Kleptographic Attacks on Elliptic Curve Signatures

Elsayed Mohamed and Hassan Elkamchouchi
Alexandria University, Alexandria, Egypt

Summary
This paper presents an approach to mount secretly embedded trapdoor with universal protection (SETUP) attacks on elliptic curve signatures. The attacked signature is the elliptic curve digital signature algorithm ECDSA. The attacker can obtain the user's private key covertly. The cryptographic black-box devices with this hidden trapdoor behave exactly like an honest devices while actually leaking the key to the attacker only. The paper also shows how to use ECDSA for encryption and key exchange.

Key words:
Elliptic Curve, Kleptography, Subliminal Channel, SETUP, ECDSA, Signature

1. Introduction

This introduction presents the cryptographic background on which the attacks are based.

- **Elliptic Curves**

Elliptic curves are known for their security. The common fields used for encryption are prime fields and characteristic 2 fields. Elliptic curves over prime fields are on the form:
$E: y^2 = x^3 + ax + b \bmod p$
where $a, b \in F_p$ and $4a^3 + 27b^2 \neq 0 \bmod p$

Formulas exist for group operations on points such as point negation, addition and multiplication by a scalar.

- **Elliptic Curve Discrete Logarithm Problem (ECDLP)**

Given a point P of order n in an elliptic curve E over a finite field F_p and a point Q in E , the ECDLP is to find an integer k , where $0 \leq k \leq n-1$, and $Q = kP$ if such a number exists.

- **Elliptic Curve Diffie-Hellman Problem (ECDHP)**

Given a point P of order n in an elliptic curve E over a finite field F_p and two points kP and lP where $0 \leq k, l \leq n-1$, the ECDHP is to find the point $k.lP$. ECDHP is used in the elliptic curve Diffie-Hellman key exchange.

- **Elliptic Curve Digital Signature Algorithm (ECDSA)**

In the elliptic curve digital signature algorithm (ECDSA) [1] a plaintext message represented by integer m is signed to the pair (r, s). The system parameters are the elliptic curve E defined over a finite field F_p, the base point G of prime order n, the private key d and the public key $Q = dG$. H is a cryptographically secure hash function that generates values less than n.

Signature Algorithm:
Input: Plaintext message m
Output: Signature (r, s)
Choose a random integer $k \leq n-1$
$R = kG = (x_1 , y_1)$
$r = x_1 \bmod n$
$s = k^{-1}(H(m) + d.r) \bmod n$
Output (r, s)

Verification Algorithm:
Input: Signature (r, s), Message m
Output: Verification of Signature (r, s) for the message m
$R` = s^{-1}(H(m)G + rQ) = (x_1` , y_1`)$
If $r = x_1` \bmod n$ then accept (r, s) as valid for message m else reject

- **Subliminal Channels**

Subliminal channels can be used to convey information in the output of a cryptosystem in a way different from the intended output. This notion was put forth by Simmons [2]. He demonstrated how a prisoner could leak secret messages to an outside partner without the warden knowing what is going on. The warden has the ability to read every message but still cannot read the secret message embedded within the cover message. Simmons further developed the concept to other applications including DSA [3].

- **Kleptography**

Kleptography is defined as the study of stealing information securely and subliminally within the context of cryptographic systems [4]. A kleptographic attack on the discrete logarithm problem has been introduced by Young and Yung in [4]. They defined a Secretly Embedded Trapdoor with Universal Protection (SETUP) as an algorithm that can be embedded within a cryptosystem to leak encrypted secret key information to the attacker in the output of that cryptosystem [5]. The encrypted secret key information is noticeable only to the attacker. The types of SETUP [4] and their definitions are listed below.

Definition 1. Assume that C is a black-box cryptosystem with a publicly known specification. A (regular) SETUP

mechanism is an algorithmic modification made to C to get C' such that:

1. The input of C` agrees with the public specifications of the input of C.
2. C` computes efficiently using the attacker's public encryption function $E$ (and possibly other functions as well), contained within $C$`.
3. The attacker's private decryption function D is not contained within C` and is known only by the attacker.
4. The output of C` agrees with the public specifications of the output of C. At the same time, it contains published bits (of the user's secret key) which are easily derivable by the attacker (the output can be generated during key-generation or during system operation like message sending).
5. Furthermore, the output of C and C' are polynomially indistinguishable (as in [6]) to everyone except the attacker.
6. After the discovery of the specifics of the SETUP algorithm and after discovering its presence in the implementation (e.g. reverse-engineering of hardware tamper-proof device), users (except the attacker) cannot determine past (or future) keys.

Definition 2. A weak SETUP is a regular SETUP except that the output of C and C ' are polynomially indistinguishable to everyone except the attacker and the owner/user of the device who is in control (knowledge) of his or her own private key (i.e., requirement 5 above is changed).

Definition 3. A strong SETUP is a regular SETUP, but in addition we assume that the users are able to hold and fully reverse-engineer the device after its past usage and before its future usage. They are able to analyze the actual implementation of C' and deploy the device. However, the users still cannot steal previously generated/future generated keys, and if the SETUP is not always applied to future keys, then SETUP-free keys and SETUP keys remain polynomially indistinguishable.

Definition 4. A kleptogram is an encryption of a value (hidden value) that is displayed within the bits of an encryption/signature of a plaintext value (outer value). Note that we say that a kleptogram is an encryption of a value, not a plaintext message. It is often the case in kleptography that the device is not free to choose this value. The device may calculate this hidden value, and then use it (for the 'randomness') in a subsequent computation, thus compromising that computation.

Definition 5. A SETUP that has (m, n)-leakage bandwidth leaks m secret messages over the course of n messages that are output by the cryptographic device (or n of its executions).

2. Proposed ECDSA SETUP Attack

ECDSA is chosen to demonstrate the possibility of embedding SETUP attacks on elliptic curve signatures. The private key of the attacker is v and the public key is $V = vG$. Hashing an elliptic curve point can be defined as hashing its x-coordinate. The device operates as follows:

Signature Algorithm with SETUP:
Input: Plaintext message m
Output: Signature (r, s)
For the first time the algorithm runs:
Choose a random integer $k_1 \leq n - 1$
$R_1 = k_1G = (x_1 , y_1)$
$r_1 = x_1 \bmod n$
$s_1 = k_1^{-1}(H(m_1) + d.r_1) \bmod n$
Output (r_1, s_1)
Store k_1 in non-volatile memory

For the next run times:
$Z = a.k_1G + b.k_1V + h.jG + e.uV$, where:
a, b, h, e are fixed integers $< n$
$j, u \in_R \{0, 1\}$ are uniformly and independently chosen at random
$k_2 = H(Z)$
$R_2 = k_2G = (x_2, y_2)$
$r_2 = x_2 \bmod n$
$s_2 = k_2^{-1}(H(m_2) + d.r_2) \bmod n$
Output (r_2, s_2)
Store k_2 in non-volatile memory

Everybody is able to verify the signature in a normal way at all times using the public key Q. The attacker can retrieve d and the next messages by obtaining (r_1, s_1) and (r_2, s_2) then computing k_2 as follows.

SETUP Decryption Algorithm:
Input: (r_1, s_1), (r_2, s_2), m_2
Output: Private key d
Assuming that the received signatures are valid, use the curve equation E over F_p to calculate the possible points R_1` on the curve whose $x$-coordinate mod $n = r_1$. There are two points at most. For each possible point $R_1$` do the following:
{
$Z_1 = aR_1` + b.vR_1` = a.k_1`G + b.v.k_1`G = a.k_1`G + b.k_1`V$
For each possible value of j, u do the following:
{
$Z_2 = Z_1 + h.jG + e.uV$
$k_2` = H(Z_2)$
$R_2` = k_2`G = (x_2`, y_2`)$
$r_2` = x_2` \bmod n$
If $r_2` = r_2$ then $k_2` = k_2$ so exit all loops
}
}
$d = (s_2k_2 - H(m_2)).\ r_2^{-1} \bmod n$

Thus the secret key d is obtained. This allows the attacker to forge signatures and decrypt encrypted messages.

3. Discussion and Analysis

A. Security

Since k_1 is random it follows that Z is uniformly distributed within the group generated by G. This SETUP attack is secure in the sense that a user not knowing the random choice k_1 cannot calculate the second private key k_2 as long as ECDHP is hard. This can be proven by supposing that an oracle A can solve the ECDH problem so that $A(aG, bG) = abG$. If A is applied on R_1 and V then: $A(R_1, bV) = b.v.k_1G = b.k_1V$, which can be used to calculate Z. Also an adversary that does not know the attacker's private key v cannot calculate Z and therefore cannot calculate k_2. This makes the universal protection property of the SETUP attack. Assuming that H is a pseudorandom function and that the device can be reverse-engineered, the outputs of C and $C`$ are polynomially indistinguishable. This results from Z being uniformly distributed and H being a pseudorandom function. A user that knows his private key d can recover k. Thus the ECDSA SETUP attack is a regular SETUP as long as ECDHP is hard and H is a pseudorandom function whose seed is kept hidden in the device. It is not strong because a user who knows his own private key can recover the choices of k and detect the presence of the SETUP given a, b and the seed. The random values j and u are used to add randomization to further ensure undetectability of SETUP in a black-box implementation. Adding them serves as a precaution so that if the random parameter k_i is available to the user and the hash function H is invertible, the user still cannot detect the presence of a SETUP in the device by running the device many times and guessing several different values of V. It also helps to curb trying to notice any possible probabilistic relations between some properties in V and some corresponding properties in Z. This kind of probabilistic detection by the user is very difficult in elliptic curve cryptosystems compared to discrete log systems where quadratic residuosity can be used to test a possible relation between the attacker's public key and Z [4]. This makes elliptic curve devices a better candidate for kleptographic attacks in addition to the improved security and key length advantages of elliptic curve systems.

B. Bandwidth

Retrieving one key requires the system to run twice. This leads to a bandwidth of (1, 2). By chaining the generation of k we can increase the bandwidth to (m, m + 1).

C. Further Usage

After signing two messages the user's private key becomes known to the attacker. The attacker and the user can then communicate in high bandwidth using Simmons' broadband subliminal channel [3]. The attack presented here avoids the partners' need to pre-arrange secure communication and agreement on the secret key before the time they need to communicate subliminally. The key can now be communicated subliminally over an insecure channel by sending two innocent messages with their signatures.

4. Using ECDSA for Encryption and Key Exchange

Young and Yung described rogue use of DSA for encryption and key exchange [7]. In this paper we extend this possibility to ECDSA. d_A and Q_A are Alice's private and public keys, respectively. d_B and Q_B are Bob's private and public keys, respectively. Alice does the following to use ECDSA to encrypt a secret message or key m_1 that only Bob can retrieve:

Choose a random integer $k \leq n - 1$
$Z = kQ_B = (x_z , y_z)$
$m_2 = E_{x_z}(m_1)$, where $E_{x_z}(m_1)$ is a symmetric encryption function that encrypts a message m_1 with key x_z.
$R = kG = (x_1 , y_1)$
$r = x_1 \bmod n$
$s = k^{-1}(H(m_2) + d_A.r) \bmod n$
Output (r, s)

When Bob receives m_2 and the signature (r, s) he verifies the signature using Alice's public key to make sure it originated from Alice then proceeds to extract m_1 as follows:

$R` = s^{-1}(H(m_2)G + rQ_A) = (x_1` , y_1`)$
If $r = x_1`$ mod n then accept (r, s) as valid for message m_2 else reject.
$Z` = d_BR` = (x_z` , y_z`)$
$x_z` = x_z$
$m_1 = D_{x_z}(m_2)$, where $D_{x_z}(m_2)$ is the reverse of E_{x_z} (the symmetric decryption function that decrypts the ciphertext m_2 with key x_z)

5. Conclusion

We have shown that a regular SETUP attack can be mounted on ECDSA. This enables a malicious manufacturer of black-box cryptosystems like smart card devices to implement such attacks to get exclusive access to the user's private key. The output of a dishonest device is indistinguishable from the output of an honest one. We have also shown how ECDSA can be used for encryption and key exchange.

References

[1] D. Johnson, A. Menezes, and S. Vanstone, "The elliptic curve digital signature algorithm (ECDSA)", *International Journal of Information Security*, Vol. 1, pp. 36–63, 2001.
[2] G. J. Simmons, "The Prisoner's Problem and the Subliminal Channel", *Crypto '83*, pp. 51-67, 1983.
[3] G. J. Simmons, "Subliminal Communication is Easy Using the DSA", *Eurocrypt '93*, pp. 218-232, 1993.
[4] A. Young, M. Yung, "Kleptography: Using Cryptography Against Cryptography", *Eurocrypt '97*, pp. 62-74, 1997.
[5] A. Young, M. Yung, "The Dark Side of Black-Box Cryptography or Should We Trust Capstone?", *Crypto '96*, pp. 89-103.
[6] S. Goldwasser, S. Micali, "Probabilistic Encryption", *J. Comp. Sys. Sci.* 28, pp. 270-299, 1984.
[7] A. Young, M. Yung, "The Prevalence of Kleptographic Attacks on Discrete-Log Based Cryptosystems", *Crypto '97*, pp. 264-276, 1997.