Invalid Curve Attack on the 5G SUCI Privacy Feature

Tobias Funke (tobias.funke@rub.de)
David Rupprecht (david@radix-security.com)

Radix Security

Making Mobile Networks Security Accessible.

Introduction

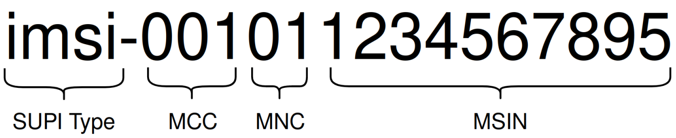
In a mobile network, each subscriber has a unique identifier called an International Mobile Subscriber Identity (IMSI). It was first introduced in 2G systems and is still used in 3G and 4G systems. The use of IMSIs in clear text over the radio link allows subscribers to be identified and tracked, leading to serious privacy implications. To address this issue, temporary identifiers have been introduced. These identifiers are randomly selected and frequently changed and are intended to be used in place of the IMSI. However, in certain situations the use of the IMSI instead of the TMSI is unavoidable. These situations can be exploited by malicious actors who can use an attack known as IMSI catching to force subscribers to reveal their IMSI. This renders the use of the TMSI obsolete.

With the introduction of 5G, the 3rd Generation Partnership Project (3GPP) has decided to take a new approach to address this privacy issue by encrypting the IMSI in cases where the use of the TMSI is unavoidable. In 5G, the SUPI is the new IMSI, a globally unique identifier assigned to each subscriber. While the Subscription Concealed Identifier (SUCI) is the privacy-preserving identifier that contains the concealed SUPI[1].

Technical Background

SUPI

There are different types of SUPIs defined in the 5G specification[2]. In the following, we only look at the IMSI-based one, which is equivalent to the previously mentioned IMSI. This type is a number containing up to 15 digits. The foremost three digits represent the Mobile Country Code (MCC). The next two or three digits define the Mobile Network Code (MNC) identifying the network operator. The remaining digits represent the individual subscriber of that particular operator, also known as the Mobile Subscriber Identification Number (MSIN)[3]. The following figure shows the structure of IMSI-based SUPIs using a fictional example.



[1] https://www.etsi.org/deliver/etsi_ts/133500_133599/133501/15.03.01_60/ts_133501v150301p.pdf, 6.12.1 Subscription Permanent Identifier

[2] https://www.etsi.org/deliver/etsi_ts/123000_123099/123003/16.03.00_60/ts_123003v160300p.pdf, 2.2A Subscription Permanent Identifier (SUPI)

[3] https://www.etsi.org/deliver/etsi_ts/123000_123099/123003/16.03.00_60/ts_123003v160300p.pdf, 2.2 Composition of IMSI

SUCI

A subscriber sends his SUPI concealed as a SUCI to the network as part of the authentication step. This identifier consists of six parts. The first part specifies the type of the containing SUPI. The second part specifies the Home Network Identifier. In the case of IMSI-based SUPIs, these are MCC and MNC. Next is the Routing Indicator, which the network uses for internal routing.

The last three parts are relevant to the Invalid Curve attack presented later. The Protection Scheme Id indicates which scheme was used by the subscriber to create the scheme output. The Home Network Public Key Id indicates which of the network operator public keys was used. The last part of the SUCI is the output of the selected Protection Scheme. The following figure shows the SUCI structure.

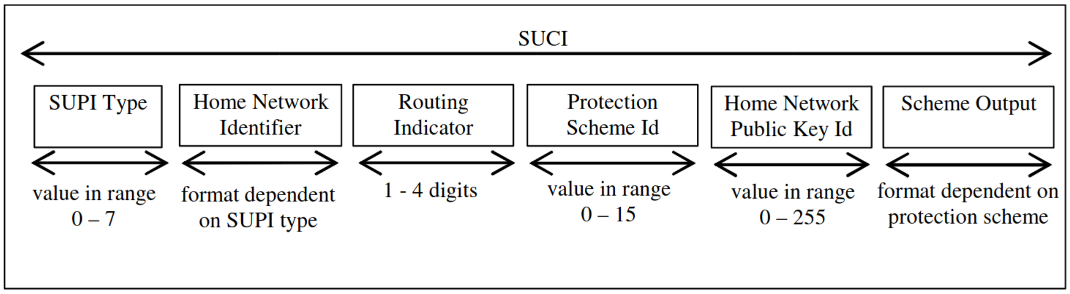


Figure 2.2B-1: Structure of SUCI

Since the MNC and MCC of the SUPI are directly included in the SUCI, only the MSIN is used as input for the selected Protection Scheme. Besides the fact that subscribers can still authenticate to the network by sending their MSIN in plaintext using the null-scheme[4], the novelty of 5G is its encrypted transmission. For this purpose, one of the two predefined Elliptic Curve Integrated Encryption Scheme (ECIES) profiles can be used[5]. These profiles (Profile A and Profile B) differ in their elliptic curve parameters; their main difference is the type of elliptic curve they use.

ECC

Before we look at the two ECIES profiles in detail, it is helpful to look at essential parts of elliptic curves and the Elliptic Curve Cryptography (ECC)[6] built on them. As a representation of elliptic curves used in ECC over the finite field $F_{p,}$ the short Weierstrass equation can be used[7]:

$$y^2 \equiv x^3 + ax + b \bmod p$$

[4] https://www.etsi.org/deliver/etsi_ts/133500_133599/133501/15.03.01_60/ts_133501v150301p.pdf, C.2 Null-scheme

[5] https://www.etsi.org/deliver/etsi_ts/133500_133599/133501/15.04.00_60/ts_133501v150400p.pdf, C.3 Elliptic Curve Integrated Encryption Scheme

[6] https://cryptobook.nakov.com/asymmetric-key-ciphers/elliptic-curve-cryptography-ecc, for ECC details

[7] https://safecurves.cr.yp.to/equation.html, Equations

All pairs (x, y) that satisfy the above equation are points that lie on the given curve and belong to the same abelian group with a point O at infinity as the identity element. The operations addition, doubling, negation, and scalar multiplication can be performed with these points[8], where the multiplication is just repeated addition, and all operations are calculated modulo p.

Addition: $P_1 + P_2 = (x_1, y_1) + (x_2, y_2) = (x_3, y_3) = P_3$
$x_3 = (y_2 - y_1)^2 / (x_2 - x_1)^2 - x_1 - x_2$
$y_3 = (2 * x_1 + x_2) * (y_2 - y_1) / (x_2 - x_1) - (y_2 - y_1)^3 / (x_2 - x_1)^3 - y_1$

Doubling: $2 * P_1 = 2 * (x_1, y_1) = (x_2, y_2) = P_2$
$x_2 = (3 * x_1^2 + a)^2 / (2 * y_1)^2 - x_1 - x_1$
$y_2 = (2 * x_1 + x_1) * (3 * x_1^2 + a) / (2 * y_1) - (3 * x_1^2 + a)^3 / (2 * y_1)^3 - y_1$

Negation: $-P_1 = -(x_1, y_1) = (x_2, y_2) = P_2$
$x_2 = x_1$
$y_2 = -y_1$

Multiplication: $n * P = P + \ldots + P = Q$

The difficulty to determine the n for two given points, P and Q (aka discrete logarithm problem of elliptic curves), is used to determine the private and public key pair for public key cryptography. A private key is a random number n, which leads to the public key Q by multiplication with a publicly known base point G:

$$n * G = Q$$

A key agreement between two parties is reached by multiplying their private key (n_A, n_B) with the other party's public key (Q_A, Q_B). The resulting point S is the shared secret between both parties.

$$n_A * Q_B = n_A * (n_B * G) = n_B * (n_A * G) = n_B * Q_A = S$$

Depending on the coefficients a, b, and the modulus p, not all curve points can be generated by multiplication with the base point. The reason for this is the existence of non-overlapping subgroups of points generated by different base points. The number of unique points that a base point can generate for any n is called order and those with a small order are called small order points. The security of subgroups generated by small order base points is weak, leading to "small-subgroup" attacks[9].

[8] https://www.hyperelliptic.org/EFD/g1p/auto-shortw.html
[9] https://www.rfc-editor.org/rfc/rfc2785

ECIES

The ECIES is an encryption method based on Elliptic Curve Cryptography (ECC) that can be used, for example, "to transmit a confidential message of arbitrary length."[10] The following figure visualizes the ECIES flow in the 5G context from a subscriber perspective.

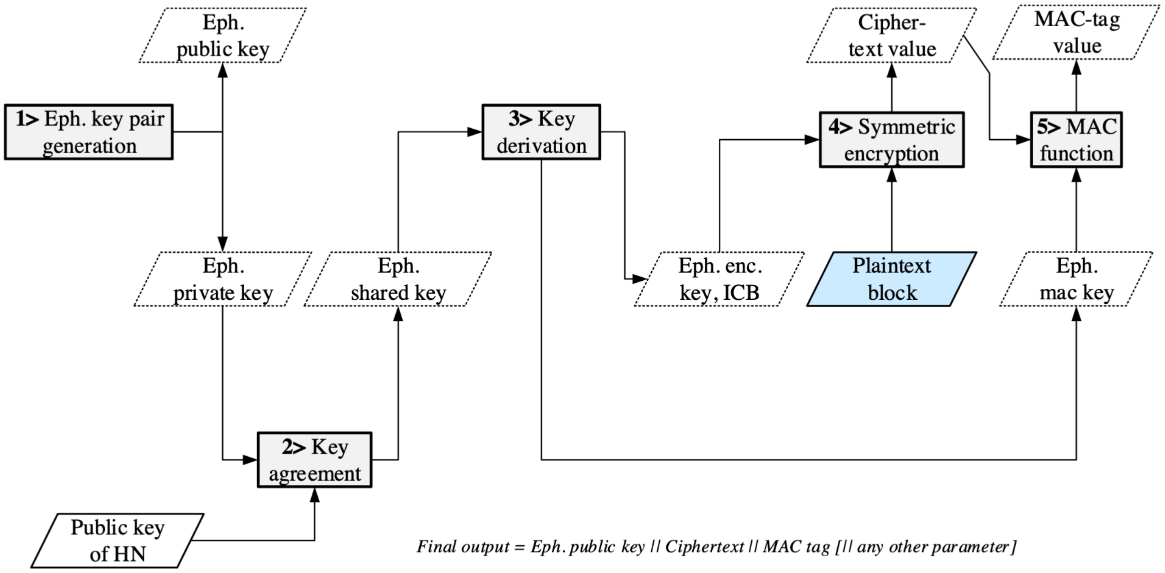


Figure C.3.2-1: Encryption based on ECIES at UE

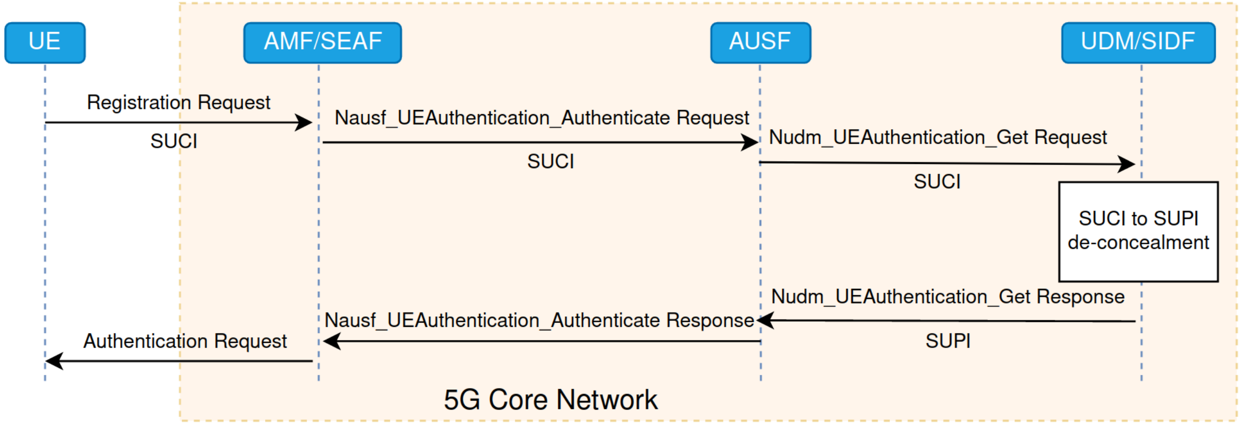
The network operator has stored its public key in the SIM card of its subscribers. When a subscriber authenticates himself to a network, he generates an ephemeral key pair [1]. He then uses the operator's public key and his fresh private key to compute a shared secret by using elliptic curve scalar multiplication [2]. Only the x coordinate of the secret is used as input to the ANSI-X9.63-KDF, which returns an Advanced Encryption Standard (AES) key and a Message Authentication Code (MAC) key [3]. With the AES key, the subscriber encrypts the MSIN of his SUPI [4]. In addition, he computes the MAC of the ciphertext via HMAC-SHA-256 using the MAC key [5][11].

Combining the subscriber's public key, the ciphertext, and the MAC results in the Scheme Output of the SUCI mentioned above, which is sent to the network. After receiving the SUCI, the network can calculate the shared secret using the included public key and its private key. After deriving the AES key from the secret, the MSIN can be decrypted. In addition, the integrity of the ciphertext is checked using the MAC.

[10] https://iacr.org/archive/pkc2003/25670211/25670211.pdf, 2.2 ECIES
[11] https://www.etsi.org/deliver/etsi_ts/133500_133599/133501/15.03.01_60/ts_133501v150301p.pdf, C.3.2 Processing on UE side

Oracle for successful decryption

Before moving on to the attack, we need a tool to determine if the network can adequately decrypt the sent SUCI. For this purpose, we consider the successful protocol flow, simplified in the following figure.


UE
AMF/SEAF
AUSF
UDM/SIDF
Registration Request
SUCI
Nausf_UEAuthentication_Authenticate Request
SUCI
Nudm_UEAuthentication_Get Request
SUCI
SUCI to SUPI de-concealment
Nudm_UEAuthentication_Get Response
SUPI
Nausf_UEAuthentication_Authenticate Response
Authentication Request
5G Core Network

Since the 5G specification describes the implementation, it is not the subscriber who is considered the communication party but the user equipment (UE) that carries out the implementation. In addition, the network is divided into individual services with different tasks.

The SUCI included in the Registration Request sent by the UE is relayed via the AMF/SEAF and AUSF services to the subscription identifier de-concealing function (SIDF) of the Unified Data Management (UDM) service[12]. This function decrypts the encrypted MSIN part of the SUCI according to the chosen ECIES profile and returns the SUPI[13].

After processing the respective responses by the intermediate services, the UE receives an Authentication Request[14]. This only happens if the de-concealment of the SUCI is successful. In all other cases, an error (e.g., MAC failure) can be assumed. This different behavior results in a successful decryption oracle.

It should also be noted that the same oracle is given by sending requests directly to the UDM from within the network. Instead of an Authentication Request, the HTTP status code 200 is used as an indicator for successful decryption.

[12] https://www.etsi.org/deliver/etsi_ts/133500_133599/133501/15.03.01_60/ts_133501v150301p.pdf, 6.1.2 Initiation of authentication and selection of authentication method
[13] https://www.etsi.org/deliver/etsi_ts/133500_133599/133501/15.03.01_60/ts_133501v150301p.pdf, 6.12.5 Subscription identifier de-concealing function (SIDF)
[14] https://www.etsi.org/deliver/etsi_ts/133500_133599/133501/15.03.01_60/ts_133501v150301p.pdf, 6.1.3.2.0 5G AKA

Invalid Curve Attack

An ECIES implementation can be vulnerable to an invalid curve attack if the check of whether a received public key is on the curve (aka public key validation) is not performed correctly. In particular, validating uncompressed public keys[15] is necessary because both coordinates (x, y) are specified. These can be used directly in calculating the shared secret without determining y beforehand.

The public key validation is not necessary for certain curves, and Curve25519 of Profile A is one of them. It uses only the x coordinate of a point to calculate the multiplication[16], which leads to the fact that public keys consist of only one coordinate, and only this coordinate is used during a transmission. In addition, the curve is designed to ensure all possible public keys lie on it[17].

Since these characteristics do not apply to the secp256r1 curve used in Profile B, it is a potentially vulnerable curve. For this curve, also known as P-256 or prime256v1, the parameters a, b, p, and G are defined as follows[18]:

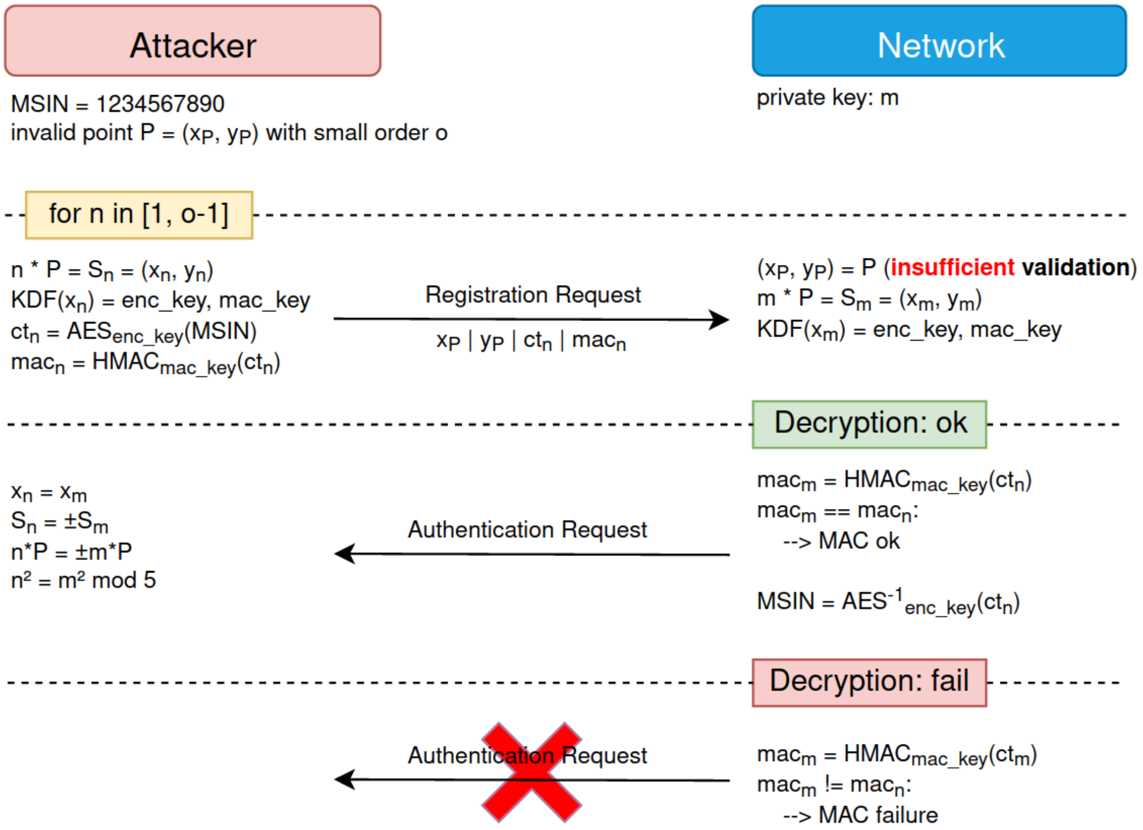
a = 0xffffffff00000001000000000000000000000000fffffffffffffffffffffffc
b = 0x5ac635d8aa3a93e7b3ebbd55769886bc651d06b0cc53b0f63bce3c3e27d2604b
p = 0xffffffff00000001000000000000000000000000ffffffffffffffffffffffff
G = (0x6b17d1f2e12c4247f8bce6e563a440f277037d812deb33a0f4a13945d898c296,
0x4fe342e2fe1a7f9b8ee7eb4a7c0f9e162bce33576b315ececbb6406837bf51f5)

As the b parameter does not influence the formulas of the operations listed above, they apply analogously for all curves with the same parameters except b. Based on this characteristic, all curves with a different b are called invalid curves to secp256r1, and points lying on one of these invalid curves are called invalid points.

By sending an invalid point as an uncompressed public key to the network, it is possible to force the shared secret calculation using an invalid curve. If the invalid point is a small order point, the scalar multiplication with the network's private key results in only a few possible values, as well as for the AES key derived from it. Due to the limited possibility of AES keys, brute force can be used to determine which shared secret the network has calculated.

For this purpose, an attacker creates a SUCI for every possible ciphertext and sends it to the network. The encrypted MSIN can only be successfully decrypted if the correct AES key is used. As an indicator for this, the oracle listed above is used, and after successfully finding the shared secret, a simple congruent equation can be constructed. The following figure shows the process described above.

[15] https://www.rfc-editor.org/rfc/rfc5480#section-2.2
[16] https://www.ietf.org/rfc/rfc7748.html#section-5
[17] https://cr.yp.to/ecdh.html#validate
[18] https://neuromancer.sk/std/secg/secp256r1


Attacker
Network
MSIN = 1234567890
invalid point $P = (x_P, y_P)$ with small order o
private key: m
for n in [1, o-1]
$n * P = S_n = (x_n, y_n)$
$KDF(x_n)$ = enc_key, mac_key
$ct_n = AES_{enc\_key}(MSIN)$
$mac_n = HMAC_{mac\_key}(ct_n)$
Registration Request
$x_P \mid y_P \mid ct_n \mid mac_n$
$(x_P, y_P) = P$ (insufficient validation)
$m * P = S_m = (x_m, y_m)$
$KDF(x_m)$ = enc_key, mac_key
Decryption: ok
$x_n = x_m$
$S_n = \pm S_m$
$n*P = \pm m*P$
$n^2 = m^2 \bmod 5$
Authentication Request
$mac_m = HMAC_{mac\_key}(ct_n)$
$mac_m == mac_n$:
--> MAC ok
$MSIN = AES^{-1}_{enc\_key}(ct_n)$
Decryption: fail
Authentication Request
$mac_m = HMAC_{mac\_key}(ct_m)$
mac_m != mac_n:
--> MAC failure

The attacker repeats this process with more invalid small order points resulting in several more equations. The entire private key can be calculated using the Chinese Remainder Theorem (CRT) using only relative prime orders[19].

The following algorithm shows the steps to recover a private key n of Profile B if the public key validation is not correctly implemented:

```
u = 1
p = 0xffffffff00000001000000000000000000000000ffffffffffffffffffffffff
equation_set = {}
    1. Choose an invalid point Q with known small order o, relatively prime to previous
       successful orders.
    2. for x in [1, o-1]
        a. Generate a SUCI by using Q as the attacker's public key and x as the private key
           of the network and send it to the network.
        b. If the decryption fails (no Auth. Req. received), go to 2.
        c. If the decryption is successful (Auth. Req. received), go to 3.
    3. Every x failed, go to 1.
    4. Add $n^2 \equiv x^2 \bmod o$ to equation_set and update $u = u * o$.
```

[19] https://artofproblemsolving.com/wiki/index.php/Chinese_Remainder_Theorem

5. If u ≤ p², go to 1.
6. n = sqrt(CRT(equation_set))

A malicious actor can gain every private key for Profile B from a network with the given algorithm. If a SUCI was generated by using the public key of one of those private keys, it could be easily decrypted. For example, the attacker only needs to eavesdrop on the initial Registration Request of the subscriber and use the gained private key to decrypt the transmitted SUCI.

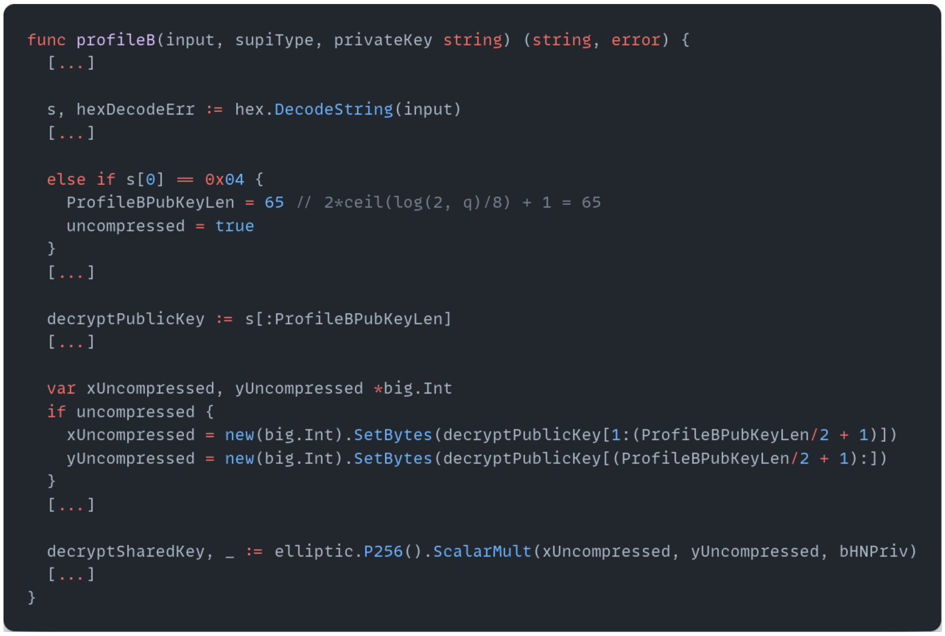
Since the success of this attack is explicitly dependent on the implementation, for example, depending on how the public key validation is performed, the algorithm may need to be adapted for a specific implementation.

free5GC

We examined three open-source 5G implementations (OpenAir CN 5G[20], Open5GS[21], free5GC[22]) to evaluate how practical such an invalid curve attack would be. The first two had no implementation of the ECIES profiles at the time of the investigation.

The third implementation was vulnerable to the invalid curve attack described above because both the function profileB[23] and the used Go library crypto/elliptic (< go1.19[24]) do not validate uncompressed public keys before calculating the shared secret. The following screenshot shows the vulnerable profileB function reduced to the essential lines of code.

[20] https://gitlab.eurecom.fr/oai/cn5g
[21] https://github.com/open5gs/open5gs
[22] https://github.com/free5gc/free5gc
[23] https://github.com/free5gc/udm/blob/main/pkg/suci/suci.go#L221-L303
[24] https://tip.golang.org/doc/go1.19#minor_library_changes, crypto/elliptic



```
func profileB(input, supiType, privateKey string) (string, error) {
  [...]

  s, hexDecodeErr := hex.DecodeString(input)
  [...]

  else if s[0] == 0x04 {
    ProfileBPubKeyLen = 65 // 2*ceil(log(2, q)/8) + 1 = 65
    uncompressed = true
  }
  [...]

  decryptPublicKey := s[:ProfileBPubKeyLen]
  [...]

  var xUncompressed, yUncompressed *big.Int
  if uncompressed {
    xUncompressed = new(big.Int).SetBytes(decryptPublicKey[1:(ProfileBPubKeyLen/2 + 1)])
    yUncompressed = new(big.Int).SetBytes(decryptPublicKey[(ProfileBPubKeyLen/2 + 1):])
  }
  [...]

  decryptSharedKey, _ := elliptic.P256().ScalarMult(xUncompressed, yUncompressed, bHNPriv)
  [...]
}
```

To exploit the vulnerability, we used SAGE[25] to precompute points on arbitrary invalid curves with small prime order and UERANSIM[26] to simulate a subscriber's UE and the radio link.

Our proof of concept makes it possible to recover a private key via a non-optimized attack within ~4 hours. Thereby ~6000 Registration Requests and ~70 small order points are used.

Mitigations

In general, an attack on Profile B can be prevented by using Profile A exclusively. Since both profiles must be implemented according to the 5G specification[27], deactivating one of the ECIES profiles would not comply with the specification and possibly reduce backward compatibility.

In detail, it is necessary to validate whether the (uncompressed) public key sent by the subscriber lies on the secp256r1 curve. In addition, monitoring can detect a possible attack if many MAC failures occur. Test cases should be created for the implementation to verify the public key validation in the long term.

Regarding the free5GC implementation, either a public key validation for uncompress points should be implemented, or the required go version should be upgraded from go1.14 to at least go1.19.

[25] https://www.sagemath.org/
[26] https://github.com/aligungr/UERANSIM
[27] https://www.etsi.org/deliver/etsi_ts/133500_133599/133501/15.03.01_60/ts_133501v150301p.pdf, C.3.4.1 and C.3.4.2: "The ME and SIDF shall implement this profile."

Conclusion

The invalid curve attack on the 5G SUCI privacy feature highlights the need for robust and secure implementation practices for 5G networks. While 3GPP's approach to encrypting the IMSI in 5G is a step towards enhancing subscriber privacy, this research shows that the scheme introduces new vulnerabilities that can be exploited by malicious actors to subvert the scheme. To ensure the effectiveness of the SUCI privacy feature, network operators and vendors must prioritize the proper implementation of their UDM. By addressing these issues, the 5G ecosystem can continue to evolve and deliver on its promise of improved connectivity and security for its subscribers.

Author	Comment	Date
Tobias Funke	Initial Version	Apr 26, 2023