The Elliptic Curve Diffie-Hellman-Merkle Key Exchange

River McCubbin, Yingjian/Raymond Jia

November 21, 2023

1 Abstract

The Elliptic Curve Diffie-Hellman-Merkle Key Exchange, abbreviated ECDHKE, is a public key exchange protocol based on the Diffie-Hellman-Merkle key exchange (DHKE) protocol. Originally outlined by Ralph Merkle and later published by Whitfield Diffie and Martin Hellman in 1976 [4], the DHKE is designed to allow two parties to publicly exchange information in order to securely generate a shared cryptographic key. This key can then be used to encrypt messages using other cryptographic systems for example RSA. The ECDHKE is a variant of the DHKE designed using elliptic curves rather than the traditional modular arithmetic. Both the DHKE and ECDHKE protocols are secured by the Discrete Logarithm Problem and the Diffie-Hellman Problem.

1.1 Elliptic Curves

Definition 1 (Elliptic Curve)**.** A *elliptic curve* over a field $\mathbb{F}$ is the set of solutions to the equation $E(\mathbb{F}) = \{(x, y) \in \mathbb{F}^2 \mid y^2 = x^3 + bx + c\} \cup \{\infty\}$.

We can make elliptic curves more interesting by defining a group operation as follows:

Definition 2 (Addition on Elliptic Curves)**.** Given $a, b \in E$ we define $a + b$ to be the intersection of the line through a and b with E; if there is no solution we say that this is the element ∞.

Rigorously we define this as follows:

$$(x_1, y_1) + (x_2, y_2) = (x_3, y_3)$$

where

$$x_3 = m^2 - x_1 - x_2$$
$$y_3 = m(x_1 - x_3) - y_1$$

and

$$m = \begin{cases} \frac{\Delta x}{\Delta y} & p_1 \neq p_2 \\ \frac{3x_1^2+b}{2y_1} = \frac{dy}{dx} & p_1 = p_2 \end{cases}$$

With this operation, we have that $(E(\mathbb{F}), +)$ is a group with identity element ∞. This group structure is what enables the ECDHKE protocol to function.

1.2 Diffie-Hellman Key Exchange

The Diffie-Hellman Key Exchange protocol is a public key exchange protocol. This means that it is designed to allow two parties to securely generate a shared cryptographic key over an insecure/public channel. This key can then be used as an encryption/decryption key in other cryptographic messaging systems.

The Diffie-Hellman key exchange functions as follows:

Step 1. First a public system is generated with a base g.

Step 2. Alice and Bob each generate private keys a and b.

Step 3. They each mix their keys with the public base to generate ga and gb. We suppose that determining x from gx is difficult in this system.

Step 4. They then share their mixture with the other party; Alice sends ga to Bob, and Bob sends gb to Alice.

Step 5. Alice and Bob then mix their private key with the mixture they receive; Alice generates gba and Bob generates gab.

Step 6. If we suppose that this system is commutative then we have that they share the value gab.

Notice that an eavesdropper would have the values g, ga, and gb, but with the supposition from above, this leaves them unable to determine the key gab. This is the idea behind the DHKE. Typically the Discrete Logarithm Problem (Section 1.3.1) and the Diffie-Hellman problem (Section 1.3.2) are used to ensure the security of this system (i.e. to make it difficult to determine a from ga).

2 Elliptic Curve Diffie-Hellman Key Exchange

2.1 Process

On elliptic curves, ECDHKE is carried out as follows:

Step 1. An elliptic curve E is chosen over $\mathbb{Z}_p$ for a prime p, along with a point G which is a generator for a cyclic subgroup of E (this generator is the "base" from earlier)

Step 2. Alice and Bob choose integers a and b respectively to be their private keys. These private keys must satisfy $1 \leq a, b \leq n-1$ where n is the order of G.

Step 3. Alice and Bob then generate their public keys by finding $A = aG$ and $B = bG$, respectively

Step 4. Alice then shares her public key, A with Bob, and vice versa Bob shares his public key B with Alice.

Step 5. Alice can then compute $aB = abG = (x_s, y_s)$ and Bob can compute $bA = baG = (x_s, y_s)$.

Step 6. Then Alice and Bob have computed the same value and share the secret value x_s, which they can use as a shared key for other purposes.

2.2 Proof

The only necessary proof for this operation to function is that both parties generate the same point at the end. If we denote the secret point S and define it to be the value aB that Alice generates, then since scalar multiplication is commutative over elliptic curves we have that

$$S = aB = abG = baG = bA$$

Which is the value that Bob generates. Therefore they have generated the same value, as wanted.

2.3 Example Key Exchange

Take $E(\mathbb{Z}_7) = \{(x, y) \mid y^2 \equiv x^3 + x + 1\}$ generated by $G = (0, 1)$.

Alice chooses $a = 3$ and computes $aG = (2, 2)$

Bob chooses $b = 4$ and computes $bG = (0, 6)$

They then share their public keys

Alice computes $abG = a(0, 6) = (2, 5)$
Bob computes $baG = b(2, 2) = (2, 5)$

They then have the shared value $(2, 5)$, and we take the x component as the shared secret.

Notice that only aG, bG and G are public known to any eavesdroppers.

3 Security/Attacks

3.1 Important Values

There are of course vulnerabilities to the ECDHKE, but we can reduce the significance of these by carefully choosing our *domain parameters*. The domain parameters of the ECDHKE are the values that must be agreed upon before completing the exchange of information. These include the parameters of the elliptic curve (p, a, b) and the generator chosen (G). We can write the parameters for the system as (p, a, b, G).

Firstly, we have that $|E(K)| \approx p + 1$, hence choosing a large prime p increases the number of possible points, decreasing the utility of brute force attacks (guessing the solution). We also want to choose p such that $p - 1$ has at least one large prime factor, as many attacks on this rely on the prime factors of $p - 1$ [6].

Second, we have the choice of G. Recall that G must generate a cyclic subgroup of E for ECDHKE. By Lagrange's Theorem we have that the order of G, $|G|$, must divide the order of the group i.e. $\frac{|E|}{|G|}$ must be an integer. In order to maximize the size of the group generated by G, we would like to find G such that $|E| = |G|$, i.e. G generates E. This maximizes the number of elements in the group and hence maximizes the number of possible keys that an attacker must check if they were to try to guess the key.

A criterion as to how large our p should be, knowing that if $2^{k-1} < p < 2^k$, we say this p is k-bit. By the bound we covered in class:

$$\begin{aligned}
&|N - (p+1)| \leq 2\sqrt{p} \\
&\text{ roughly:} \\
&2^k - 2 \cdot 2^{k/2} < N - 1 < 2^k + 2 \cdot 2^{k/2}
\end{aligned}$$

Then, the order of our elliptic curve will be close to our choice of p.

If a prime number p is $k-$bit, then the 'effort' of breaking will be measured by about $2^{k/2}$. Normal computers can do anything beyond 2^{100}, hence one might see '128-bit/256-bit security.'

3.2 Securing Problems and Their Weaknesses

There are two primary problems that secure ECDHKE. The first is the Discrete Logarithm Problem (DLP) and the second is the Diffie-Hellman Problem (DHP).

3.2.1 The Discrete Logarithm Problem

The DLP is stated as follows:

Problem 1 (Discrete Logarithm Problem)**.** Given a group G generated by g and an element $h = g^a \in G$, find a.

Under normal circumstances this problem has no known polynomial time algorithm, meaning it is considered inefficient and infeasible for sufficiently large groups, and remains an open problem in mathematics and computing science. This problem is not perfect though; The Logjam attack was developed for well-known fields by pre-computing the majority of the necessary steps for an attack, leaving only a small amount of computational work to be done to compromise a system. When this algorithm was developed in 2015 it was shown that 8.4% of the top 1 million domains on the internet were vulnerable to this attack [1]. Luckily the solution to this is to simply increase the size of the group/curve or to use unfamiliar curves to prevent this sort of precomputation. The authors of the Logjam attack and associated paper recommend a minimum size of 2048 bits to ensure the security of these types of systems.

3.2.2 The Diffie-Hellman Problem

The DHP is stated as follows:

Problem 2. Given a group G generated by g and two element g^a and g^b, find g^{ab}.

It has been shown that solving this problem is at most as hard as solving the DLP (if you can find a and b from g^a and g^b then computing g^{ab} is trivial). This problem can be approached in a different way: rather than computing g^{ab}, instead use g^a and g^b to distinguish g^{ab} from other group elements. This is referred to as the Decisional Diffie-Hellman Problem (DDHP). Neither the DHP nor the DDHP have any efficient algorithms and are easily protected once again by simply ensuring that the group used is large.

3.3 Attacks

Since ECDHKE is secured by the DLP and DHP, any attack on ECDHKE must be able to solve these problems. Though there is no known polynomial time algorithm, the following algorithms can be viable given certain circumstances:

3.3.1 Pohlig-Hellman Algorithm

The PH algorithm was introduced by Roland Silver and published by Stephen Pohlig and Martin Hellman [7]. The algorithm solves the DLP in $O(\sqrt{p})$ group operations; this is once again exponential time in terms of the digits of the size of the group, but it is a significant improvement from $O(\sqrt{n})$. The algorithm is as follows:

Given a generator g of a cyclic group G of order $p-1$ and a factorization $p-1 = \prod p_i^{\alpha_i}$ with p_i distinct primes, apply the following steps for each p_i:

Intuitively, this algorithm is to break apart the problem into smaller pieces, then putting all the pieces back to get the answer.
On an elliptic curve, suppose we are given P, Q and want to know $kP = Q$ for some k. Let n be the order of our elliptic curve.

1. Find the factorization of
$$n = p_1^{e_1}...p_r^{e_r}$$

2. Take any $p_i^{e_i}$, express k in terms of:
$$k = a_0^i + a_1^i p_i + a_2^i p_i^2 + ... + a_{e_i-1}^i p_i^{e_i-1}$$
Then, we have:
$$Q_0^i = (a_0^i + a_1^i p_i + a_2^i p_i^2 + ... + a_{e_i-1}^i p_i^{e_i-1})P$$

3. Multiply both side by $\frac{n}{p_i}$:
$$\frac{n}{p_i} Q_0^i = (a_0^i + a_1^i p_i + a_2^i p_i^2 + ... + a_{e_i-1}^i p_i^{e_i-1})\frac{n}{p_i}P$$

By Lagrange's Theorem, the order of P divides the order of the group, hence we can cancel all terms except for a_0.
$$\frac{n}{p_i} Q_0^i = \frac{a_0^i n}{p_i} P$$

4. Knowing that $0 \leq a_0^i < p_i$. Hence, we can brute-forcing the calculation to get a_0^i, usually done by computer.

5. Continue this fashion, try this on Q_1^i, which is defined by:
$$Q_1^i = Q_0^i - a_0^i P = (a_1^i p_i + a_2^i p_i^2 + ... + a_{e_i-1}^i p_i^{e_i-1})\frac{n}{p_i^2}P$$

This time, we can brute-forcing a_2, all the way until we crack all of the coefficient $a's$, and now we have:
$$x \equiv a_0^i + a_1^i p_i + a_2^i p_i^2 + ... + a_{e_i-1}^i p_i^{e_i-1} \bmod p_i^{e_i}$$

6. Iterate until we have a system of congruence, base on different prime factor:
$$x \equiv a_0^1 + a_1^1 p_1 + ... + a_{e_1-1}^1 p_1^{e_1-1} \bmod p_1^{e_1}$$
$$\cdots$$
$$x \equiv a_0^r + a_1^r p_1 + ... + a_{e_r-1}^r p_r^{e_r-1} \bmod p_r^{e_r}$$

7. Since all of $p_i^{e_i}$'s are coprime, then we can use *Chinese Remainder Theorem* to solve for x.

Remark: Even knowing the order of our elliptic curve given a large prime is not an easy task, as the bound becomes larger.

3.4 Quantum Computers

Of course, these rules apply only for classical computers; it has been shown that Shor's algorithm for quantum computers is able to defeat solve both of these problems in polynomial time. A solution to this was proposed by de Feo, Jao and Plut in 2014. They proposed the Super-Singular Isogeny Diffie-Hellman Key Exchange, abbreviated SIDH that claimed to be quantum-secure, meaning that no polynomial quantum algorithm existed to solve the necessary problems [5]. Unfortunately in June 2022 a key-recovery attack was published, making this system insecure [3]. Further details are well beyond the scope of this report.

4 Advantages Over DHKE

It is interesting that we can use ECDHKE as well as DHKE, but it is unclear at this point why we would choose to do this. The reason is that elliptic curve cryptography, in general, provides equivalent security at much smaller key sizes, for example using RSA and DHKE the security of a 15360-bit key can be achieved using only 521 bits with ECDHKE [2]. This drastic reduction in key size provides substantial storage and transmission benefits.

5 Appendix

5.1 Chinese Remainder Theorem

Let $\{n_i\}^k \subset \mathbb{N}$ such that $gcd(n_i, n_j) = 1, \forall i \neq j$, then the system:

$$\begin{aligned} &x \equiv a_1 \mod n_1 \\ &\dots \\ &x \equiv a_k \mod n_k \end{aligned}$$

has an unique solution for x up to congruence of $N = \prod n_i$.

5.2 The fundamental theorem of finite abelian groups

The cyclic group $\mathbb{Z}_{mn}$of order mn is isomorphic to the direct sum of $\mathbb{Z}_n$ and $\mathbb{Z}_m$ if and only if m, n are coprime. It follows that any finite abelian group is isomorphic to a direct sum of the form

$$\bigoplus_i^u \mathbb{Z}_{k_i}$$

in either of the following canonical ways:

- the numbers $k_1, k_2, ..., k_u$ are powers of (not necessarily distinct) primes,
- or k_1 divides k_2, k_2 divides k_3, and so on up to k_u.

References

[1] David Adrian et al. "Imperfect Forward Secrecy: How Diffie-Hellman Fails in Practice". In: *22nd ACM Conference on Computer and Communications Security*. Oct. 2015.

[2] National Security Agency. *The Case for Elliptic Curve Cryptography*. `http://www.nsa.gov/business/programs/elliptic_curve.shtml`. 2009. URL: `http://www.nsa.gov/business/programs/elliptic_curve.shtml`.

[3] Wouter Castryck and Thomas Decru. *An efficient key recovery attack on SIDH (preliminary version)*. Cryptology ePrint Archive, Paper 2022/975. `https://eprint.iacr.org/2022/975`. 2022. URL: `https://eprint.iacr.org/2022/975`.

[4] W. Diffie and M. Hellman. "New directions in cryptography". In: *IEEE Transactions on Information Theory* 22.6 (1976), pp. 644–654. DOI: `10.1109/TIT.1976.1055638`.

[5] Luca De Feo, David Jao, and Jérôme Plût. "Towards quantum-resistant cryptosystems from supersingular elliptic curve isogenies". In: *Journal of Mathematical Cryptology* 8.3 (2014), pp. 209–247. DOI: `doi:10.1515/jmc-2012-0015`. URL: `https://doi.org/10.1515/jmc-2012-0015`.

[6] Richard A. Mollin. *An Introduction to Cryptography*. Chapman and Hall, 2007.

[7] S. Pohlig and M. Hellman. "An improved algorithm for computing logarithms overGF(p)and its cryptographic significance (Corresp.)" In: *IEEE Transactions on Information Theory* 24.1 (1978), pp. 106–110. DOI: `10.1109/TIT.1978.1055817`.