Private Key Recovery on Bitcoin with Duplicated Signatures

Ju-Seong Ko[1] and Jin Kwak[2*]
[1] ISAA Lab., Department of Computer Engineering, Ajou University, Suwon, Republic of Korea
[e-mail: jsko.isaa@gmail.com]
[2] Department of Cyber Security, Ajou University, Suwon, Republic of Korea
[e-mail: security@ajou.ac.kr]
*Corresponding author: Jin Kwak

Received September 30, 2019; revised December 4, 2019; accepted January 5, 2020; published March 31, 2020

Abstract

In the modern financial sector, interest in providing financial services that employ blockchain technology has increased. Blockchain technology is efficient and can operate without a trusted party to store all transaction information; additionally, it provides transparency and prevents the tampering of transaction information. However, new security threats can occur because blockchain technology shares all the transaction information. Furthermore, studies have reported that the private keys of users who use the same signature value two or more times can be recovered. Because private keys of blockchain identify users, private key leaks can result in attackers stealing the ownership rights to users' property. Therefore, as more financial services use blockchain technology, actions to counteract the threat of private key recovery must be continually investigated. Private key recovery studies are presented here. Based on these studies, duplicated signatures generated by blockchain users are defined. Additionally, scenarios that generate and use duplicated signatures are applied in an actual bitcoin environment to demonstrate that actual bitcoin users' private keys can be recovered.

Keywords: Blockchain, Bitcoin, ECDSA, Private Key Recovery, Duplicated Signatures

http://doi.org/10.3837/tiis.2020.03.020 **ISSN : 1976-7277**

1. Introduction

An important issue in the current financial sector is blockchain technology, which is based on fintech, i.e. finance and IT (Information Technology) combined [1]. In blockchain technology, all members have the same information linked in a chain form, which prevents data tampering [2].

Conventional centralized financial systems determine the ownership rights for property using ledgers that are managed by trusted third parties. However, a large amount of social capital must be expended to establish, maintain, and guarantee the reliability of the trusted third party. As such, blockchain-style financial systems have garnered interest. These systems have the advantage of not requiring a trusted third party because all users record and manage the ledger [1].

However, security problems exist when using blockchain technology in the financial sector. An advantage of blockchain is security; however, this is different from the security that is required by the financial sector. Blockchain security aims to disable the falsification of past transaction data or disable double payments; however, in the financial sector, security implies that current users' property is not hacked. Therefore, the high security of the blockchain approach is not suitable for the security required by the financial sector [3]. In actual attacks on blockchain-based cryptocurrency, the main attack targets are currency exchanges and wallets rather than past transactions [4].

Recent studies have reported that the private keys of transaction principals can be leaked via signature values used in major cryptocurrency bitcoins [5][6]. This threat is due to the blockchain's property of sharing transaction information and transaction signature values, and it can occur regardless of the security strength provided by the blockchain. However, because the property is owned by the owner of the private key and not the user, an attacker can steal the property if the user's private key is leaked.

In this study, scenarios in which private keys can be recovered are applied to prove that users' private keys can be recovered in an actual bitcoin environment. In Section 2, centralized and blockchain financial systems are compared and the operating principles of ECDSA (Elliptic Curve Digital Signature Algorithm), bitcoin signatures, and previous studies are discussed. Additional threats that are not discussed in previous studies are analyzed in Section 3. Duplicated signatures that are determined by blockchain platform users are defined in Section 4. Additionally, two scenarios in which duplicated signatures are generated and users' private keys are recovered are proposed. In Section 5, the two proposed scenarios are used to demonstrate the process of recovering actual bitcoin private keys, and Chapter 6 presents the conclusions of this paper.

2. Related Work

2.1 Digital Signature

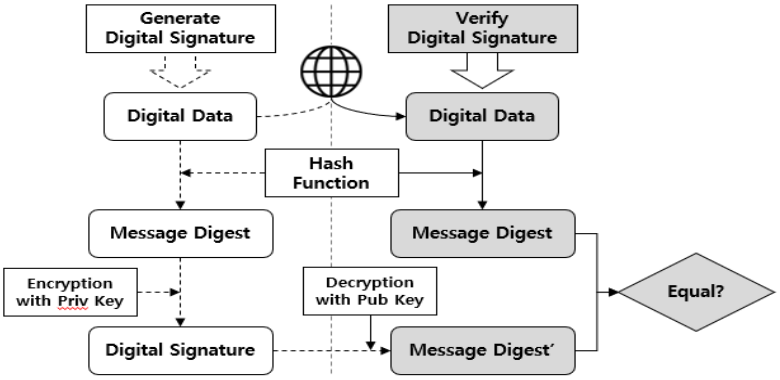


Fig. 1. Operation of Digital Signature

Digital signatures are used to secure electronic transactions as attached values to guarantee the data source, integrity, and non-repudiation. **Fig. 1** shows the operation of a digital signature. To view the message, the receiver decrypts the signature and then recalculates the hash value. If the hash value matches, the authenticity of the message and the sender are proven. The integrity of the message can be confirmed through the hash function, which produces different results if the text has been changed if even slightly. The sender can be confirmed because only the correct sender can generate a signature with the correct private key [7].

2.2 ECDSA

ECDSA is an implementation of the DSA (the National Institute of Standards and Technology's signature algorithm) on an elliptic curve. The elliptic curve and domain parameters for ECC calculations were predefined. In an ECDSA, a mathematically related private and public key pair are used to generate and verify the digital signature [8]. This section describes the process of generating and verifying ECDSA signatures.

2.2.1 Generating ECDSA Signature

The signer's public key Q is calculated as Q = dG through a private key d and a generator G. Based on the discrete logarithm problem, d cannot be obtained from G and Q. Shown below is the process of using a private key to sign a message in an ECDSA [9]. If the message is signed, the signature pair (r, s) is generated.

① Select a nonce: k $(1 \leq k \leq n-1)$
② $(x_1, y_1) = kG$
③ r = $x_1 \, mod \, n$
④ e = SHA256(message)
⑤ s = $k^{-1}(e + dr) mod \, n$

Result: Signature (r, s)

2.2.2 Verifying ECDSA Signature

Shown below is the process of using a public key to verify the validity of a signature [9]. Ultimately, if 'r' and 't' are the same, the message can be confirmed as signed with a private key that is mathematically related to the public key.

① $e = SHA256(message)$
② $a = s^{-1}e \ mod \ n$
③ $b = s^{-1}r \ mod \ n$
④ $(x_2, y_2) = aG + bQ$
⑤ $t = x_2 \ mod \ n$
Result: if 't' is equal to 'r', verification is successful

2.3 Additional Cryptographic Algorithms

Supplementary cryptological algorithms used for the ECDSA include a nonce generation algorithm and a hash function.

Nonce is important in cryptography for a variety of security applications, and nonce generation requires randomness and unpredictability [10]. **Table 1** shows the randomness and unpredictability required for nonce generation.

Table 1. Requirements for Nonce Generation

Requirements	Contents
Randomness	- The distribution of the sequence's bits is uniform. - A partial sequence of a sequence can't be estimated from another sequence.
Unpredictable	- It must not be possible to predict the next value of sequences.

The hash function receives a message of random length as input and produces a hash value of a fixed length. The hash function serves to generate unchangeable proof values for the message to provide integrity such that it message errors or falsifications can be detected. When long messages are signed, a short hash value is calculated for the entire message, and this value is signed. For the hash function to be secure, it must be difficult to find collisions [11]. **Table 2** shows the strength of the SHA-256 used in bitcoin.

Table 2. Strength of SHA-256

Strength of SHA-256	length(bits)
Collision Resistance Strength	128
Preimage Resistance Strength	256
Second Preimage Resistance Strength	201-256

2.4 Bitcoin

Bitcoin is a system designed such that users can trust each other without a trusted third party, and it was proposed in 2009 by Satoshi Nakamoto [14]. Because all members have the same ledger, tampering is not possible. When new transactions occur, new blocks are

connected to the existing ledger block in the form of a chain [2]. Because bitcoin is an electronic currency that only moves between transaction parties, it does not require a trusted party, and P2P networks are used to prevent double payments. Furthermore, the anonymity of members can be ensured by maintaining the anonymity of the public keys [14]. As of June 9, 2019, bitcoin contains 579,952 blocks in its chain [15]. All of the block data included in bitcoin can be synchronized with the bitcoin core [16].

The header of a bitcoin block contains a magic number that distinguishes the block, block size, block version, previous block's hash, current block's hash, time of block creation, target that shows the block creation level of difficulty, and the nonce for proof-of-work. Furthermore, the number of transactions and the transaction data are included after the header. **Table 3** shows the structure of a bitcoin block [17].

Table 3. Structure of Bitcoin Block

Area	Components of Block	Size
Header	Magic Number	4 bytes
	Block Size	4 bytes
	Version	4 bytes
	Previous Block Hash	32 bytes
	Block Hash	32 bytes
	Time	4 bytes
	Target	4 bytes
	Nonce	4 bytes
Transaction	Transaction Counter	1- 9 bytes
	Transaction Data	Up to 1 MB

Bitcoin transactions comprises the transaction's version, input, and output. The input includes the input number, previous transaction's hash, previous output's index, scriptSig length, the scriptSig that contains the signature data, and a 4-byte sequence. The output includes the output number, the value that is the coin amount in Satoshi units, scriptPubKey length, scriptPubKey that contains the sender's public key hash data, and locktime at which the transaction was created. **Table 4** shows the structure of a bitcoin transaction [17].

Table 4. Structure of Bitcoin Transaction

Area	Components of Transaction	Size
Header	Version	4 bytes
Input	Number of Inputs	1 byte
	Previous Transaction Hash	32 bytes
	Previous Output Index	4 bytes
	Script Length	1 byte
	ScriptSig	"Script Length" bytes
	Sequence	4 bytes
Output	Number of Outputs	1 byte
	Value	8 bytes
	Script Length	1 byte
	ScriptPubkey	"Script Length" bytes
	Locktime	4 bytes

2.5 Nonce Reusing Problems

Nicolas T. Courtois et al. [5] analyzed a problem in which ECDSA private keys can be recovered when nonces that have already been used by bitcoin are reused. When previously used nonces are used, the same r value of signature is generated without regard to the signer's private key and message (Order 1–3 in Section 2.2.1). Because the nonces that were used in the signatures can be found, private key recovery attacks can be performed via the nonces used in the signatures. The Courtois study classified three types of private key recovery attacks that can occur according to the number of reused nonces and the number of users. Shown below are descriptions of the classified attacks.

• Attack on ECDSA with Bad RNG

When two users use the same nonce, the two users can calculate each other's private keys.

• Bad RNG ECDSA Same User Attack

When one user uses the same nonce two or more times, anyone can calculate the user's private key.

• Attack with Two Bad Random Events

When two users each use two of the same nonces, anyone can calculate the two users' private keys.

2.6 Occurrences of Same Signatures on Bitcoin

Michael Brengel et al. [6] proposed a specific recovery process for private key recovery attacks that are classified according to the study by Nicolas T. Courtois et al. [5]. In the study, data in which the same r value of signature occurred are presented. **Table 5** shows some of the r values that were used two or more times and the number of occurrences, as reported in the paper.

Table 5. Five Most Frequent Occurrences of R Values

R Values	Occurrences
0x000000000000000000000003b78ce563f89a0ed9414f5aa28ad0d96d6795f9c63	2,276,718
0x00006fcf15e8d272d1a995af6fcc9d6c0c2f4c0b6b0525142e8af866dd8dad4b	7,895
0x1206589b08a84cb090431daa4f8d18934a20c8fa52ad534c5ba0abb3232be1d9	265
0x79be667ef9dcbbac55a06295ce870b07029bfcdb2dce28d959f2815b16f81798	251
0x2ef0d2ae4c49c37703ba16a3126e27763e124ff3338fb93577ed7bd79ed0d19e	91

Shown below are two situations in which an attack can recover a user's private key as well as the recovery processes, as reported by Michael Brengel et al. [6]. Private key recovery is accomplished by modifying the formula for the signature value as in Order 5 of Section 2.2.1.

2.6.1 Bad RNG ECDSA Same User Attack

When User A uses the same key two or more times, the attacker can calculate the user's private key. Shown below are the processes of signature generation, in which one user uses the same nonce, and the attacker's private key recovery process.

Step 1. User A uses nonce k to generate two signatures

$$s_1 = k^{-1}(e_1 + rd_A)\ mod\ \text{n}, \qquad s_2 = k^{-1}(e_2 + rd_A)\ mod\ \text{n}$$

Step 2. Two formulas are established for k

$$s_1 s_2 \text{k} = s_2(e_1 + rd_A)\ mod\ \text{n}, \qquad s_1 s_2 \text{k} = s_1(e_2 + rd_A)\ mod\ \text{n}$$

Step 3. The first formula in Step 2 is subtracted from the second

$$s_2(e_1 + rd_A) - s_1(e_2 + rd_A) = 0\ mod\ \text{n}, \text{then}\ \ \text{r}(s_1 - s_2)d_A = s_2 e_1 - s_1 e_2\ mod\ \text{n}$$

Step 4. Step 3's formula is established for d_A and calculated

$$d_A = (s_2 e_1 - s_1 e_2)[r(s_1 - s_2)]^{-1}$$

2.6.2 Attack with Two Bad Random Events

When two users each use two nonces one or more times, the attacker can calculate the two users' private keys. Shown below are the processes of signature generation, in which the two users use the two nonces one time or more, and the attacker's private key recovery process. The process below shows the steps by which one user's private key is recovered. The other user's private key is recovered by the same process.

Step 1. User A uses nonces k_1 and k_2 to generate two signatures

$$s_{1,A} = {k_1}^{-1}\left(e_{1,A} + r_1 d_A\right) mod\ \text{n}, \qquad s_{2,A} = {k_2}^{-1}\left(e_{2,A} + r_2 d_A\right) mod\ \text{n}$$

Step 2. User B uses nonces k_1 and k_2 to generate two signatures

$$s_{1,B} = {k_1}^{-1}\left(e_{1,B} + r_1 d_B\right) mod\ \text{n}, \qquad s_{2,B} = {k_2}^{-1}\left(e_{2,B} + r_2 d_B\right) mod\ \text{n}$$

Step 3. The four formulas used in Steps 1 and 2 are established for k_1 and k_2

$$k_1 = \left(e_{1,A} + r_1 d_A\right){s_{1,A}}^{-1}\ mod\ \text{n}, \qquad k_1 = \left(e_{1,B} + r_1 d_B\right){s_{1,B}}^{-1}\ mod\ \text{n}$$

$$k_2 = \left(e_{2,A} + r_2 d_A\right){s_{2,A}}^{-1}\ mod\ \text{n}, \qquad k_2 = \left(e_{2,B} + r_2 d_B\right){s_{2,B}}^{-1}\ mod\ \text{n}$$

Step 4. The two formulas in Step 3 are established for d_B

$$d_B = \left(\left(e_{1,A} + r_1 d_A\right){s_{1,A}}^{-1} s_{1,B} - e_{1,B}\right){r_1}^{-1}\ mod\ \text{n}$$

$$d_B = \left(\left(e_{2,A} + r_2 d_A\right){s_{2,A}}^{-1} s_{2,B} - e_{2,B}\right){r_2}^{-1}\ mod\ \text{n}$$

Step 5. The second formula in Step 4 is subtracted from the first formula

$$\left(r_2 s_{1,B} s_{2,A} e_{1,A} + r_1 r_2 s_{1,B} s_{2,A} d_A - r_2 s_{1,A} s_{2,A} e_{1,B}\right) - \left(r_1 s_{1,A} s_{2,B} e_{2,A} + r_1 r_2 s_{1,A} s_{2,B} d_A - r_1 s_{1,A} s_{2,A} e_{2,B}\right) = 0\ mod\ \text{n}$$

Step 6. The formula in Step 5 is established for d_A and calculated

$$\left(s_{1,A} s_{2,B} - s_{1,B} s_{2,A}\right) r_1 r_2 d_A = \left(r_2 s_{2,A} s_{1,B} e_{1,A} - r_2 s_{2,A} e_{1,B} - r_1 s_{1,A} s_{2,B} e_{2,A} + r_1 s_{1,A} e_{2,B}\right) mod\ \text{n}$$

$$d_A = \left[r_1 s_{1,A}\left(e_{2,B} - s_{2,B} e_{2,A}\right) - r_2 s_{2,A}\left(e_{1,B} - s_{1,B} e_{1,A}\right)\right]\left[r_1 r_2\left(s_{1,A} s_{2,B} - s_{1,B} s_{2,A}\right)\right]^{-1}\ mod\ \text{n}$$

The study reported that the same r values of signatures can occur not only in cases where the same nonce was reused, but also in cases in which the used nonces have a complementary relationship with previously used nonces. Owing to the properties of the elliptic curve, when nonces k and -k (inverse of addition for n) are used in Order 1-2 of Section 2.2.1, kG =

(x_1, y_1) and $-\mathrm{kG} = (x_1, -y_1)$. This yields the same r value of signature in Order 3 of Section 2.2.1.

3. Analyses of Inverse Nonce Problems

Michael Brengel et al. [6] proposed that the same r values of signatures occur because the same nonces are used and nonces that complement each other are used. The study analyzed the private key recovery process that can occur when the same nonce is used. Therefore, the private key recovery process that can occur when using nonces that complement each other is analyzed in this section.

When the same nonces are used, two or more formulas are subtracted from the signature values during private key recovery. However, when two nonces that complement each other are used, the formulas are added during private key recovery.

3.1 Inverse RNG ECDSA Same User Attack

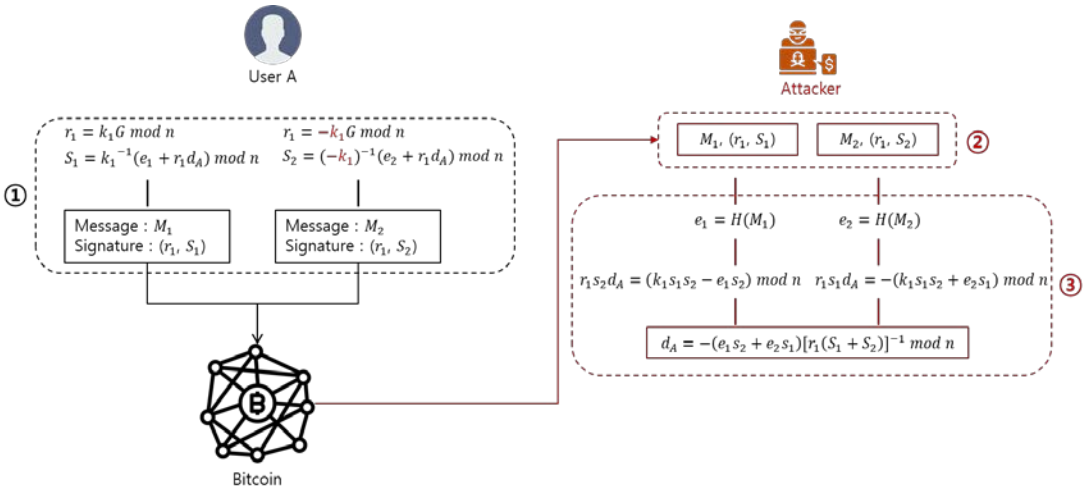


Fig. 2. Processes of Inverse RNG ECDSA Same User Attack

In this threat, the private key recovery for when the user uses the same nonces, as mentioned in Section 2.7.1, is changed into a private key recovery for when the user uses nonces that complement each other. **Fig. 2** shows the private key recovery process when the duplicate nonce k that was used in Section 2.7.1 is changed to k_1, $-k_1$.

Step 1. User A uses nonces k_1 and $-k_1$ to generate two signatures

Step 2. The two transactions' messages and signatures are collected from the bitcoin block

Step 3. User A's private key is calculated from the collected data

3.2 Attack with Two Inverse Random Events

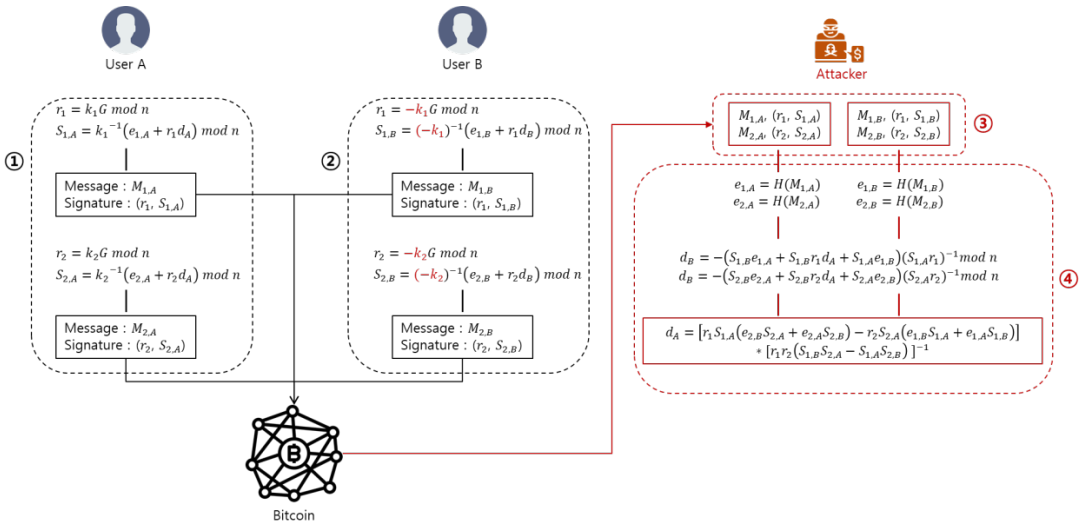


Fig. 3. Processes of Attack with Two Inverse Random Events

In this threat, the private key recovery for when two users have reused two nonces, as mentioned in Section 2.7.2, is changed into the private key recovery for when nonces that complement each other are used. When User A has used nonces k_1 and k_2 and User B uses one pair from among $(k_1, -k_2)$, $(-k_1, k_2)$, and $(-k_1, -k_2)$, private key recovery is possible. **Fig. 3** shows the private key recovery when User A uses nonces k_1 and k_2 and User B uses $(-k_1, -k_2)$.

Step 1. User A uses nonces (k_1, k_2) to generate two signatures
Step 2. User B uses nonces $(-k_1, -k_2)$ to generate two signatures
Step 3. The two transactions' messages and signatures are collected from the bitcoin block
Step 4. The private keys of Users A and B are calculated from the collected data

Table 6 shows the threats by which anyone can recover a user's private key through a bad/inverse RNG.

Table 6. Four Recovery Attacks with Bad/Inverse RNG

No.	Attack Types
1	Bad RNG ECDSA Same User Attack
2	Inverse RNG ECDSA Same User Attack
3	Attack with Two Bad Random Events
4	Attack with Two Inverse Random Events

4. Proposed Private Key Recovery Scenarios

The threat of private key recovery can occur because the blockchain shares all transaction information. Therefore, in this section, duplicated signatures that can be used in private key recovery threats is defined, and scenarios in which private keys can be recovered because of duplicated signatures are proposed.

4.1 Definition of Duplicated Signatures

Duplicated signatures are r values of signatures for which anyone can find the nonce that was used to generate a bitcoin transaction's signature. As described in Section 2.2, nonces that are used in signature generation are kept secure based on the discrete logarithm problem. However, if a case belongs to one of the attack types in **Table 6**, anyone can obtain the nonce that corresponds to the r value of signature, thus resulting in duplicated signatures. Therefore, duplicated signatures are not selected beforehand but depend on the user that is using the blockchain. Moreover, as the number of signatures that are used in bitcoin increases, the number of duplicated signatures can increase.

4.2 Recovery Scenarios with Duplicated Signatures

Cases in which private keys can be recovered through duplicated signatures are divided into cases of generating duplicated signatures and cases of using duplicated signatures. These two cases are explained below.

4.2.1 Recovery of Private Key Generating Duplicated Signatures

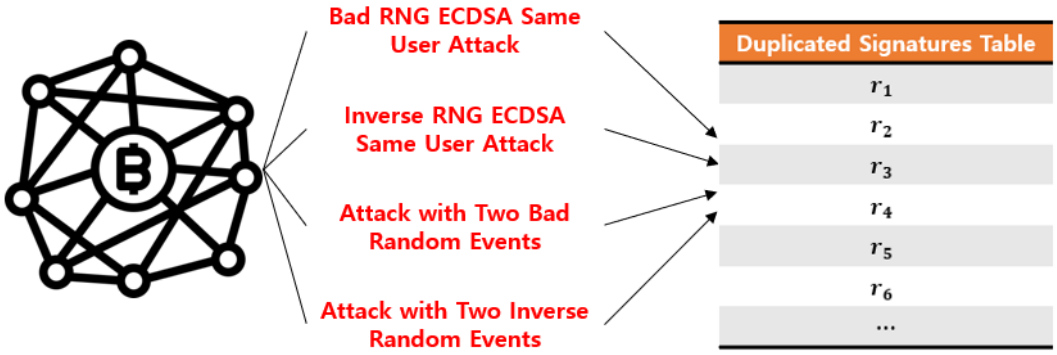


Fig. 4. Cases of Generating Duplicated Signatures

Fig. 4 shows cases in which a user's private key can be recovered when the user uses a duplicated nonce or nonces that complement each other, as in the attack types shown in **Table 6**. Furthermore, the r value of signature used by the user is a duplicated signature. **Figs. 2** and **3** show the recovery of user's nonces and private keys that belong to these attack types.

4.2.2 Recovery of Private Key Using Duplicated Signatures

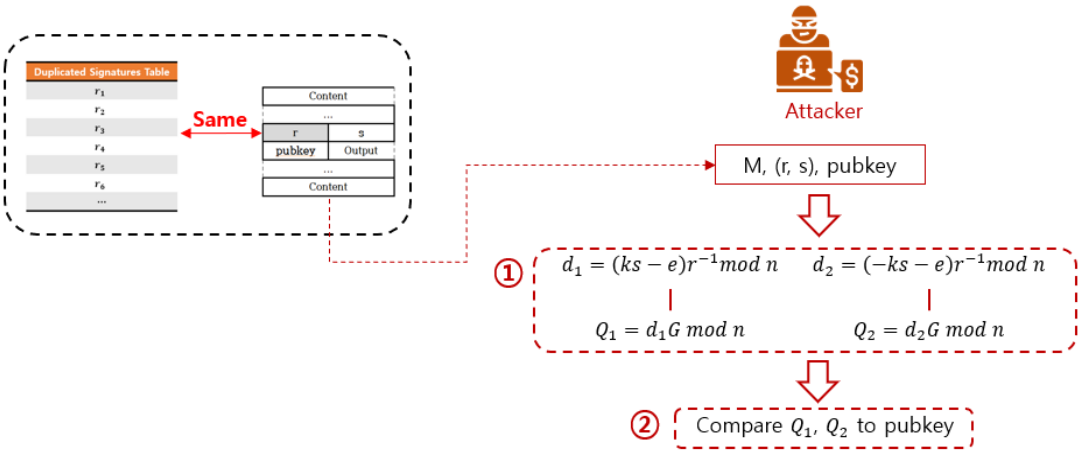


Fig. 5. Cases of Using Duplicated Signatures

Fig. 5 shows the private key recovery of users who use duplicated signatures. The first step is to verify whether the signature that was generated by the user is one of the duplicated signatures that were generated by the attack types in **Table 6**. If the signature is one of the duplicated signatures, the nonce that was used to generate the signature can be calculated, and the nonce can be used to recover the user's private key. Because two nonces can generate one signature value, two private keys and two public keys are generated with the nonces. Therefore, the two generated public keys and the user's actual public key can be compared to determine the user's actual private key. Shown below is a description of the private key recovery from **Fig. 5**.

Step 1. New public keys Q_1 and Q_2 are generated from the two recovered private keys d_1 and d_2

Step 2. The generated public keys Q_1 and Q_2 are compared to the user's public key to determine the user's private key

Even if a signature is not a duplicated signature when it is generated, the user's private key can be recovered if the r value of signature becomes a duplicated signature. Furthermore, recovery of private keys that generate duplicated signatures can only occur if the user generates two or more signatures, but recovery of private keys that use duplicated signatures can occur even if the user generates only one signature.

5. Bitcoin Private Key Recovery

The actual private key recovery by applying the proposed private key recovery scenario is described in this section. The process of synchronizing bitcoin block data via the bitcoin core and the recovery of users' private keys from synchronized block data are described. **Table 7** details the environment in which this study was performed.

Table 7. Analysis Environment

Processes	Environment	Details
Synchronizing block data	OS	Windows 10, 64bit
	RAM	16 GB
	Tool	Bitcoin Core v0.18.0 (64-bit)
All other processes	OS	Ubuntu 16.04.6 LTS
	RAM	128 GB
	Tool	Python 2.7.12

5.1 Collecting Data from Bitcoin

This section includes the processes of synchronizing bitcoin blocks and extracting message and signature values from synchronized block data.

5.1.1 Synchronizing Block Data of Bitcoin

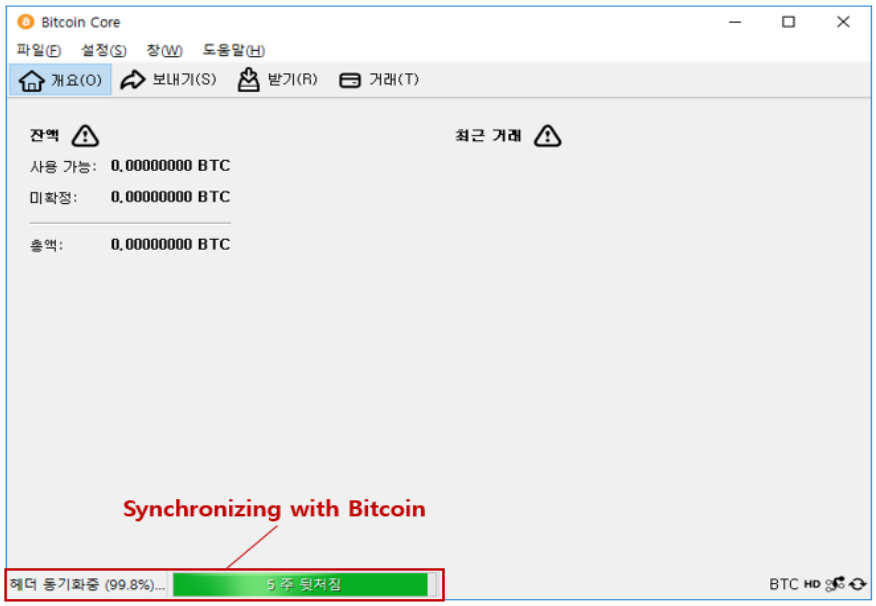


Fig. 6. Synchronizing Block Data with Bitcoin Core

In this study, the bitcoin core was used to participate in bitcoin and synchronize the entire bitcoin block data, as shown in **Fig. 6**. The synchronized block data was saved as a file named "blkxxxxx.dat". It was synchronized up to the 481,614th chain included in the block on August 22, 2017, and the total size of the synchronized data was approximately 121 GB.

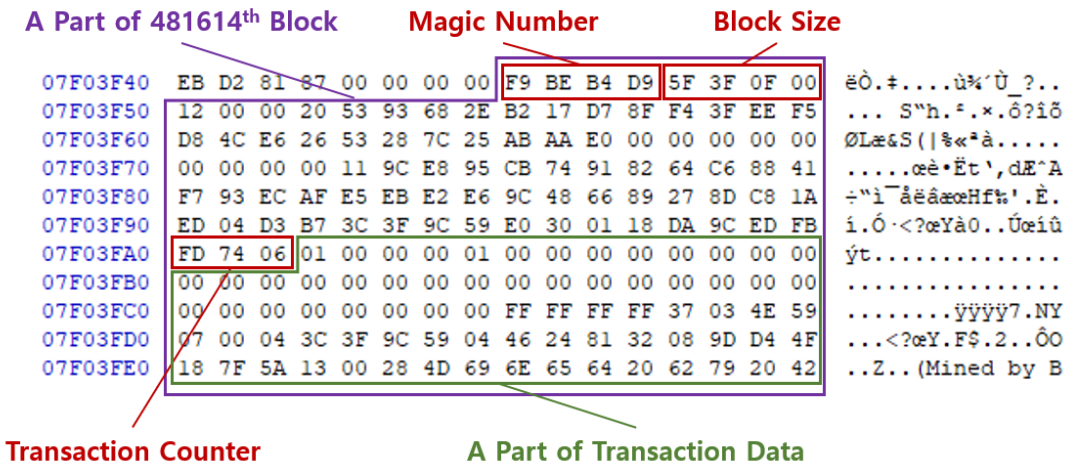


Fig. 7. A Part of 'blk00974.dat'

Fig. 7 shows a part of the "blk00974.dat" file that has more than 130 block data including the 481,614th block. The total size of the 481,614th block is 999,263 (0xF3F5F) bytes, and it includes data on 1,652 (0x674) transactions.

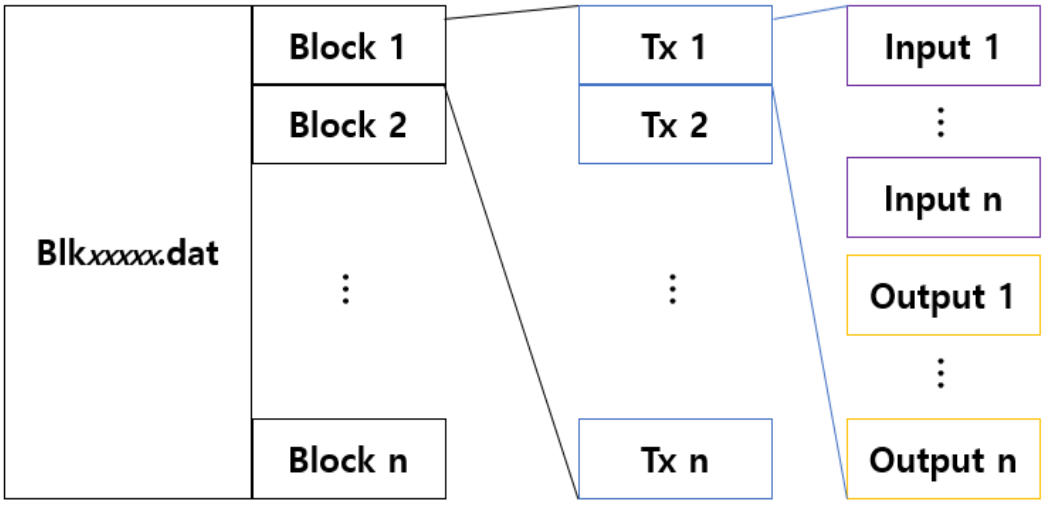


Fig. 8. Structure of Bitcoin Block Data

Fig. 8 shows the blocks and signature data that are stored in each "blkxxxxx.dat" file. During signature value extraction, the entire data file must be divided into blocks, each block into transactions, and each transaction into an input and output. The ultimately divided inputs include signature values and public keys, and the outputs include the public keys' hash values. Therefore, each transaction is a message, and the input can be analyzed to extract the signature value and public key.

5.1.2 Extracting Message from Block Data

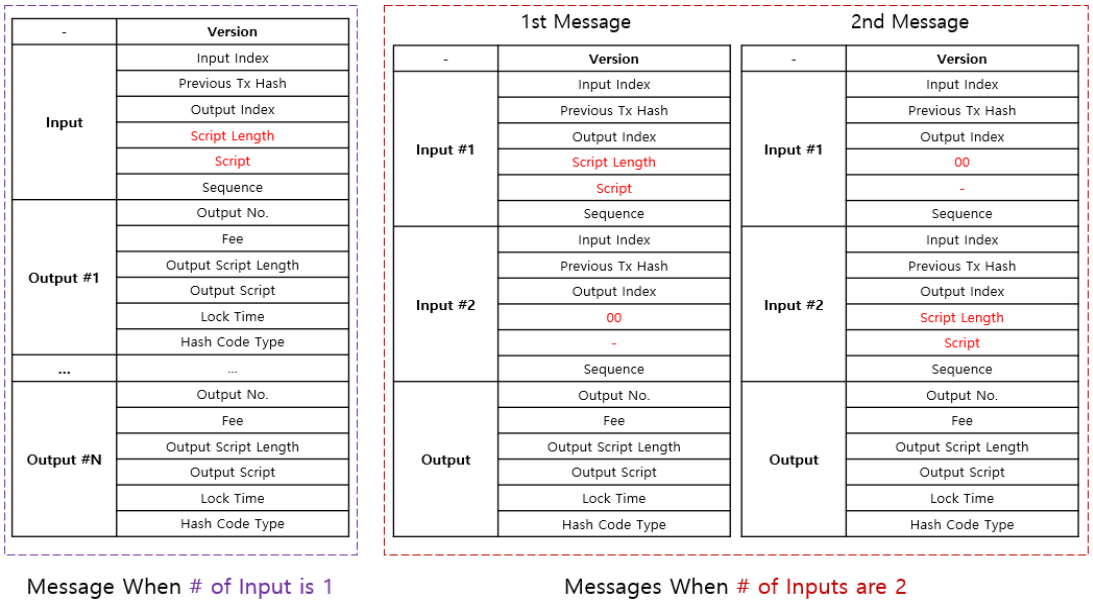


Fig. 9. Structure of Bitcoin Transaction Message

The process of extracting messages from the block data can be divided into two types according to the number of inputs included in the transaction. The left side of **Fig. 9** shows the case of a single input. An area in the input script contains the signature value, but it must be filled with a different value because the message is from before the signature has been performed. Therefore, the area is filled as shown below. All of the outputs are combined, as shown in the figure, and the message is used.

Script Length: 0x19
Script: 0x76A914 || RIPEMD160[SHA256(sender's pubkey)] || 0x88AC

The right side of **Fig. 9** shows a case with two inputs. Therefore, two messages must exist, and the messages are organized as shown below. The '0x00' in the length means a value of size 1 byte, and '–' means no value.

- i-st Message

Input #i's Script Length: 0x19
Input #i's Script: 0x76A914 || RIPEMD160[SHA256(sender's pubkey)] || 0x88AC
Others' Script Length: 0x00
Others' Script: -

Transactions that have two or more inputs contain a message that is similar to that above, and all outputs are added after the input, as in the case of a single input.

5.1.3 Extracting Signatures from Block Data

To collect signature values from the block data, the bitcoin Wiki and the bitcoin API were referenced to write the data collecting source code. The source code consists of "main," "divide_blocks," "divide_transactions," and "extract_sign" functions. Shown below is a summary of the source code's structure and roles.

The "main" function, shown in **Table 8**, extracts the signature values of all "blkxxxxx.dat" files. Namely, each file is read and transferred to the extraction function, and this process is repeated. By executing this function, all signature values and public keys are extracted.

Table 8. Pseudo Code of "main"

Contents	Details
function name	main
Input	All 'blk*xxxxx*.dat' files
Output	A set of all signatures and public key
Pseudo Code	for i is 0 to 974 file = open(i-th file) blocks = divide_blocks(file) signs = divide_transactions(blocks) append signs to all_signs return all_signs

The "divide_blocks" function in **Table 9** divides all the blocks stored in a single file. In this process, a magic number is used to find the beginning of a block, and the block's size is used to find the end of the block. By executing this function, all the divided blocks in the file are returned.

Table 9. Pseudo Code of "divide_blocks"

Contents	Details
function name	divide_blocks
Input	'blk*xxxxx*.dat' file
Output	Array of blocks
Pseudo Code	magic_num = file.read(4bytes) while len(magic_num) is not 0 block_size = file.read(4bytes) block = file.read(block_size-bytes) append block to blkArray magic_num = file.read(4bytes) return blkArray

The "divide_transactions" function in **Table 10** takes a block array and extracts the scripSig from the transaction input included in each block. Data are not extracted from the output, but the size of the output must be calculated to divide the transactions. By passing the scriptSig through the "extract_sign" function, the signature value (r, s) and public key are divided. By executing this function, a set of the signature value (r, s) and public key is returned.

Table 10. Pseudo Code of "divide_transactions"

Contents	Details
function name	divide_transactions
Input	Array of blocks
Output	Sets of Signatures

Pseudo Code	for block in blkArray{
	numOfTx = read # of Tx
	for i is 0 to numOfTx{
	if(i is equal to 0)
	continue (*first Tx is passed.*)
	numOfInput = read # of Inputs
	for j is 0 to numOfInputs{
	scriptSig_len = read length of scriptSig
	scriptSig = read data by scriptSig_len
	extract_sign(signs, scriptSig)
	}
	numOfOutput = read # of Outputs
	for j is 0 to numOfOutputs{
	(*don't extract any data.*)
	}
	}
	}
	
	return signs

The "extract_sign" function in **Table 11** extracts the signature value and public key from the scriptSig. By executing this function, a set that adds the signature value (r, s) and public key to the sign set that was received as input is returned.

Table 11. Pseudo Code of "extract_sign"

Contents	Details
function name	extract_sign
Input	scriptSig
Output	Sets of Signatures
Pseudo Code	r_len = read length of r from scriptSig r_value = read Signature r by r_len s_len = read length of s from scriptSig s_value = read Signature s by s_len pubkey_len = read length of pubkey from scriptSig pubkey = read public key by pubkey_len append a set of [rvalue, svalue, pubkey] to signs return signs

5.2 Private Key Recovery with Proposed Scenarios

The proposed scenario is applied to the data from bitcoin's first block to its 481,614th block, which was included in the chain on August 22, 2017, to demonstrate the user private key recovery. Among the 515,429,782 r values of signatures collected from the block data, 1,231 r values of signatures were used two or more times, and **Table 12** shows some of them.

Table 12. Five Most Frequent R Values in Block Data

R Values	Occurrences
0x000000000000000000000003b78ce563f89a0ed9414f5aa28ad0d96d6795f9c63	2,520,988
0x00006fcf15e8d272d1a995af6fcc9d6c0c2f4c0b6b0525142e8af866dd8dad4b	7,873
0x79be667ef9dcbbac55a06295ce870b07029bfcdb2dce28d959f2815b16f81798	242
0x1206589b08a84cb090431daa4f8d18934a20c8fa52ad534c5ba0abb3232be1d9	234
0x06cce13d7911baa7856dec8c6358aaa1fb119b5a77d0e4d75d5a61acae05fcfb	78

Table 13 shows a list and the description of parameters used in the bitcoin user private key recovery presented in the following sections.

Table 13. Description of Parameters for Private Key Recovery

Parameters	Description	Parameters	Description
n	Modulo	r	r value of Signature
G	Generator	s	s value of Signature
M	Message	Q_x	x-coordinate of Public Key
e	Hashed Message	Q_y	y-coordinate of Public Key
k	Nonce	d	Private Key

5.2.1 Recovery of Private Key Generating Duplicated Signatures

Regarding the use of r values of signatures that have been used two or more times to recover private keys and signature values, the cases were divided according to the four attack types shown in **Table 6**, and private key recovery formulas were calculated according to the type. Data analysis demonstrated that duplicated signatures belonged to the first type. Therefore, the actual private key recovery is shown via the private key recovery formula from Section 2.7.1. **Tables 14** and **15** show two signatures of users who have generated duplicated signatures; as shown, the public keys and r values of signatures are the same.

Table 14. 1st Signature Values Generating Duplicated Signatures

Parameters	Values(Hex)
Q_x	DBD0C61532279CF72981C3584FC32216E0127699635C2789F549E0730C059B81
Q_y	AE133016A69C21E23F1859A95F06D52B7BF149A8F2FE4E8535C8A829B449C5FF
r	D47CE4C025C35EC440BC81D99834A624875161A26BF56EF7FDC0F5D52F843AD1
s_1	965C9814F4DE166D003E2D37C53122FDF92CD72E410D01A71C3833FD78626BF9
e_1	6B339AD927CAC0FF077428856E77E2E68238B7BABC628294DD10901AD4CCB678

Table 15. 2nd Signature Values Generating Duplicated Signatures

Parameters	Values(Hex)
Q_x	DBD0C61532279CF72981C3584FC32216E0127699635C2789F549E0730C059B81
Q_y	AE133016A69C21E23F1859A95F06D52B7BF149A8F2FE4E8535C8A829B449C5FF
r	D47CE4C025C35EC440BC81D99834A624875161A26BF56EF7FDC0F5D52F843AD1
s_2	E0BF353E30DF652F819893958FBE88C98913A773A77E1FDE7243C1FE9A2097F0
e_2	1244A3EF7880374FF2D6B53C21566336E372427C2D333315BD8F635ED72C61B

Table 16 shows the recovery of private keys and nonces of users who have generated duplicated signatures via the data in **Tables 14** and **15**. To confirm that the recovered private key (d') matches the user's actual private key, a new public key (Q_x', Q_y') was generated.

Because the newly generated value matches that of the public key (Q_x,Q_y) in **Table 14**, the recovered private key is the same as the user's actual private key. To confirm that the recovered nonce (k') is the nonce used in the actual signature, a new signature value (r') was generated. Because the newly generated value matches the r signature value in **Table 14**, the recovered nonce is the same as the nonce used in the actual signature.

Table 16. Recovery of Private Key Generating Duplicated Signatures

Parameters	Values(Hex)
n	FFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFEBAAEDCE6AF48A03BBFD25E8CD0364141
$s_1 - s_2$	B59D62D6C3FEB13D7EA599A235729A332AC80CA148D7820469C6D08BAE78154A
$e_1 - e_2$	6A0F509A3042BD8A0846BD31AC627CB314019392F98F4F63813799E4E759F05D
k'	7A1A7E52797FC8CAAA435D2A4DACE39158504BF204FBE19F14DBB427FAEE50AE
r'	D47CE4C025C35EC440BC81D99834A624875161A26BF56EF7FDC0F5D52F843AD1
d'	**C477F9F65C22CCE20657FAA5B2D1D8122336F851A508A1ED04E479C34985BF96**
Q_x'	DBD0C61532279CF72981C3584FC32216E0127699635C2789F549E0730C059B81
Q_y'	AE133016A69C21E23F1859A95F06D52B7BF149A8F2FE4E8535C8A829B449C5FF

In the analyzed block data, 1,194 private keys generated duplicated signatures, and these private keys can be recovered from the block data by attackers.

5.2.2 Recovery of Private Key Using Duplicated Signatures

The analysis of the bitcoin block data indicated cases where duplicated signatures were used in addition to cases where duplicated signatures were generated. **Table 17** shows the user transaction data for which the signature value was generated only once. Although the user generated the signature value only once, the r value of signature is the same as those of the duplicated signatures shown in **Table 14**, **15**.

Table 17. Parameters of Signatures Using Duplicated Signatures

Parameters	Values(Hex)
Q_x	37905189FD5E9039948CC4D02406823FF9517ED12EC7D4EEFF4C4D6009CE5FAB
Q_y	1F172295C2248FF6E092FB06C6DD47A24FB682CF7749FD370E6F94E5660B865D
r	D47CE4C025C35EC440BC81D99834A624875161A26BF56EF7FDC0F5D52F843AD1
s	2A6F6DD517F761E5EF5F3597D5E5A250B2B16C4E44EB24B1420CF8AA7CA1AC58
e	BEB01CFA8C4C33B34D67B8A69B7A2B261C343192E41008FFD2DE444D925A8ECF

The nonce that corresponds to the r value of signature in **Table 17** can be found, and this is presented in **Table 18**. Therefore, an attacker can use the nonce to find the user's private key.

Table 18. Example of Signature and Corresponding Nonce

Parameters	Values(Hex)
r	D47CE4C025C35EC440BC81D99834A624875161A26BF56EF7FDC0F5D52F843AD1
k	7A1A7E52797FC8CAAA435D2A4DACE39158504BF204FBE19F14DBB427FAEE50AE
-k mod n	85E581AD8680373555BCA2D5B2531C6D625E90F4AA4CBE9CAAF6AA64D547F093

Table 19 shows the recovery of a user's private key using a nonce that corresponds to the duplicated signature shown in **Table 18**. 'k' and '-k' are the nonces that can generate the same value as the r value of signature in **Table 18**, and the private keys are calculated for both cases.

The public keys (Q^1, Q^2) calculated from the two private keys (d_1,d_2) are compared to the user's actual public key (Q) in **Table 17** to determine the private key. A comparison between the two public keys (Q^1, Q^2) and the user's actual public key demonstrates that d_1 is the user's actual private key.

Table 19. Recovery of Private Key Using Duplicated Signatures

Parameters	Values(Hex)
n	FFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFEBAAEDCE6AF48A03BBFD25E8CD0364141
d_1	**B91989F4E578471194547BABF3BC80A5FB58C4F753A62C48ED0D01954C110F19**
$Q_x{}^1$	37905189FD5E9039948CC4D02406823FF9517ED12EC7D4EEFF4C4D6009CE5FAB
$Q_y{}^1$	1F172295C2248FF6E092FB06C6DD47A24FB682CF7749FD370E6F94E5660B865D
d_2	C2B634205A3E8C034943D2FFB82A92676C1CE6A12FCC3F3E82E7C8FB4036ACC4
$Q_x{}^2$	745C820C1ED88CB4DD47A6C9FFF66EB97CA3808B24D36079A13932B46B0B16AB
$Q_y{}^2$	50AE177E60DCD90FE04DB2802045EE597D777A4290D55949C8DE0C6927625ABC

An analysis of the block data indicated that 621 private keys used duplicated signatures. These private keys can be recovered from the block data by attackers.

Table 20 shows the number of private keys that generated or used duplicated signatures, and the number of private keys that could be recovered from the analyzed block data; 239,039,487 private keys were used in the block data that was analyzed in this study; 1,815 private keys could be recovered owing to the generation and use of duplicated signatures.

Table 20. The # of Recoverable Private Key

Scenarios of Private Key Recovery	# of Private Key
Generating Duplicated Signatures	1,194
Using Duplicated Signatures	621
Sum	1,815

5.2.3 Possible Recovery of Private Key with Duplicated Signatures

All duplicated signatures in bitcoin block data were generated by reusing same nonce. However, someone may use complementary nonce to create digital signature and it can result duplicated signatures. **Table 21** shows an artificial transaction data for which the signature value was generated with complementary nonce shown in **Table 18**.

Table 21. Parameters of Signatures with Complementary Nonce

Parameters	Values(Hex)
Q_x	3F7C27E3833F20D6C1E7B5E7B7271223E585780267DA2ACAF5CB27AAE9C9C2A1
Q_y	752398912D921A619BDFFF5E8FF690EE32A1D28393134581D0E413A50047C79E
r	D47CE4C025C35EC440BC81D99834A624875161A26BF56EF7FDC0F5D52F843AD1
s	D23B2358837605C0C38399DE0CD83F585796EE06D1CE50DE1960D598EEBC387A
e	171BBEF1BEB7EF68B4D354C0C4022F9903F952DC172CC3754D84490C754359D7

Table 22 shows the recovery of a user's private key with nonce. 'k' and '-k' are the nonces that can generate the same value as the r value of signature in **Table 18**, and the private keys are calculated for both cases. The public keys ($\boldsymbol{Q^1}$, $\boldsymbol{Q^2}$) calculated from the two private keys

($\boldsymbol{d_1}$,$\boldsymbol{d_2}$) are compared to the user's actual public key (Q) in **Table 21** to determine the private key. A comparison between the two public keys ($\boldsymbol{Q^1}$, $\boldsymbol{Q^2}$) and the user's actual public key demonstrates that $\boldsymbol{d_2}$ is the user's actual private key.

Table 22. Recovery of Private Key with Duplicated Signatures

Parameters	Values(Hex)
n	FFFFFFFFFFFFFFFFFFFFFFFFFFFFFFFEBAAEDCE6AF48A03BBFD25E8CD0364141
d_1	2A83C4FBFA1C3C14B772A288C63359792E3D91626B6818DC7F2E57905417BF98
${Q_x}^1$	AFF8C6064848B09ADF73B7F1707EA4D6C393869DFE1070F8F06E0AF028177B4E
${Q_y}^1$	8674D5D233CB08B8AEB83737A730A5C0C3462AF016A5BC6E1784671D079E8A61
d_2	**B91989F4E578471194547BABF3BC80A5FB58C4F753A62C48ED0D01954C110F19**
${Q_x}^2$	37905189FD5E9039948CC4D02406823FF9517ED12EC7D4EEFF4C4D6009CE5FAB
${Q_y}^2$	1F172295C2248FF6E092FB06C6DD47A24FB682CF7749FD370E6F94E5660B865D

6. Conclusion

A user private key recovery scenario that could occur because of the characteristics of blockchain systems where all transaction information is shared was presented herein. It demonstrated the possibility of recovering private keys in an actual bitcoin environment. The analysis of the actual bitcoin environment showed that 1,815 among 239,039,487 private keys could be recovered. As bitcoin usage increases, a larger number of private keys could be recovered from the overall block data. Because the blockchain grants property ownership rights to the private key owner, private key leaks can result in property theft. Therefore, to provide secure and efficient blockchain systems to the financial sector, continued research on the prevention of private key recovery threats is necessitated.

Acknowledgment

This research was supported by the MSIT(Ministry of Science and ICT), Korea, under the ITRC(Information Technology Research Center) support program (IITP-2019-2016-0-00304) supervised by the IITP(Institute for Information & communications Technology Planning & Evaluation)

References

[1] Dong Sup Kim, "Current Status and Implications of Distributed Ledger Technology and Digital Currency," *Payment Systems Research Team, The Bank of Korea*, Jan. 2016. Article (CrossRefLink).

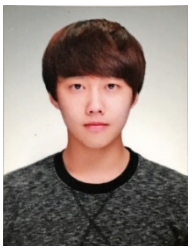
[2] Korea Internet & Security Agency, "A Study on the Applications and Use Cases of Blockchain Technology," *KISA-WP-2016-0040*, Nov. 2016. Article (CrossRefLink).

[3] IBK Economic Research Institute, "A Study on the Understanding of Blockchain and the Utilization of Financial Industry," Jul. 2017. Article (CrossRefLink).

[4] LG CNS, "Cryptocurrency and Blockchain Hacking Cases," *IT Solutions/Security*, Nov. 2018. Article (CrossRefLink).

[5] Nicolas T. Courtios, Pinar Emirdag, Filippo Valsorda, "Private Key Recovery Attacks: On Extreme Fragility of Popular Bitcoin Key Management, Wallet and Cold Storage Solutions in Presence of Poor RNG Events," *Cryptology ePrint Archive*, Oct. 2014. Article (CrossRefLink).

[6] Michael Brengel, Christian Rossow, "Identifying Key Leakage of Bitcoin Users," *RAID 2018: Research in Attacks, Intrusions, and Defenses*, pp. 623-643, Sep. 2018. Article (CrossRefLink).
[7] TTA, "Security Requirements for Digital Hospital," *TTAK.KO-12.0305*, pp.6-7, Nov. 2016. Article (CrossRefLink).
[8] Information Technology Laboratory National Institute of Standards and Technology, "Digital Signature Standard (DSS)," *FIPS PUB 168-4*, pp.26-30, Jul. 2013. Article (CrossRefLink).
[9] American Bankers Association Standards Department, "Public Key Cryptography for The Financial Services Industry: The Elliptic Curve Digital Signature Algorithm (ECDSA)," *American National Standard X9.62-1998*, Sep. 1998. Article (CrossRefLink).
[10] William Stallings, *Cryptography & Network Security 5th ed*, Pearson Education, Mar. 2010.
[11] Korea Internet & Security Agency, "Hash Function". Article (CrossRefLink).
[12] Ilias Giechaskiel, Cas Cremers, Kasper B. Rasmussen, "On Bitcoin Security in the Presence of Broken Cryptographic Primitives," *Lecture Notes in Computer Science*, vol. 9879, pp. 201-222, 2016. Article (CrossRefLink).
[13] Quynh Dang, "Recommendation for Applications Using Approved Hash Algorithms," *NIST Special Publication 800-107, NIST Computer Security Division Information Technology Laboratory*, Aug. 2012. Article (CrossRefLink).
[14] Satoshi Nakamoto, "Bitcoin: A Peer-to-Peer Electronic Cash System," pp. 1-9, 2009. Article (CrossRefLink).
[15] Blockchain.com, "Block at depth 579952 in the Bitcoin blockchain," Jun. 2019. Article (CrossRefLink).
[16] Bitcoin Core. Article (CrossRefLink).
[17] CCN, "Bitcoin for Mere Mortals: Blocks," Nov. 2014. Article (CrossRefLink).



Ju-Seong Ko received Korea B.S. degree in Department of Cyber Security from Ajou University. He is currently pursuing the M.S. degree in Department of Computer Engineering with Ajou University, Korea. His research interests include Cryptographic protocols and Blockchain & Fintech analysis.

Jin Kwak is a professor at Dept. Of Cyber Security in Ajou University, Korea. He received the Ph.D. degree from SKKU, Korea. His research interests include Cryptographic protocols, Applied security mechanisms for Cloud and Big Data system and so on.