally, it allows for groups A and B to calculate their group public keys and use their group secret keys directly, without the use of a secret key center, and without revealing the respective group secret to any individual within the group.

This paper begins by explaining the preparation steps for groups A and B. These steps are similar to those described in Chen and Hsu's paper, but with the secret key center removed. They are included for the sake of clarity and completeness. Next, it describes shared key generation for both groups. It then provides an example to demonstrate how this could be used in practice. It concludes with an analysis of the security of the scheme and the freshness of the group and individual secrets.

Group A Preparation

Let E be an elliptic curve over a field of p elements, let G be a point on E with large prime order N, and let t_A be the number of members of Group A who must collaborate to send or receive a message. The preparation for Group A is as follows:

1. The Certificate Authority (CA) chooses a secret key a_0 and a secret interpolation polynomial for Group A
$$P_A(x) = a_{t_A-1}x^{t_A-1} + ... + a_1x + a_0$$
The CA is the only party who ever knows a_0 and $P_A(x)$.

2. The CA sends each member A_i of Group A his secret share $n_{A_i} = P_A(i)$. The secret share n_{A_i} is never revealed to any other member of A.

3. The CA verifies the members of Group A. For each member A_i:

   (a) The CA chooses a random integer X_i, computes $Y = X_iG$, and sends Y_i to A_i.

   (b) A_i computes $C_i = n_{A_i}Y_i = n_{A_i}(X_iG)$ and sends C_i to the CA.

   (c) The CA computes $C_i' = n_{A_i}(X_iG)$. If $C_i = C_i'$ then A_i must know his secret share n_{A_i}, thus A_i is verified.

4. Group A generates their public key a_0G:

   (a) Each member A_i shares $n_{A_i}G$ and i with the other members of A. Each secret share n_{A_i} is masked by the discrete logarithm problem.

   (b) Group A calculates the public key

$$\begin{aligned} a_0G &= P_A(0)G \\ &= \left(\sum_{j=1}^{t_A} n_{A_j} \prod_{\substack{i,j=1 \\ i\neq j}}^{t_A} \frac{-i}{j-i}\right)G \\ &= \sum_{j=1}^{t_A}\left(\prod_{\substack{i,j=1 \\ i\neq j}}^{t_A} \frac{-i}{j-i} \pmod N\right)n_{A_j}G \end{aligned}$$

Group A's secret key a_0 is masked from the public by the discrete logarithm problem. Additionally, it is never computed explicitly by Group A. No individual member of A knows a_0 unless t_A members collaborate. Each individual in A shares $n_{A_i}G$ with the other members of A rather than n_{A_i}. This prevents any individual in A from interpolating to find a_0. It is instead masked by G, so that Group A's secret key a_0 is never calculated explicitly.

Group B Preparation

Let t_B be the number of members of Group B who must collaborate to send or receive a message. The preparation for Group B is as follows:

1. The Certificate Authority (CA) chooses a secret key b_0 and a secret interpolation polynomial for Group B
$$P_B(x) = b_{t_B-1}x^{t_B-1} + ... + b_1x + b_0$$
The CA is the only party who ever knows b_0 and $P_B(x)$.

2. The CA sends each member B_j of Group B his secret share $n_{B_j} = P_B(j)$. The secret share n_{B_j} is never revealed to any other member of B.

3. The CA verifies the members of Group B. For each member B_j:

   (a) The CA chooses a random integer X_j, computes $Y_j = X_jG$, and sends Y_j to B_j.

   (b) B_j computes $C_j = n_{B_j}(X_jG)$ and sends C_j to the CA.

   (c) The CA computes $C'_j = n_{B_j}(X_jG)$. If $C_j = C'_j$ then B_j must know his secret share n_{B_j}, thus B_j is verified.

4. Group B generates their public key b_0G:

   (a) Each member B_j shares $n_{B_j}G$ and j with the other members of B. Each secret share n_{B_j} is masked by the discrete logarithm problem.

   (b) Group B calculates the public key

$$\begin{aligned} b_0G &= P_B(0)G \\ &= \left(\sum_{j=1}^{t_B} n_{B_j} \prod_{\substack{i,j=1 \\ i\neq j}}^{t_B} \frac{-i}{j-i}\right)G \\ &= \sum_{j=1}^{t_B}\left(\prod_{\substack{i,j=1 \\ i\neq j}}^{t_B} \frac{-i}{j-i} \pmod N\right)n_{B_j}G \end{aligned}$$

Group B's secret key b_0 is masked from the public by the discrete logarithm problem. Additionally, it is never computed explicitly by Group B. No individual member of A knows b_0 unless t_B members collaborate. Each individual in B shares $n_{B_j}G$ with the other members of A rather than n_{B_j}. This prevents any individual in B from interpolating to find b_0. It is instead masked by G, so that Group B's secret key b_0 is never calculated explicitly.

Shared Key Generation

Shared Key Generation for Group A

1. The CA picks a random integer k, calculates kG, and shares a signed version of this with Group A and Group B.

2. Each member A_i computes $n_{A_i}(kG)$ and shares this with the other members of Group A.

3. Group A computes

$$
\begin{aligned}
P &= a_0(kG) \\
&= \sum_{j=1}^{t_A} \Bigg(\prod_{\substack{i,j=1 \\ i \neq j}}^{t_A} \frac{-i}{j-i} \pmod N \Bigg) n_{A_j} kG
\end{aligned}
$$

4. Group A computes the symmetric key $K = H(< P, b_0 G >)$, where H is a cryptographic hash function and $b_0 G$ is Group B's public key.

Note that the CA chooses kG to prevent anyone in Group A or Group B from knowing k. If any individual in A were to know k, then the individual could use the public key shares $n_{A_i} G$ to compute

$$
\begin{aligned}
P &= a_0(kG) \\
&= \sum_{j=1}^{t_A} \Bigg(\prod_{\substack{i,j=1 \\ i \neq j}}^{t_A} \frac{-i}{j-i} \pmod N \Bigg) k(n_{A_j} G)
\end{aligned}
$$

This would reveal the symmetric key $K = H(< P, b_0 G >)$ to an individual in A. Similarly, Group B is only given kG rather than k, thus no individual in Group B can compute k due to the difficulty of the discrete logarithm problem.

Shared Key Generation for Group B

1. Each member B_j computes $n_{B_j}(kG)$ and shares this with the other members of Group B.

2. Group B computes

$$
\begin{aligned}
Q &= b_0(kG) \\
&= \sum_{j=1}^{t_B} \Bigg(\prod_{\substack{i,j=1 \\ i \neq j}}^{t_B} \frac{-i}{j-i} \pmod N \Bigg) n_{B_j} kG
\end{aligned}
$$

3. Group B computes the symmetric key $K = H(< Q, a_0 G >)$, where H is a cryptographic hash function and $a_0 G$ is Group A's public key.

Note that the symmetric key K computed by groups A and B will be the same due to bilinearity:

$$
\begin{aligned}
K &= H(< Q, a_0 G >) \\
&= H(< b_0(kG), a_0 G >) \\
&= H(< a_0(kG), b_0 G >) \\
&= H(< P, b_0 G >)
\end{aligned}
$$

The key K can now be used by A and B to communicate.

Example

Let $t_A = 3$, $t_B = 4$, and $N = 101$. Let the curve E be $y^2 = x^3 + 176x + 1 \pmod{607}$, and let $G = (304, 131, 1)$.

1. The CA chooses a secret key $a_0 = 27$ and a secret interpolation polynomial
$$P_A(x) = a_2x^2 + a_1x + a_0 = 14x^2 + 62x + 27$$
2. The CA sends the members of Group A their secret shares:
$$n_{A_1} = P_A(1) = 14 + 62 + 27 = 2 \pmod{101}$$
$$n_{A_2} = P_A(2) = 14 \cdot 4 + 62 \cdot 2 + 27 = 5 \pmod{101}$$
$$n_{A_3} = P_A(3) = 14 \cdot 9 + 62 \cdot 3 + 27 = 36 \pmod{101}$$
3. The CA verifies the members of Group A:
   (a) The CA chooses a random integer $X_1 = 22$, computes $Y_1 = 22G$, and sends Y_1 to A_1.
   (b) A_1 computes $C_1 = n_{A_1}(22G) = 2(22G) = 44G$ and sends C_1 to the CA.
   (c) The CA computes $C_1' = n_{A_1}(Y_1) = 2(22G) = 44G$ and verifies that $C_1 = C_1'$, so A_1 is verified.
   (d) The CA chooses a random integer $X_2 = 79$, computes $Y_2 = 79G$, and sends Y_2 to A_2.
   (e) A_2 computes $C_2 = n_{A_2}(79G) = 5(79G) = 92G$ and sends C_2 to the CA.
   (f) The CA computes $C_2' = n_{A_2}(Y_2) = 5(79G) = 92G$ and verifies that $C_2 = C_2'$, so A_2 is verified.
   (g) The CA chooses a random integer $X_3 = 10$, computes $Y_3 = 10G$, and sends Y_3 to A_3.
   (h) A_3 computes $C_3 = n_{A_3}(10G) = 36(10G) = 57G$ and sends C_3 to the CA.
   (i) The CA computes $C_3' = n_{A_3}(Y_3) = 36(10G) = 57G$ and verifies that $C_3 = C_3'$, so A_3 is verified.
4. Group A generates their public key a_0G:
   (a) A_1 shares $n_{A_1}G = 2G$ and 1 with the other members of A, A_2 shares $n_{A_2}G = 5G$ and 2, and A_3 shares $n_{A_3}G = 36G$ and 3.
   (b) Group A calculates the public key

$$\begin{aligned}
a_0G &= P_A(0)G \\
&= \left(\sum_{j=1}^{3} n_{A_j} \prod_{\substack{i,j=1 \\ i\neq j}}^{3} \frac{-i}{j-i}\right)G \\
&= \sum_{j=1}^{3} n_{A_j}G \prod_{\substack{i,j=1 \\ i\neq j}}^{3} \frac{-i}{j-i} \\
&= nA_1G\left(\frac{(-2)(-3)}{(1-2)(1-3)}\right) + nA_2G\left(\frac{(-1)(-3)}{(2-1)(2-3)}\right) + nA_3G\left(\frac{(-1)(-2)}{(3-1)(3-2)}\right) \\
&= 2G\left(\frac{6}{2}\right) + 5G\left(\frac{3}{-1}\right) + 36G\left(\frac{2}{2}\right) \\
&= 6G - 15G + 36G = 27G = (215, 79, 1)
\end{aligned}$$

5. The CA chooses a secret key $b0 = 4$ and a secret interpolation polynomial

$$P_B(x) = b_3x^3 + b_2x^2 + b_1x + b_0 = 20x^3 + 95x^2 + 16x + 4$$

6. The CA sends the members of Group B their secret shares:

$$n_{B_1} = P_B(1) = 20 + 95 + 16 + 4 = 34 \pmod{101}$$

$$n_{B_2} = P_B(2) = 20 \cdot 8 + 95 \cdot 4 + 16 \cdot 2 + 4 = 71 \pmod{101}$$
$$n_{B_3} = P_B(3) = 20 \cdot 27 + 95 \cdot 9 + 16 \cdot 3 + 4 = 33 \pmod{101}$$
$$n_{B_4} = P_B(4) = 20 \cdot 64 + 95 \cdot 16 + 16 \cdot 4 + 4 = 40 \pmod{101}$$

7. The CA verifies the members of Group B:

   (a) The CA chooses a random integer $X_1 = 52$, computes $Y_1 = 52G$, and sends Y_1 to B_1.

   (b) B_1 computes $C_1 = n_{B_1}(52G) = 34(52G) = 51G$ and sends C_1 to the CA.

   (c) The CA computes $C_1' = n_{B_1}(Y_1) = 34(52G) = 51G$ and verifies that $C_1 = C_1'$, so B_1 is verified.

   (d) The CA chooses a random integer $X_2 = 79$, computes $Y_2 = 12G$, and sends Y_2 to B_2.

   (e) B_2 computes $C_2 = n_{B_2}(12G) = 71(12G) = 44G$ and sends C_2 to the CA.

   (f) The CA computes $C_2' = n_{B_2}(Y_2) = 71(12G) = 44G$ and verifies that $C_2 = C_2'$, so B_2 is verified.

   (g) The CA chooses a random integer $X_3 = 86$, computes $Y_3 = 86G$, and sends Y_3 to B_3.

   (h) B_3 computes $C_3 = n_{B_3}(86G) = 33(86G) = 10G$ and sends C_3 to the CA.

   (i) The CA computes $C_3' = n_{B_3}(Y_3) = 33(86G) = 10G$ and verifies that $C_3 = C_3'$, so B_3 is verified.

   (j) The CA chooses a random integer $X_4 = 41$, computes $Y_4 = 41G$, and sends Y_4 to B_4.

   (k) B_4 computes $C_4 = n_{B_4}(41G) = 40(41G) = 24G$ and sends C_4 to the CA.

   (l) The CA computes $C_4' = n_{B_4}(Y_4) = 40(41G) = 24G$ and verifies that $C_4 = C_4'$, so B_4 is verified.

8. Group B generates their public key b_0G:

   (a) B_1 shares $n_{B_1}G = 34G$ and 1 with the other members of B, B_2 shares $n_{B_2}G = 71G$ and 2, B_3 shares $n_{B_3}G = 33G$ and 3, and B_4 shares $n_{B_4}G = 40G$ and 4.

   (b) Group B calculates the public key

$$
\begin{aligned}
b_0G &= P_B(0)G \\
&= \left(\sum_{j=1}^{3} n_{B_j} \prod_{\substack{i,j=1 \\ i\neq j}}^{3} \frac{-i}{j-i}\right)G \\
&= \sum_{j=1}^{3}\left(\prod_{\substack{i,j=1 \\ i\neq j}}^{3} \frac{-i}{j-i}\right)n_{B_j}G \\
&= nB_1G\left(\frac{(-2)(-3)(-4)}{(1-2)(1-3)(1-4)}\right) + nB_2G\left(\frac{(-1)(-3)(-4)}{(2-1)(2-3)(2-4)}\right) \\
&+ nB_3G\left(\frac{(-1)(-2)(-4)}{(3-1)(3-2)(3-4)}\right) + nB_4G\left(\frac{(-1)(-2)(-3)}{(4-1)(4-2)(4-3)}\right) \\
&= 34G\left(\frac{-24}{-6}\right) + 71G\left(\frac{-12}{2}\right) + 33G\left(\frac{-8}{-2}\right) + 40G\left(\frac{-6}{6}\right) \\
&= 35G - 22G + 31G - 40G = 4G = (395, 354, 1)
\end{aligned}
$$

9. The CA picks a random integer $k = 82$, computes $82G = (536, 423, 1)$, signs this, and sends it to Group A and Group B.

10. A_1 computes $n_{A_1}(kG) = 2(82G) = 63G$ and shares this with the other members of Group A, A_2 computes $n_{A_2}(kG) = 5(82G) = 6G$, and A_3 computes $n_{A_3}(kG) = 36(82G) = 23G$.

11. Group A computes

$$
\begin{aligned}
P &= a_0(kG) \\
&= \sum_{j=1}^{3} n_{A_j} kG \prod_{\substack{i,j=1 \\ i\neq j}}^{3} \frac{-i}{j-i} \\
&= 63G\left(\frac{6}{2}\right) + 6G\left(\frac{3}{-1}\right) + 23G\left(\frac{2}{2}\right) \\
&= 93G = (11, 422, 1)
\end{aligned}
$$

12. Group A computes the symmetric key $K = H(< P, b_0G >) = H(< 93G, 4G >) = H(< (11, 422, 1), (395, 354, 1) >) = H(193)$.

13. B_1 computes $n_{B_1}(kG) = 34(82G) = 61G$ and shares this with the other members of Group B, B_2 computes $n_{B_2}(kG) = 71(82G) = 65G$, B_3 computes $n_{B_3}(kG) = 33(82G) = 80G$, and B_4 computes $n_{B_4}(kG) = 40(82G) = 48G$.

14. Group B computes

$$
\begin{aligned}
Q &= b_0(kG) \\
&= \sum_{j=1}^{t_B} n_{B_j} kG \prod_{\substack{i,j=1 \\ i \neq j}}^{t_B} \frac{-i}{j-i} \pmod{p} \\
&= 61G\left(\frac{-24}{-6}\right) + 65G\left(\frac{-12}{2}\right) + 80G\left(\frac{-8}{-2}\right) + 48G\left(\frac{-6}{6}\right) \\
&= 25G = (554, 260, 1)
\end{aligned}
$$

15. Group B computes the symmetric key $K = H(< Q, a_0 G >) = H(< 25G, 27G >) = H(< (554, 260, 1), (215, 79, 1) >) = H(193)$. Using SHA-1 on the ASCII character given by 193 gives 0x7860BB419C8FDAA76A6F6F1EFD28FE595CD531CD. This symmetric key can now be used by groups A and B to communicate.

Security

The following table summarizes what each party learns at any given point in the protocol and is included as a reference. It is assumed that at any point, each party knows what it has learned at all previous points (i.e. in all of the rows above the given row for its column). It is assumed that the individual secret shares n_{A_i} and n_{B_j} are sent securely to the individuals A_i and B_j, and that anything sent from an individual member of a group to the rest of the group is also sent securely. Anything that an eavesdropper can trivially discern that does not need to be securely transmitted is considered public.

		Parties					
		CA	A	B	a_i	b_j	Public
Group A Preparation	1	$P_A(x), a_0$					G, H, N, p, t_A, t_B
	2	n_{A_i}			n_{A_i}		
	3	X_i, C_i'					Y_i, C_i
	4		$i, n_{A_i} G$				$a_0 G$
Group B Preparation	1	$P_B(x), b_0$					
	2	n_{B_j}				n_{B_j}	
	3	X_j, C_j'					Y_j, C_j
	4			$j, n_{B_j} G$			$b_0 G$
Group A Shared Key Generation	1	k				kG	
	2		$n_{A_i} kG$				
	3		$a_0 kG$				
	4		K				
Group B Shared Key Generation	1			$n_{B_j} kG$			
	2			$b_0 kG$			
	3			K			

From the very beginning, this protocol assumes a trusted certificate authority. The CA chooses the interpolation polynomials $P_A(x)$ and $P_B(x)$ and the random k, and is thus aware of all information that would be necessary to establish the shared secret key K.

The group secret key and individual secret shares are always masked by the discrete logarithm problem. The use of a random k ensures that all t_A or t_B members must know n_{A_i} or n_{B_j} in order for the group to be able to calculate K. Note that the CA only provides each group with kG rather than k, so that k is also masked by the discrete logarithm problem. Knowledge of k by either of the groups would compromise the protocol, as they would be able to use $n_{A_i}G$ or $n_{B_j}G$ to calculate the shared secret.

A single member of Group A or Group B can commit a denial of service attack on its own group by contributing the incorrect secret share. This could possibly be fixed with a majority vote.

In a single session, t_A members of Group A are required in order to generate the group secret key a_0 at all, and analogously, t_B members of Group B are required in order to generate b_0. This follows trivially from the use of an interpolation polynomial.

Suppose t'_A members of Group A collaborate in an attempt to generate a key K in a subsequent session, and that $t'_A < t_A$. We can assume they know the values of $n_{A_i}G$ for at least t_A members of A. In order to generate a K, they must know values of $n_{A_i}kG$ for $t_A - t'_A$ members of A. k changes between sessions, so the values from the previous section are no longer valid. They cannot collaborate to fix a random k because kG is sent by the CA to Group B as well, and so if they choose k, the value B uses will result in a different key K. Since kG is signed, Group A will not be able to send a fake kG to Group B, otherwise Group B will know it is invalid.

Note that if any number of people in Group A collaborates with any number of people from Group B, they can easily fix kG and send messages to one another, assuming only members who have agreed to collaborate are present. However, presumably if any number of people in Group A and any number of people in Group B agree to communicate, they can do so outside of the protocol; thus, this is considered out of scope.

It needs to be ensured that the point G does not pair with itself to one. For this reason, the Weil pairing cannot be used, as every point would pair with itself to one. The Tate pairing is sufficient for this.

Freshness

This protocol has the advantage that several fields can be reused in a future iteration of the algorithm without any apparent loss of security:

1. a_0 and b_0: The group secret keys are never revealed to anybody except the CA because they are always masked by the Discrete Logarithm Problem. In future iterations, generating a different random k will change the key K regardless of reuse of a_0 and b_0. This will prevent any individual in Group A or Group B from being able to send another message, assuming that K is not allowed to be used during future communication.

2. n_{A_i} and n_{B_j}: Likewise, the individual secret shares are always masked by the discrete logarithm problem. Each n_{A_i} and n_{B_j} can be reused in a future iteration, thus any subset of t_A or greater people in Group A and t_B or greater people in Group B can generate the group secret keys and thus the key K.

3. a_0G and b_0G: It follows that the group public key can be reused, as it is related to the group secret keys.

4. $P_A(x)$ and $P_B(x)$: These polynomials determine the secret key, public key, and the secret shares, and can thus be reused as well.

If there are more than t_A total members of A then any subset of people in A of size t_A or greater can generate K. It appears that the same group secret key, secret shares, public key, and interpolation

polynomial can be reused even if a different subset of people in A are present in a future session, as long as there are at least t_A of them. The same follows for B.

Note that while t_A members of A and t_B members of B are required in order to generate K, once K has been established, if it is used directly then it is revealed to everyone in Group A and Group B. Therefore, after a session of communication is terminated, the shared secret key K should be reestablished. Since the group keys and individual shares can be reused, in a subsequent session, the algorithm can skip to shared key generation after the verification step.

Conclusion

This scheme solves the problem of two groups who want to communicate securely if and only if a certain number of members of each group are present. As a key establishment scheme, it approaches this problem in an angle that is unique from that taken by existing schemes. In using a bilinear pairing to generate the key, it utilizes elliptic curve cryptology beyond the boost in performance it provides. Finally, it allows for the reuse of individual secret shares and group keys without an apparent loss of security. In future work, it would be interesting to integrate the use of a majority vote from existing threshold schemes into the ideas mentioned in this paper. This could prevent the denial of service attack that is possible from within a group and possibly eliminate the need for a trusted third party.

References

[1] Tzer-Shyong Chen and Chieh-Yen Hsu, A Novel Threshold Scheme based on Elliptic Curve Cryptosystem and Grey System Theory. *International Mathematical Forum*, 1(21):1051,1059, 2006.

[2] Lawrence C. Washington, *Elliptic Curves: Number Theory and Cryptography, 2nd edition*. CRC Press, 2008.

[3] Maged H. Ibrahim I. A. Ali I. I. Ibrahim A. H. El-sawi, *A Robust Threshold Elliptic Curve Digital Signature Providing a New Verifiable Secret Sharing Scheme*, 2003 IEEE 46th Midwest Symposium on Circuits and Systems, 1:276,280, 2003.

[4] Rosario Gennaro, Stanislaw Jarecki, Hugo Krawczyk, and Tal Rabin, Secure Distributed Key Generation for Discrete-Log Based Cryptosystems. *Journal of Cryptology*, 1(20):51,83, 2007.