# Algorithm for Increasing the Stability Level of Cryptosystems

Roman Kvyetnyy, Yaroslav Ivanchuk, Andrii Yarovyi and Yurii Horobets

*Vinnytsia National Technical University, Khmelnytsky highway 95, Vinnytsia, 21021, Ukraine*

**Abstract**

The article considers the types of DSA and ECDSA cryptosystems affected by side channel attacks. A method for identifying the type of impact based on lattice attacks has been developed, which showed that if there are common bits in ephemeral keys, the sender's private key can be restored in polynomial time, if there are the required number of messages, depending on the number of common bits. As countermeasure this attack an ephemeral key generation algorithm was developed, using a deterministic signature, to increase the crypto-resistance of systems to lattice attacks, and a modular inverse algorithm was improved to perform the operation at a constant time to prevent sidechannel attack on modular inverse operation, which using for generation signature.

**Keywords** [1]

lattice, shared bits, modular inverse, constant-time, deterministic signature, cryptosystems

## 1. Introduction

Cryptosystem is the implementation of cryptographic methods and their infrastructure for the provision of information security services [1]. Cryptosystems convert source data into unreadable form using encryption and decryption keys, which in turn allows for confidentiality of information. At the heart of any encryption system is the availability of information on the exchange of plaintext data only from the sender and recipient. Depending on the method of encrypting and decrypting information, cryptosystems are divided into two types [1]:

1. A symmetric encryption system is based on the use of the same type of key, the information on the use of which is agreed by the parties before the data session. For example, cryptosystems of this type: DES, Triple-DES, BLOWFISH, IDEA, AES [2]. The main disadvantage of this encryption system is the complexity of key management in large networks.

2. The asymmetric encryption system uses different types of keys, which are interconnected by a certain mathematical dependence. This approach allows you to get the original plaintext by using the inverse functions of mathematical operations. The main representatives of these cryptosystems encryption are DSA, DSS, RSA, ECDSA, El-Gamal [3 – 5]. The main disadvantages of these encryption systems are the availability of information about the public key of the sender and recipient when creating an encrypted message; the level of security of encryption systems is proportional to the level of complexity of operations based on various mathematical problems (integer factorization complexity, discrete logarithm problem [6]).

For some encryption systems, an ephemeral key (nonce) is used, which is needed to add more complexity to the encryption algorithm, the main requirement for it, the ephemeral key must be used only once. For DSA, ElGamal, ECDSA encryption systems - when creating encrypted messages, an ephemeral key is additionally used, which in turn leads to attacks on side channels, FLUSH+RELOAD ephemeral key data, attacks on the pseudo-random number generator [7], using oscillography for timings attack [8], using masked digital signal [9]. All of these attack types result in the substitution or interception of individual bits of ephemeral key data, giving third parties access to encrypted messages, which in turn compromises the sender's secret key. A particularly dangerous cryptoattack types are side channel attack, which allow, by analyzing the time of generation of the ephemeral key, to find the most

*Information Technology and Implementation (IT&I-2021), December 01–03, 2021, Kyiv, Ukraine*
EMAIL: rkvetny@sprava.net (A. 1); ivanchuk@ukr.net (A. 2); a.yarovyy@gmail.com (A. 3); yuriy.sparrow@gmail.com (A. 4)
ORCID: 0000-0002-9192-9258 (A. 1); 0000-0002-4775-6505 (A. 2); 0000-0002-6668-2425 (A. 3); 0000-0002-0483-1042 (A. 4)


or least significant bits of the ephemeral key. Then using special methods the sender's private key is searched. Articles [11, 12] show a crypto attack on a DSA-type encryption system, in which the sender's private key can be found using methods for solving number theory problems, namely the problem of hidden numbers, to solve which uses the method of short vectors or the method of close vectors. by collecting from the n-number of encrypted messages less and / or more significant bits of the ephemeral key, using a side channel attack, but this attack is not possible if the numerical value of these bits is unknown.Scientific research [12] investigates the cryptoattack type on the private key of the sender using the method of fast Fourier transformation, which allows you to effectively search for a private access key among a large number of encrypted messages. The high efficiency of this crytoattack leaves open the question of developing a reliable method of protecting systems from such cryptoattack.

The article [14, 15] reveals the issue of protection against side channel attack, using the method of constant-time addition of numbers when creating encrypted messages. Using this method requires significant system resources, which generally slows down the process of generating encrypted messages. Therefore, it is important to increase the level of resilience of cryptosystems to various sidechannel attack, which will increase the overall level of security of data encryption when exchanging information. The aim of the work is to increase the degree of protection of the process of information exchange in system networks on the basis of the developed effective data encryption algorithm.

To achieve this goal it is necessary to determine the vulnerability type of cryptosystems to sidechannel attack, as well as to develop an effective method protection cryptosystems from sidechannel attack to prevent unauthorized access to encrypted information.

## 2. A summary of the main material

To identify a potentially dangerous type attack on private key, in based on the DSA encryption algorithm (ECDSA), it is necessary to analyze the mathematical description of this algorithms.

To use the DSA algorithm, the signer selects two prime numbers *p* and *q*, and generator *g*, where *p* – prime number of size between 512 and 1024 bits, with increments of 64 bits; *q* – prime number, recommended by NIST standards [14] with size at least 160 bits, and corresponds to equality $2^{N-1} < q < 2^N$, where $N \in \{160, 224, 256\}$, and $q \mid p-1$; *g* – generator of order *q* subgroup *G* of finite field $\mathrm{F}_p^{\times}$ abelian group. Furthermore, he selects randomly $a \in \{1, ..., q-1\}$ and computes $A = g^a \bmod p$. The public key of the signer is $(p, q, g, A)$ and his private key *a*. He also publishes a hash function $h: \{0,1\}^* \to \{0, ..., q-1\}$, for mapping message into range $\{1, \ldots, q-1\}$.

To sign a message *m*, signer selects randomly $k \in \{1, ..., q-1\}$, which is the ephemeral key (or nonce), and computes:

$$r = (g^k \bmod p) \bmod q; s = k^{-1}(h(m) + ar) \bmod q. \quad (1)$$

The signature of *m* is the pair (*r*, *s*). The signature (1) is valid if and only if:

$$r = ((g^{s^{-1}h(m) \bmod q} A^{s^{-1}r \bmod q}) \bmod p) \bmod q. \quad (2)$$

For the ECDSA the signer chooses an elliptic curve *E* over finite field $\mathrm{F}_p$, a point $P \in E(\mathrm{F}_p)$ (respectively to DSA is generator) with prime order *q* with size at least 160 bits. Appropriately, to FIPS 186-3 binary length of prime number *p* must be in set {160, 224, 256, 512}. Furthermore, for randomly chosen $a \in \{1, ..., q-1\}$ computes $Q = aP$. The public key of signer is $(E, p, q, P, Q)$ and his private key *a*. Also, signer publishes hash function $h: \{0,1\}^* \to \{0, ..., q-1\}$. To sign a message *m*, signer chooses random number $k \in \{1, ..., q-1\}$, computes $kP = (x, y)$ (where *x* and *y* is integers in range $\{0, \ldots, p-1\}$. Further, computes the signature of message *m*, is pair of integers (*r*, *s*):

$$r = x \bmod q; \quad s = k^{-1}(h(m) + ar) \bmod q. \quad (3)$$

For verification the signatures, receiver computes:

$$u_1 = s^{-1}h(m) \bmod q, \quad u_2 = s^{-1}r \bmod q, \quad u_1P + u_2Q = (x_0, y_0). \quad (4)$$

The signature (3) is valid if and only if, when $r = x_0 \bmod p$. The security of the two systems s relied on the assumption that the only way to forge the signature is to recover either the secret key *a*, or the ephemeral key *k*. Thus, the parameters of these systems were chosen in such a way that the computation of discrete logarithms is computationally infeasible.

To perform a potentially dangerous crypto attack by an attacker, which will reveal the signer private key, an attacker can block some bits of register or memory that contain the value of the ephemeral key k, while the value of the blocked bits is unknown. Assume that an attacker was able to collect *n*-number encrypted messages $m_i$ (*i*=1, …, *n*) with the associated signatures ($r_i$, $s_i$), where all the corresponding ephemeral keys $k_i$ shared total $\delta$ bits between more significant (MSB) and less significant (LSB) bits independent of *i*. In this way, they will meet the following dependency for all *i*=1, ..., *n*:

$$k_i = k + 2^t \tilde{k}_i + 2^{t'} k'. \tag{5}$$

where $0 \le k < 2^t, 0 \le k' < 2^{N-t'}, \delta = \mathrm{N} - t' + t, 0 \le k_i < 2^{N-\delta}$, with *k* and *k´* common for all $k_i$.

Figure 1 shows a diagram of the placement of bits in the ephemeral key.

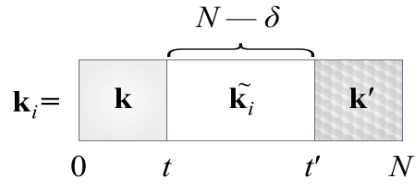


**Figure 1:** Scheme of the ephemeral key
*k* – numerical value of the total more significant bits (MSB), *k′* – numerical value of common less significant bits (LSB), $\tilde{k}_i$ - numeric value of the current bits, *t* – number of common more significant bits, *t′* – number of common less significant bits

It should be noted that the values of the variables $k_i, k, \tilde{k}_i, k'$ are unknown. In the *n* of equations (4), which define the signature:

$$\begin{cases} m_1 + ar_1 - s_1 k_1 \equiv 0 \pmod q; \\ m_2 + ar_2 - s_2 k_2 \equiv 0 \pmod q; \\ \dots \\ m_n + ar_n - s_n k_n \equiv 0 \pmod q, \end{cases} \tag{6}$$

where $m_i$ – message, *a* – signer private key, $r_i$ – numerical value of the first part of the signature, $s_i$ – numerical value of the second part of the signature, $k_i$ – ephemeral key, *i*-message index.

If in the system of equations (6) replace the parameters $m_i$, $r_i$, $s_i$ with the value (5) and eliminate common variables *k* and *k′*, then we obtain:

$$\begin{cases} (s_1^{-1} m_1 - s_2^{-1} m_2) + a(s_1^{-1} r_1 - s_2^{-1} r_2) - 2^t (\tilde{k}_1 - \tilde{k}_2) \equiv 0 (\mathrm{mod}\, q); \\ (s_1^{-1} m_1 - s_3^{-1} m_3) + a(s_1^{-1} r_1 - s_3^{-1} r_3) - 2^t (\tilde{k}_1 - \tilde{k}_3) \equiv 0 (\mathrm{mod}\, q); \\ \dots \\ (s_1^{-1} m_1 - s_n^{-1} m_n) + a(s_1^{-1} r_1 - s_n^{-1} r_n) - 2^t (\tilde{k}_1 - \tilde{k}_n) \equiv 0 (\mathrm{mod}\, q). \end{cases} \tag{7}$$

Let $\alpha_i, \beta_i, \kappa_i \in Z$ be such that:

$$\begin{cases} \alpha_i := 2^{-t} (s_1^{-1} m_1 - s_i^{-1} m_i) \bmod q; \\ \beta_i := 2^{-t} (s_1^{-1} r_1 - s_i^{-1} r_i) \bmod q; \\ \kappa_i := \tilde{k}_1 - \tilde{k}_i. \end{cases} \tag{8}$$

Then, system of equalities (7) becomes:

$$\begin{cases} \alpha_2 + a\beta_2 - k_2 \equiv 0 \,(\mathrm{mod}\, q); \\ \alpha_3 + a\beta_3 - \kappa_3 \equiv 0 \,(\mathrm{mod}\, q); \\ \alpha_n + a\beta_n - \kappa_n \equiv 0 \,(\mathrm{mod}\, q), \end{cases} \quad (9)$$

where $a$, $\kappa_i$ and $\alpha_i$, $\beta_i$ – are known value and unknown value.

The set of solutions:

$$L = \{(x_0, x_1, ..., x_n) \in \mathbf{Z}^{n+1} \mid x_0\alpha_i + x_1\beta_i - x_i \equiv 0 \,(\mathrm{mod}\, q)\}, \quad (10)$$

forms an $(n+1)$–dimension lattice spanned by the row vectors of the following basis matrix:

$$M = \begin{pmatrix} 1 & 0 & \alpha_2 & \dots & \alpha_n \\ 0 & 1 & \beta_2 & \dots & \beta_n \\ 0 & 0 & q & \dots & 0 \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & 0 & 0 & q \end{pmatrix}. \quad (11)$$

To find the short vector in the matrix (11), LLL or BKZ algorithms [17, 18] can be used. If the system of equations (7) has a solution ((determining the value of a short vector $\overline{\upsilon}_0$ with elements $v_0 = (1, a, \kappa_2, \kappa_3, ..., \kappa_n)$, which will be identical to the element of the system of solutions (10), then the value of the signer's private key $a$ will be obtained. However, the norm $\overline{\upsilon}_0$ is lower bounded by secret key $a$, which can be an integer of approximately $N$ bits. Then the next element of the vector $\overline{\upsilon}_0$ is much larger than the next ones, which are $N$-$\delta$ integers. Vector $\overline{\upsilon}_0$ there is no ground to be a short vector $L$, when signer private key $a$ is upper-bounded. Therefore, assuming that private key $a$, smaller than $2^{N-\delta}$, then, it is necessary to adapt the method described above to find the signer's private key $a$ length of $N$ bits, to solve this problem can be used one of methods described below:

If the value $\delta$ is small (no more then 8) , then the method goes to the previous, with searching for most significant bits of $a$, then we have $a = \tilde{a} + 2^{N-\delta}a'$, where $\tilde{a} < 2^{N-\delta}$ and $a' < 2^{\delta}$. Then, in (8) $\alpha_i$ changing to $\alpha_i = 2^{-t}(s_1^{-1}m_1 - s_i^{-1}m_i) + a'2^{-t}(s_1^{-1}r_1 - s_i^{-1}r_i)$.

Removing the second column from the matrix (11), then get the matrix:

$$M' = \begin{pmatrix} 1 & \alpha_2 & \dots & \alpha_n \\ 0 & \beta_2 & \dots & \beta_n \\ 0 & q & \dots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & 0 & q \end{pmatrix}, \quad n > 2. \quad (12)$$

But this method requires more messages and larger matrix that negatively smashes the speed of work. Using a weighted Euclidean product of vectors to get a vector $v_0$ in the basis of reduced algorithms using knowledge about the size of the target vector $v_0$. For example, we take a weighted Euclidian product while performing reduction of basis:

$$\left\langle (x_0, ..., x_n), y_0, ..., y_n) \right\rangle := \sum_{i=0}^{n} x_i y_i 2^{2(N - [\log_2(v_{0,i})])}, \quad (13)$$

The use of this method reduces the number of common bits $\delta$. Weights can be used without changing the norm that in reality corresponds to the multiplication of all columns to the corresponding weight. The previous method is used when the ephemeral keys are shared by multiple bits blocks. Now assume that we have $n$ messages $m_i$ ($i$ = 1, …, $n$) with the associated signatures ($r_i$, $s_i$), where ephemeral keys $k_i$ share a total $\delta$ bits distributed between $l$ block of bits. Denote by $\delta_i$ the number of bits $i$-th block at

position $p_i$. Let $t = (t_1, \ldots, t_l)$ be a $l$-set of integers, then set $2^t = (2^{t_1}, ..., 2^{t_l})$. Then ephemeral key $k_i$ has form:

$$k_i = 2^p \cdot b + 2^t \cdot k_i, \tag{14}$$

where $b=(b_1, \ldots, b_i)$ is the vector of shared bits blocks, $p=(p_1, \ldots, p_l)$ is vector of position a $l$ blocks, $k_i=(k_{i,0}, \ldots, k_{i,l})$ the vector of no shared bits blocks at positions $t = (t_0, \ldots, t_l)$.
On figure 2, depicts the scheme of location of bits blocks.

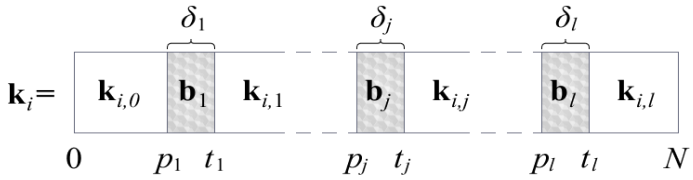


**Figure 2:** Scheme of bits location

After stating that $t := 0$ and $p_{l+1} := N$, it follows that for all $i$ =1, …, $n$ i and for all $j$ = 1, …, $n$:

$$t_j = p_j + \delta_j, \quad \delta = \sum_j \delta_j, \quad 0 \le b_j < 2^{\delta_j}, \quad 0 \le k_{i,j} < 2^{(p_{j+1} - t_j)}. \tag{15}$$

Values of $k_i, k_{i,j}, b_j$ are unknown. In signatures (7), substitute $k_i$ by (15) and eliminate the common variable $b$:

$$\begin{cases} (s_1^{-1}m_1 - s_2^{-1}m_2) + a(s_1^{-1}r_1 - s_2^{-1}r_2) - \sum_{j=0}^{l} 2^{t_j}(k_{1,j} - k_{2,j}) \equiv 0 \pmod q; \\ (s_1^{-1}m_1 - s_3^{-1}m_3) + a(s_1^{-1}r_1 - s_3^{-1}r_3) - \sum_{j=0}^{l} 2^{t_j}(k_{1,j} - k_{3,j}) \equiv 0 \pmod q; \\ \ldots \\ (s_1^{-1}m_1 - s_n^{-1}m_n) + a(s_1^{-1}r_1 - s_n^{-1}r_n) - \sum_{j=0}^{l} 2^{t_j}(k_{1,j} - k_{n,j}) \equiv 0 \pmod q. \end{cases} \tag{16}$$

Let $\alpha_i, \beta_i \in \mathbf{Z}$ та $\kappa_i, t \in \mathbf{Z}^l$ are those:

$$\begin{cases} \alpha_i := s_1^{-1}m_1 - s_i^{-1}m_i \bmod q; \\ \beta_i := s_1^{-1}r_1 - s_i^{-1}r_i \bmod q; \\ \kappa_i := (k_{1,1}, ...., k_{1,l}) - (k_{i,1}, \ldots, k_{i,l}); \\ t := (t_1, ...., t_l), \end{cases} \tag{17}$$

then (16) becomes:

$$\begin{cases} \alpha_2 + a\beta_2 - 2^t\kappa_2 \equiv k_{1,0} - k_{2,0} \pmod q; \\ \alpha_3 + a\beta_3 - 2^t\ \kappa_3 \equiv k_{1,0} - k_{3,0} \pmod q; \\ \ldots\ldots \\ \alpha_n + a\beta_n - -2^t\kappa_n \equiv k_{1,0} - k_{n,0} \pmod q, \end{cases} \tag{18}$$

where $a$ and $\kappa_i$ – unknown $\alpha_i$ and $\beta_i$ known. Embedding this equation in a lattice $L$, spanned by row vectors of the following basis matrix.

$$M = \left(\begin{array}{c|ccc} I_{l(n-1)+2} & \alpha_2 & \cdots & \alpha_n \\ & \beta_2 & \cdots & \beta_n \\ \hline I_{l(n-1)+2} & \multicolumn{3}{c}{2^{t_1} I_{(n-1)}} \\ \hline I_{l(n-1)+2} & \multicolumn{3}{c}{\vdots} \\ \hline I_{l(n-1)+2} & \multicolumn{3}{c}{2^{t_1} I_{(n-1)}} \\ \hline 0 & \multicolumn{3}{c}{qI_{(n-1)}} \end{array}\right). \tag{19}$$

Then $\alpha_i, \beta_i, \kappa_i \in \mathbf{Z}$ be this form:

$$\begin{cases} \alpha_i := (s_1^{-1} m_1 - s_i^{-1} m_i) \bmod q; \\ \beta_i := (s_1^{-1} r_1 - s_i^{-1} r_i) \bmod q; \\ \kappa_i := k_{1,1} - k_{i,1}. \end{cases} \tag{22}$$

In this case, (18) be this:

$$\begin{cases} \alpha_2 + a\beta_2 - 2^{t_1}\kappa_2 \equiv k_{1,0} - k_{2,0} \pmod q; \\ \alpha_3 + a\beta_3 - 2^{t_1}\kappa_3 \equiv k_{1,0} - k_{3,0} \pmod q; \\ \vdots \\ \alpha_n + a\beta_n - 2^{t_1}\kappa_n \equiv k_{1,0} - k_{n,0} \pmod q. \end{cases} \tag{23}$$

Then, lattice $L$ spanned by row vectors of the following basis matrix be such that:

$$M = \left(\begin{array}{ccc|ccc} 1 & 0 & 0 \cdots 0 & \alpha_2 & \cdots & a_n \\ 0 & 1 & 0 \cdots 0 & \beta_2 & \cdots & \beta_n \\ \hline 0 & 0 & 1 \cdots 0 & 2^{t_1} & \cdots & 0 \\ \vdots & \vdots & \vdots \ddots \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & 0 \cdots 1 & 0 & \cdots & 2^{t_1} \\ \hline 0 & 0 & 0 \cdots 0 & q & \cdots & 0 \\ \vdots & \vdots & \vdots \ddots \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & 0 \cdots 0 & 0 & \cdots & q \end{array}\right) \tag{24}$$

Target element $L$ is a vector:

$$v_0 = (1, a, \kappa_2, ..., \kappa_n, k_{1,0} - k_{2,0}, ..., k_{1,0} - k_{n,0}) = (1, a, \kappa_2, ...., \kappa_n, \lambda_2, ...., \lambda_n) \cdot M \ . \tag{25}$$

To search for a private key $a$ length of $N$ bits, can also apply the methods described above.

The developed approach to obtaining access to private keys from signed messages proves that the DSA (ECDSA) algorithm is vulnerable to side-channel attack, in case when the ephemeral keys of each encrypted message have shared bits in less and/or more significant bits. Figure 3 shows the relationship between the number of shared bits and the number of necessary messages to the success rate.

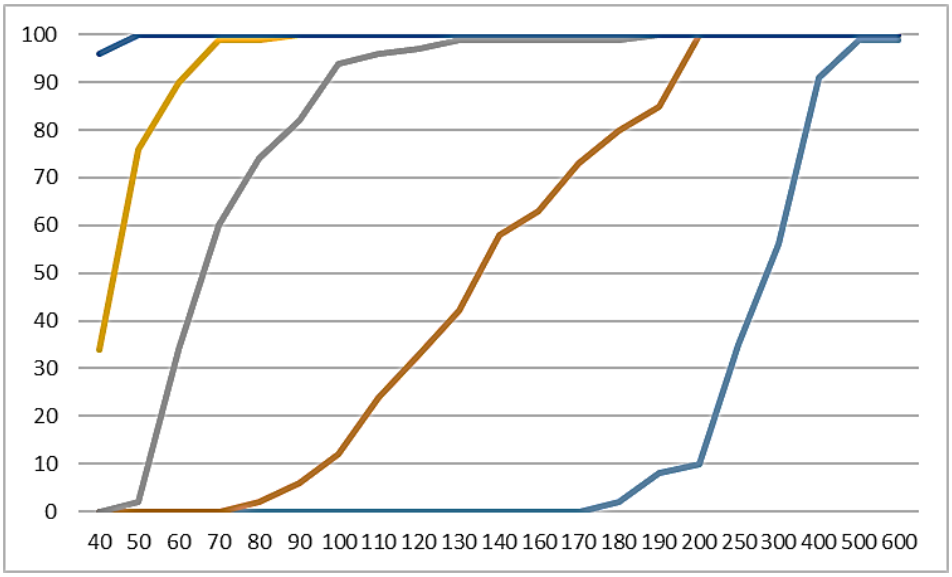


**Figure 3**: Diagram of dependence between the number of shared bits and the number of messages necessary to the percentage of the probability attack success

To achieve this goal, a function for generating an ephemeral key in the form of a pseudocode, which will receive a message ($m$) and private key of the sender ($d$), is proposed – algorithm 1 (Deterministic Ephemeral Key Generation Algorithm):

**function** $(EC)DSA\_SIGN(m,d)$
  $e = H(m)$
  **repeat**
    **repeat**
      $u = HMAC\_SIZE(d,e)$
      $(x,y) = uP \in E(\mathrm{F}_p)$ # *for ECDSA*
      $r = x \bmod q$ # *for ECDSA*
      $r = (g^k \bmod p) \bmod q$ # *for DSA*
    **until** $r \neq 0$
    $s = u^{-1}(e + dr) \bmod q$
  **until** $s \neq 0$
  **return** $(r,s)$.

The feature of the represented function for ephemeral key generation, from others, is the use of an HMAC algorithm, which allows you to generate more reliable random numbers, and prevents attacks by side channels. Due to the use of additional hash function that hides bits of ephemeral key from a potential attack, when other algorithms are presented in various cryptographic libraries use direct operations with the bits of the ephemeral key in the memory registers, which leads to attacks on side channels. The use of a new generation algorithm does not require changes to the message verification procedure, and will provide protection against attacks by side channels, thanks to the use of additional hash function that follows the integrity of the data, and mixes the ephemeral key for greater randomness.

Also, one of the vulnerable points in the creation of signatures (1), (3) is modular inverse operation in the – algorithm 2 (Not constant-time modular inverse algorithm [17]):

**Input:** $a, b \in \mathbf{Z}_{>0}$ *with* $\gcd(a,b) = 1, 0 \le b < a$,

**Output:** $\begin{cases} (b^{-1} \cdot 2^k \bmod a, k) \\ [\log_2(a)] \le k \le 2[\log_2(a)]. \end{cases}$

  $u \leftarrow a,\ v \leftarrow b, r \leftarrow 0, s \leftarrow 1, k \leftarrow 0$
  **while** $v \neq 1$ **do**
    **if** $u = 0 \pmod 2$ **then**
      $u \leftarrow u / 2,\ s \leftarrow 2 \cdot s$
    **else if** $v \equiv 0 \pmod 2$ **then**
      $v \leftarrow v / 2, r \leftarrow 2 \cdot r$
    **else if** $u > v$ **then**
      $u \leftarrow (u - v) / 2, r \leftarrow r + s,\ s \leftarrow 2 \cdot s$
    **else**
      $v \leftarrow (v - u) / 2,\ s \leftarrow r + s, r \leftarrow 2 \cdot r$
    **end if**
    $k \leftarrow k + 1$
  **end while**
  **return** $(s,k)$.

The execution time of the algorithm 2 depends on both values $a$ and $b$, namely the number of iterations of the cycle while, namely the number of iterations of the While cycle, as well as the degree

of two, which should be removed in the end of the algorithm, are significantly different for input data. Therefore, to convert an algorithm 2 into an algorithm performed by a constant time for a given *wn* bit module *m*, it is necessary to save the following requirements:

1. One iteration is always calculated by the same amount of time, it means to calculate all four branches from the algorithm 2 and select the correct values for a constant time. This ensures that the time of execution of one iteration does not depend on the branch taken, but means that the calculation time increases until the calculation of all branches;
2. The algorithm must perform the one-time number of iterations (for a constant time), which means that it is necessary to calculate the worst number of $2wn$ iterations. This can be realized by determining when the algorithm 2 is completed (when $v$=1). Depending on this condition, we create a bit mask and choose the input value for this iteration (when $v$=1), or the value, calculation with iteration with constant time ($v \neq 1$).

After making modifications in the algorithm 2, we will receive a new algorithm 3 that will perform a modular inverse operation for a constant time – algorithm 3 (Constant-time algorithm of modular inverse):

**Input:** $a, b \in \mathbf{Z}_{>0}$ *with* $\gcd(a,b)=1, 0 \le b < a$,

**Output:** $\begin{cases}(b^{-1}\cdot 2^k \bmod a, k)\\ [\log_2(a)] \le k \le 2[\log_2(a)].\end{cases}$

$u \leftarrow a,\ v \leftarrow b, r \leftarrow 0, s \leftarrow 1, k \leftarrow 0$

**for i** $= 1 to\ 2[\log_2(a)]$ **do**

$\mathbf{uv}_{<} \leftarrow sub(u', u, v)$

$\mathbf{uv}_{=} \leftarrow equal(u', 0)$

$\mathbf{d} \leftarrow 0 - \mathbf{uv}_{=} \quad \#\mathbf{d} = \begin{cases}0 \ \ if\ u \neq v\\ 2^{\omega}-1 if\ u = v\end{cases}$

$lshift_1(\tilde{s}, s)$
$add(rs, r, s)$
$rshift_1(\tilde{u}, u)$
$\mathbf{m_1} \leftarrow \mathbf{d} \vee (0 - (\mathbf{u_0} \wedge 1))$
$\mathbf{m_2} \leftarrow bitflip(\mathbf{m_1})$
$select(u, \tilde{u}, \mathbf{m_2}, u, \mathbf{m_1})$
$select(s, \tilde{s}, \mathbf{m_2}, s, \mathbf{m_1})$
} # **if** $u \equiv 0$

$lshift_1(\tilde{r}, r)$
$\mathbf{S} \leftarrow (\mathbf{d} \vee bitflip(\mathbf{m_1}))$
$\mathbf{m_3} \leftarrow S \vee 0 - (\mathbf{v_0} \wedge 1)$
$\mathbf{m_4} \leftarrow bitflip(\mathbf{m_3})$
$rshift_1(\tilde{v}, v)$
$select(v, \tilde{u}, \mathbf{m_4}, v, \mathbf{m_3})$
$select(r, \tilde{r}, \mathbf{m_4}, r, \mathbf{m_3})$
} # **else if** $v \equiv 0$

$\mathbf{S} \leftarrow \mathbf{S} \vee bitflip(m_3)$
$\mathbf{m_5} \leftarrow \mathbf{S} \vee (0 - \mathbf{uv}_{<})$
$\mathbf{m_6} \leftarrow bitflip(\mathbf{m_5})$
$rshift_1(u', u')$
$select(u, u', \mathbf{m_6}, u, \mathbf{m_5})$
$select(r, rs, \mathbf{m_6}, r, \mathbf{m_5})$
$select(r, \tilde{s}, \mathbf{m_6}, s, \mathbf{m_5})$
} # **else if** $u > v$

$\mathbf{S} \leftarrow \mathbf{S} \vee bitflip(\mathbf{m_5})$
$\mathbf{m_7} \leftarrow bitflip(\mathbf{S})$
$sub(\tilde{v}, v, u)$
$rshift_1(\tilde{v}, \tilde{v})$
$select(v, \tilde{v}, \mathbf{m_7}, v, \mathbf{S})$
$select(s, rs, \mathbf{m_7}, s, \mathbf{S})$
$select(r, \tilde{r}, \mathbf{m_7}, r, \mathbf{S})$
} # **else**

$\mathbf{k} \leftarrow ((\mathbf{k} \wedge \mathbf{d}) \vee ((\mathbf{k}+\mathbf{1}) \wedge bitflip(\mathbf{d})))$

**end for**

**return** $(s, \mathbf{k})$.

## 3. Conclusions

The vulnerability type of cryptosystems is determined for side channel attack, which is concluded in blocking the bits of the ephemeral key or determining the signature time of each message. Using the developed method, the signers private key can be determined in the presence of a sufficient number of

messages in less than a minute, bypassing a discrete logarithm problem, which in turn fully compromises the stability of cryptosystems.

Was developed a effective method for protecting cryptosystems to attack by side channels to prevent unauthorized access to encrypted information, based on a new ephemeral key generation algorithm to prevent attacks by side channels, which increases the count a mathematical operation to $2^N$. The use of the developed constant time algorithm of modular inverse, provides a constant number of clock cycles, namely, the number of bits of 256, the number of cycles will not be constant 50-60. When using this algorithm, the number of clock cycles is always 486, which provides resistant to the attack.

## 4. References


[1] A. J. Menezes, P. C. Van Oorschot, and S. A. Vanstone, Handbook of applied cryptography. 1996.

[2] M. Ubaidullah and Q. Makki, "A Review on Symmetric Key Encryption Techniques in Cryptography," Int. J. Comput. Appl., vol. 147, no. 10, 2016, doi: 10.5120/ijca2016911203.

[3] T. Elgamal, "A Public Key Cryptosystem and a Signature Scheme Based on Discrete Logarithms," IEEE Trans. Inf. Theory, vol. 31, no. 4, 1985, doi: 10.1109/TIT.1985.1057074.

[4] M. Kirschmer, J. Voight, Algorithmic enumeration of ideal classes for quaternion orders, SIAM J. Comput. 39 (2010) 1714–1747. URL: http://dx.doi.org/10.1137/080734467. doi:10.1137/080734467.

[5] Roman N. Kvyetnyy, Yevhenii A. Titarchuk, Volodymyr Y. Kotsiubynskyi, Waldemar Wójcik, Nursanat Askarova. Partially homomorphic encryption algorithm based on elliptic curves.- Proc. SPIE 10808, Photonics Applications in Astronomy, Communications, Industry, and High-Energy Physics Experiments 2018, 108082H (1 October 2018); 8 p. doi: 10.1117/12.2501583; https://doi.org/10.1117/12.2501583.

[6] P. Q. Nguyen and P. Q. Nguyen, "Public-key Cryptanalysis," 2008.

[7] P. A. Fouque, M. Tibouchi, and J. C. Zapalowicz, "Recovering private keys generated with weak prngs," in Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics), 2013, vol. 8308 LNCS, doi: 10.1007/978-3-642-45239-0_10.

[8] Y. Ivanchuk. Mechatronic Systems II. Applications in Material Handling Processes and Robotics, edited by Leonid Polishchuk, Orken Mamyrbayev, Konrad Gromaszek, (2021), Taylor & Francis Group, CRC Press, Balkema book Boca Raton, London, New York, Leiden, 352 P. ISBN 978-1-032-10585-7, DOI: 10.1201/9781003225447.

[9] A. A. Yarovyy, L. I. Timchenko, N. I. Kokriatskaia, "Parallel-Hierarchical Computing System for Multi-Level Transformation of Masked Digital Signals," Advances in Electrical and Computer Engineering, vol.12, no.3, pp.13-20, 2012, doi:10.4316/AECE.2012.03002.

[10] D. Boneh and R. Venkatesan, "Hardness of computing the most significant bits of secret keys in diffie-hellman and related schemes," in Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics), 1996, vol. 1109, doi: 10.1007/3-540-68697-5_11.

[11] N. A. Howgrave-Graham and N. P. Smart, "Lattice Attacks on Digital Signature Schemes," Des. Codes, Cryptogr., vol. 23, no. 3, 2001, doi: 10.1023/A:1011214926272.

[12] E. De Mulder, M. Hutter, M. E. Marson, and P. Pearson, "Using Bleichenbacher's solution to the hidden number problem to attack nonce leaks in 384-bit ECDSA: Extended version," J. Cryptogr. Eng., vol. 4, no. 1, 2014, doi: 10.1007/s13389-014-0072-z.

[13] D. Jayasinghe, R. Ragel, and D. Elkaduwe, "Constant time encryption as a countermeasure against remote cache timing attacks," 2012, doi: 10.1109/ICIAFS.2012.6419893.

[14] P. D. Gallagher and C. Romine, "FIPS PUB 186-4 Digital Signature Standard (DSS)," Encycl. Cryptogr. Secur., no. July, 2013.

[15] P. Q. Nguyễn and D. Stehlé, "Floating-point LLL revisited," in Lecture Notes in Computer Science, 2005, vol. 3494, doi: 10.1007/11426639_13.

[16] C. P. Schnorr and M. Euchner, "Lattice basis reduction: Improved practical algorithms and solving subset sum problems," in Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics), 1991, vol. 529 LNCS, doi: 10.1007/3-540-54458-5-51.

[17] B. S. Kaliski, "The Montgomery Inverse and Its Applications," IEEE Trans. Comput., vol. 44, no. 8, 1995, doi: 10.1109/12.403725.