TencentCloud API

Development Guide

Product Documentation

Contents

Development Guide
API for C++

Last updated：2023-03-07 18:16:40

TencentCloud API has been upgraded to v3.0. This version is optimized for performance and deployed in all regions. It supports nearby access and access by region for significantly reduced access latency. In addition, it features more detailed API descriptions and error codes and API-level comments for SDKs, enabling you to use Tencent Cloud services more conveniently and quickly. This document describes how to call APIs for C++.
This version currently supports various Tencent Cloud services such as CVM, CBS, VPC, and TencentDB and will support more services in the future.

Request Structure

1. Service address (endpoint)

TencentCloud API supports access from either a nearby region (such as `cvm.tencentcloudapi.com` for CVM) or a specified region (such as `cvm.ap-guangzhou.tencentcloudapi.com` for CVM in the Guangzhou region). For values of the region parameter, please see the region list in the "Common Parameters" section below. To check whether a region is supported by a specific Tencent Cloud service, please see its "Request Structure" document.

Note:

For latency-sensitive businesses, we recommend you specify a domain name with a region.

2. Communications protocol

All TencentCloud APIs communicate over HTTPS, providing highly secure communications tunnels.

3. Request method

Supported HTTP request methods:
POST (recommended)
GET
`Content-Type` types supported by POST request:
application/json (recommended). The signature algorithm v3 (TC3-HMAC-SHA256) must be used.
application/x-www-form-urlencoded. The signature algorithm v1 (HmacSHA1 or HmacSHA256) must be used.
multipart/form-data (only supported by certain APIs). The signature algorithm v3 (TC3-HMAC-SHA256) must be used.

The size of a GET request packet cannot exceed 32 KB. The size of a POST request cannot exceed 1 MB for the signature algorithm v1 (HmacSHA1 or HmacSHA256) or 10 MB for the signature algorithm v3 (TC3-HMAC-SHA256).

4. Character encoding

UTF-8 encoding is always used.

Common Parameters

Note:
The common parameters are used to identity the user and API signature. They should be carried by each request to initiate properly.

Signature algorithm v3

The signature algorithm v3 (sometimes referred to as "TC3-HMAC-SHA256") is more secure than the signature algorithm v1 (referred to as signature algorithm in certain documents), supports larger request packets and POST JSON format, and has a higher performance. We recommend you use it to calculate signatures. For more information on how to use it, please see below.

Parameter Name	Type	Required	Description
X-TC-Action	String	**Yes**	Name of the API for the desired operation. For the specific value, please see the description of common parameter `Action` in the input parameters in the related API document. For example, the API for querying CVM instance list is `DescribeInstances` .
X-TC-Region	String	-	Region parameter, which is used to identify the region where the data you want to manipulate resides. For values supported for an API, please see the description of common parameter `Region` in the input parameters in related API documentation. **Note: this parameter is not required for some APIs (which will be indicated in related API documentation) and will not take effect even if it is passed.**
X-TC-Timestamp	Integer	**Yes**	The current UNIX timestamp that records the time when the API request was initiated, such as 1529223702. **Note: if the difference between the UNIX timestamp and the server time is greater than 5 minutes, a signature expiration error may occur.**
X-TC-Version	String	**Yes**	Version of the API for the desired operation, such as 2017-03-12 for CVM. For the specific value, please see the description of common parameter `Version` in the input parameters in related API documentation.

Authorization	String	**Yes**	HTTP authentication request header, such as TC3-HMAC-SHA256 Credential=AKIDEXAMPLE/Date/service/tc3_request, SignedHeaders=content-type;host, Signature=72e494ea8**********a96525168 Here, TC3-HMAC-SHA256: signature algorithm, currently fixed as this value. Credential: signature credential. `AKIDEXAMPLE` indicates the `SecretId`. `Date` indicates a UTC date which must match the value of `X-TC-Timestamp` (a common parameter) in UTC format. `service` indicates the name of the service and is generally a domain name prefix; for example, the domain name `cvm.tencentcloudapi.com` means the CVM service, and the value for this service is `cvm`. SignedHeaders: the headers that contain the authentication information. `content-type` and `host` are required. Signature: signature digest. For the calculation process, please see below.
X-TC-Token	String	No	Token used for temporary credentials. It must be used with a temporary key. You can get the temporary key and token by calling a CAM API. No token is required for a long-term key.

Region list

As the supported regions vary by service, please refer to the region list in each service's product documentation for specific details.

For example, you can see the region list of CVM.

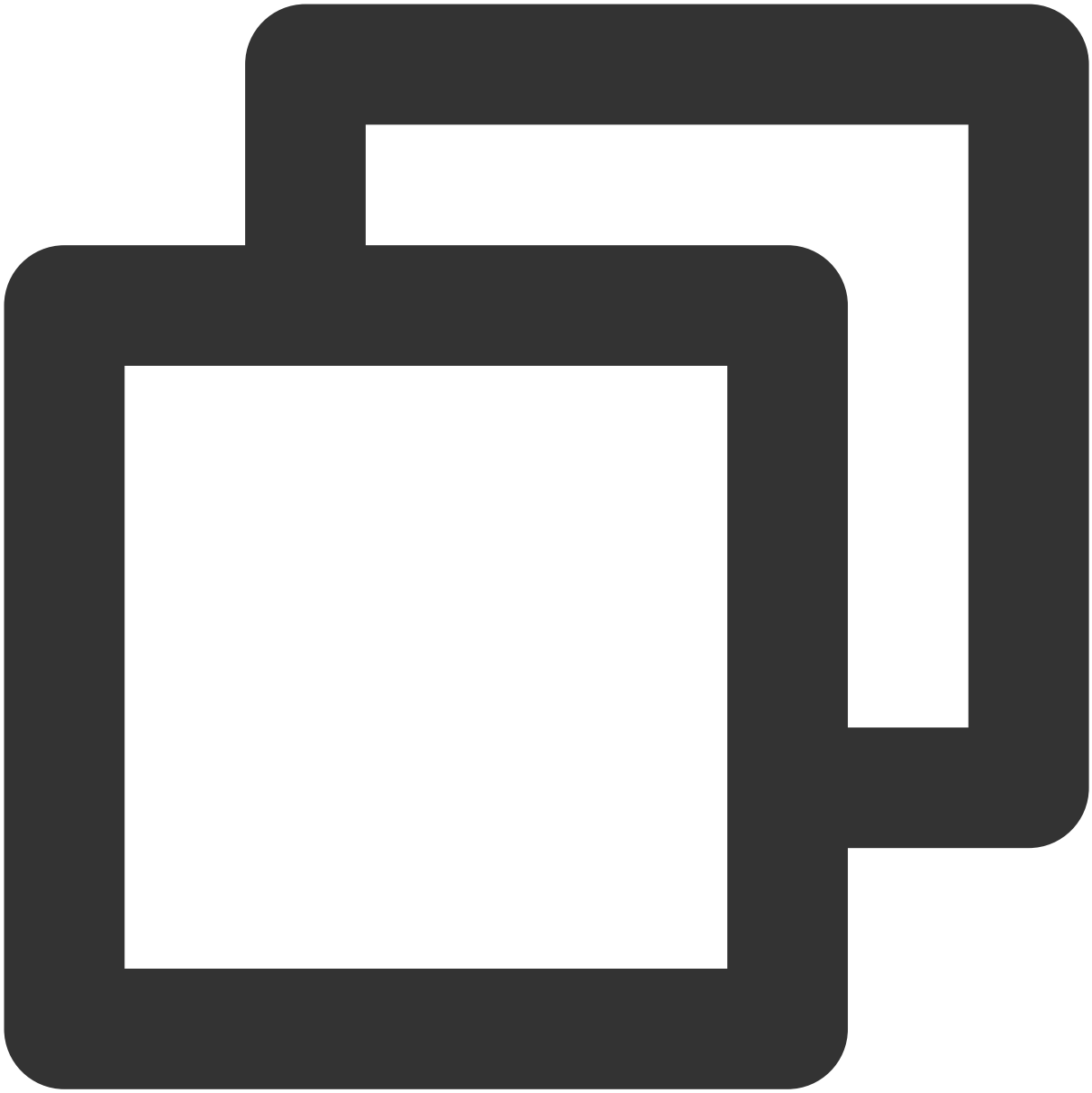
API Call Method for C++

TencentCloud API authenticates every request, that is, the request must be signed with the security credentials in the designated steps. Each request must contain the signature information in the common request parameters and be sent in the specified way and format.

Note:

Currently, API 3.0 signature v1 is not supported for C++.

Suppose your `SecretId` and `SecretKey` are `AKIDz8krbsJ5**********mLPx3EXAMPLE` and `Gu5t9xGAR***********EXAMPLE`, respectively. If you want to view the status of an unnamed instance in the Guangzhou region and have only one data entry returned, the request may be:



```
curl -X POST https://cvm.tencentcloudapi.com \\
-H "Authorization: TC3-HMAC-SHA256 Credential=AKIDz8krbsJ5**********mLPx3EXAMPLE/20
-H "Content-Type: application/json; charset=utf-8" \\
-H "Host: cvm.tencentcloudapi.com" \\
-H "X-TC-Action: DescribeInstances" \\
-H "X-TC-Timestamp: 1551113065" \\
-H "X-TC-Version: 2017-03-12" \\
-H "X-TC-Region: ap-guangzhou" \\
-d '{"Limit": 1, "Filters": [{"Values": ["\\u672a\\u547d\\u540d"], "Name": "instanc
```

Step 1. Apply for security credentials

In this document, the security credential used is a key pair, which consists of a `SecretId` and a `SecretKey`.

Each user can have up to two key pairs.

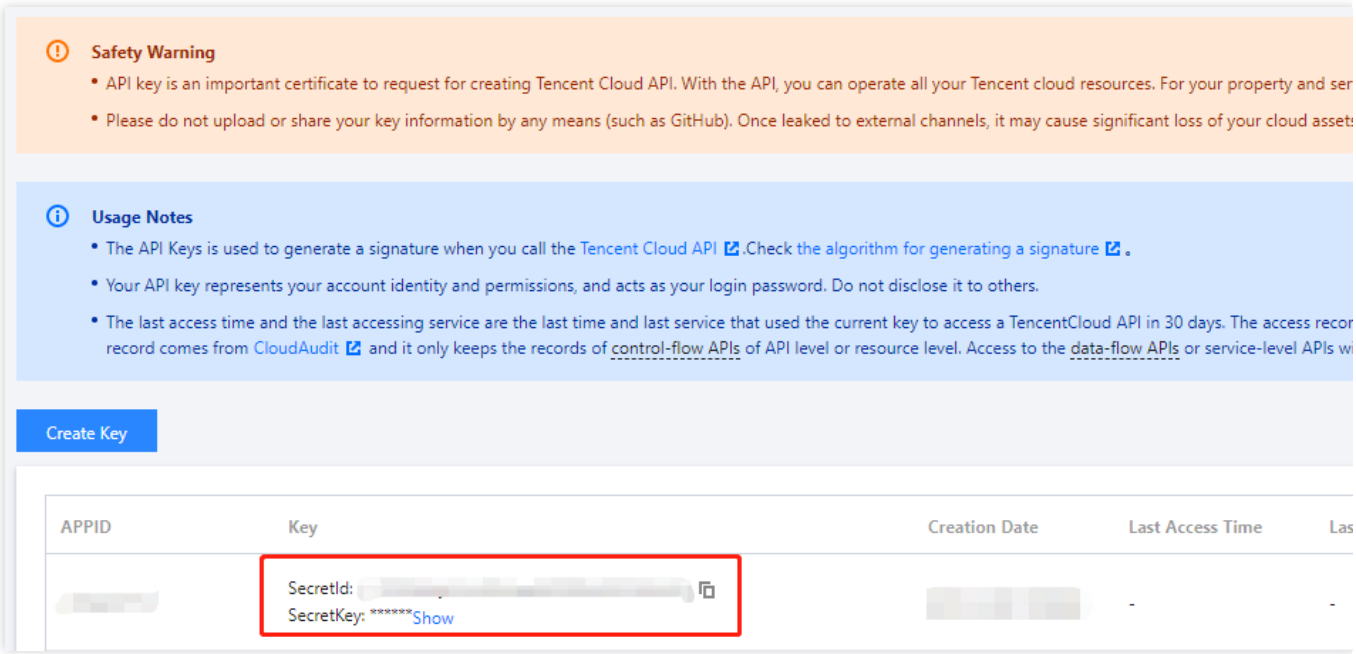
SecretId: identifies the user that calls an API, which is similar to a username.

SecretKey: authenticates the user that calls the API, which is similar to a password.

Note:

You must keep your security credentials private and avoid disclosure; otherwise, your assets may be compromised. If they are disclosed, please disable them as soon as possible.

Go to the API key management page to get API keys as shown below:

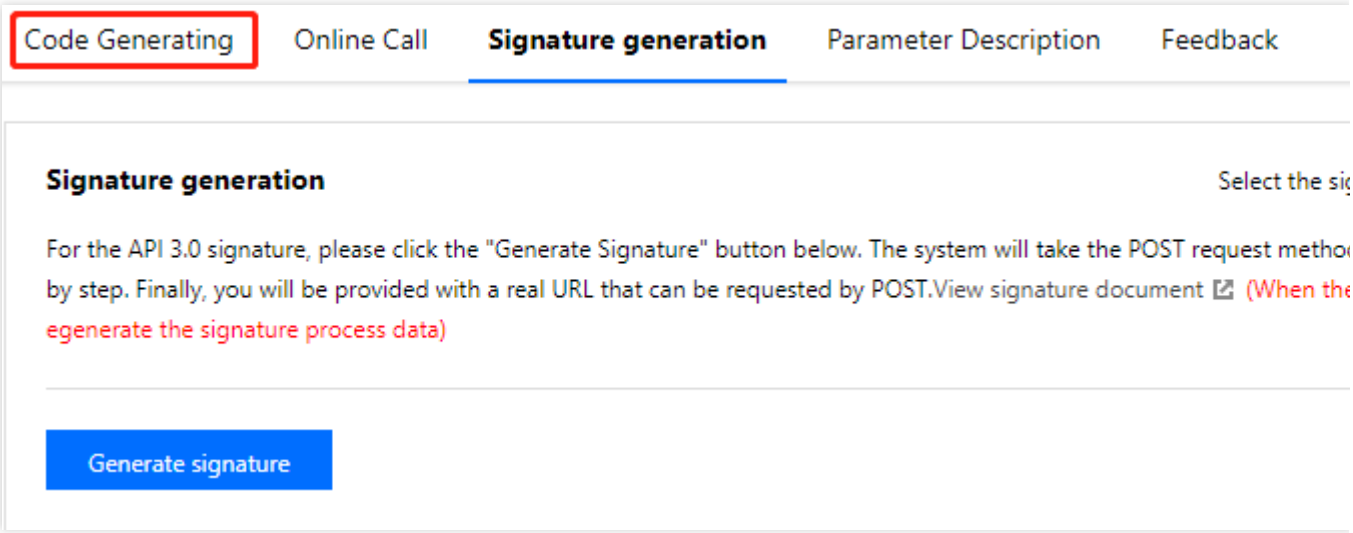


Step 2. Get an API 3.0 signature v3

The signature algorithm v3 (TC3-HMAC-SHA256) is compatible with the previous signature algorithm v1 and more secure, supports larger request packets and POST JSON format, and has a higher performance. We recommend you use it to calculate signatures as shown below:

Note:

If you are using the signature algorithm for the first time, we recommend you use the "signature string generation" feature in API Explorer and select "API 3.0 signature v3" as the signature version, which can generate a signature for demonstration and verification. Plus, it can also generate SDK code directly. Seven common open-source programming language SDKs are available for TencentCloud API, including Python, Java, PHP, Go, Node.js, .NET, and C++.

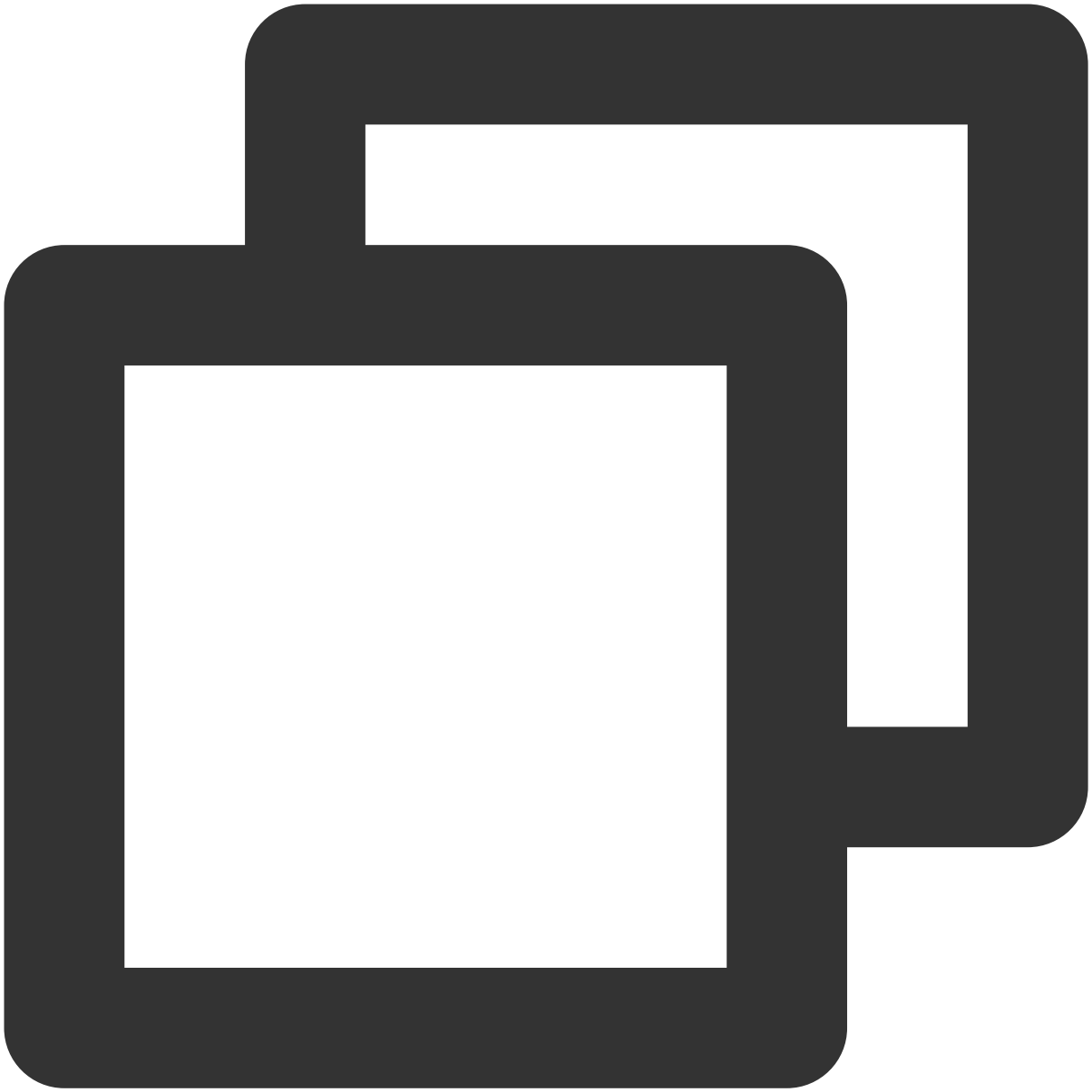


TencentCloud API supports both GET and POST requests. For the GET method, only the `Content-Type: application/x-www-form-urlencoded` protocol format is supported. For the POST method, `Content-Type: application/json` and `Content-Type: multipart/form-data` are supported. The JSON format is supported by all business APIs, while the multipart format is supported only by specific APIs (in this case, an API cannot be called in JSON format). For more information, please see the specific business API document. We recommend you use the POST method because the two methods generate the same results, but the GET method only supports request packets below 32 KB in size.

The following describes how to calculate a signature by calling the DescribeInstances API. This API is chosen because:

1. The CVM API is enabled by default, and this API is often used.
2. It is read-only and does not change the status of existing resources.
3. It covers many types of parameters so that it is easy to show how to use an array that contains data structures.

1. Concatenate the canonical request string

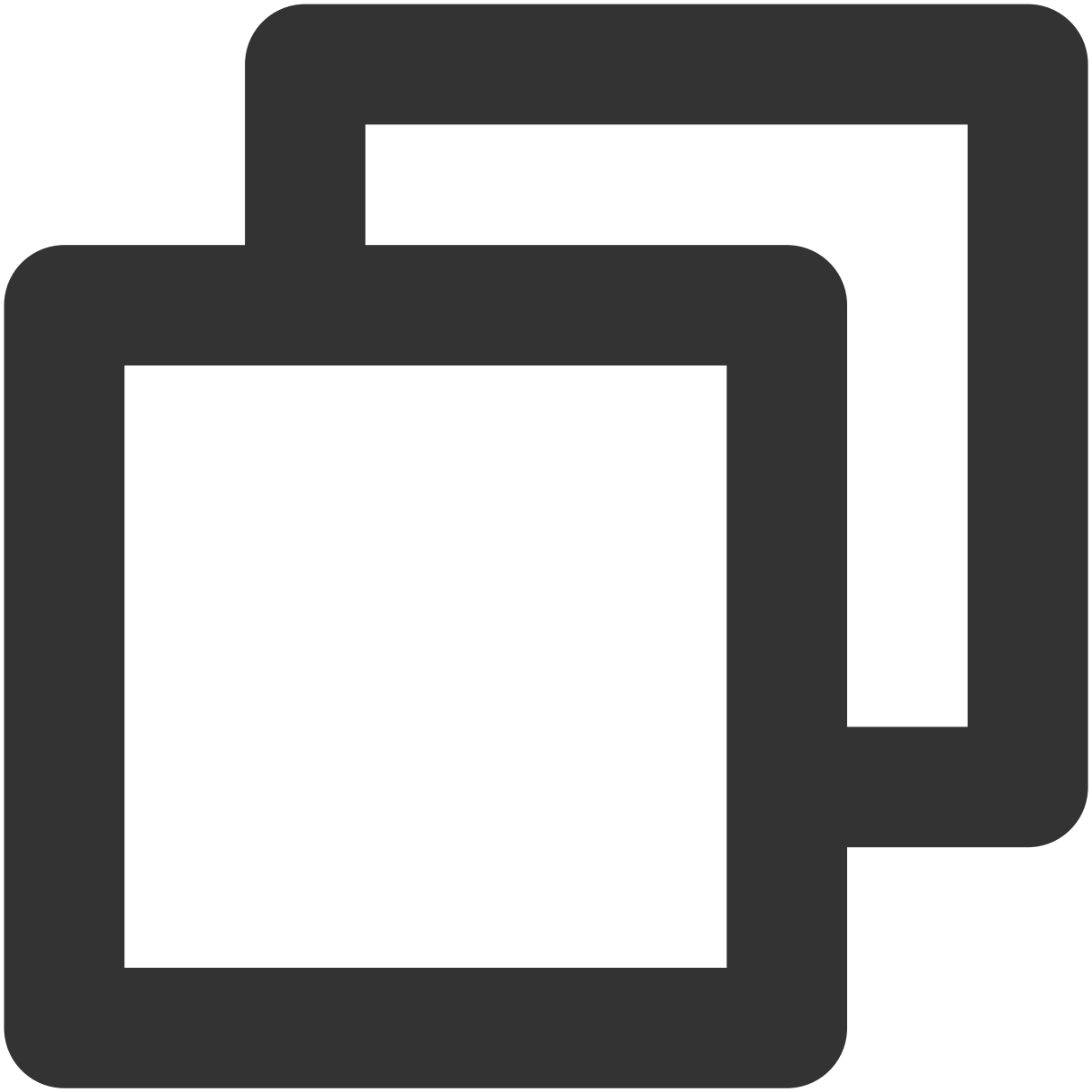


```
CanonicalRequest =
    HTTPRequestMethod + '\\n' +
    CanonicalURI + '\\n' +
    CanonicalQueryString + '\\n' +
    CanonicalHeaders + '\\n' +
    SignedHeaders + '\\n' +
    HashedRequestPayload
```

Field	Description

HTTPRequestMethod	HTTP request method (GET or POST). This example uses `POST` .
CanonicalURI	URI parameter. Slash ("/") is used for API 3.0.
CanonicalQueryString	Query string in the URL of the originating HTTP request. This is always an empty string POST requests and is the string after the question mark (?) for GET requests such as `Limit=10&Offset=0` . Note: `CanonicalQueryString` must be URL-encodec instructed in RFC 3986 with the UTF-8 character set. The applicable standard program language library is recommended. All special characters must be encoded and capitaliz
CanonicalHeaders	Header information for signature calculation, including at least `host` and `content type` . Custom headers can also be added to the signature process to improve the uniqueness and security of the request. Concatenation rules: both the key and value of header should be converted to lowercase with the leading and trailing spaces removed that they are concatenated in the `key:value\\n` format. If there are multiple heade they should be sorted in ASCII ascending order by header key (lowercase). The calcula result in this example is `content-type:application/json; charset=utf-8\\nhost:cvm.tencentcloudapi.com\\n` . Note: `content-type` must mat the content that is actually sent. In some programming languages, a `charset` value automatically added even if it is not specified. In this case, the request sent will be differ from the one signed, and the sever will return a signature verification failure.
SignedHeaders	Header information for signature calculation, indicating the request headers that are inv in the signature process. The request headers must correspond to the headers in `CanonicalHeaders` . `Content-type` and `host` are required headers. Concatenation rules: both the key and value of a header should be converted to lowerca there are multiple headers, they should be sorted in ASCII ascending order by header k (lowercase) and separated by semicolons (;). The value in this example is `content-type;host` .
HashedRequestPayload	Hash value of `Requestpayload` (i.e., the request body, such as `{"Limit": 1, "Filters": [{"Values": ["\\u672a\\u547d\\u540d"], "Name": "instance-name"}]}` in this example). The pseudo-code for calculation is `Lowercase(HexEncode(Hash.SHA256(RequestPayload)))` , which means t SHA256 hashing is performed on the payload of the HTTP request, then hexadecimal encoding is performed, and finally the encoded string is converted to lowercase letters. GET requests, `RequestPayload` is always an empty string. The calculation result i example is `35e9c5b0e3ae67532d3c9f17ead6c90222632e5b1ff7f6e89887f1398934f(`

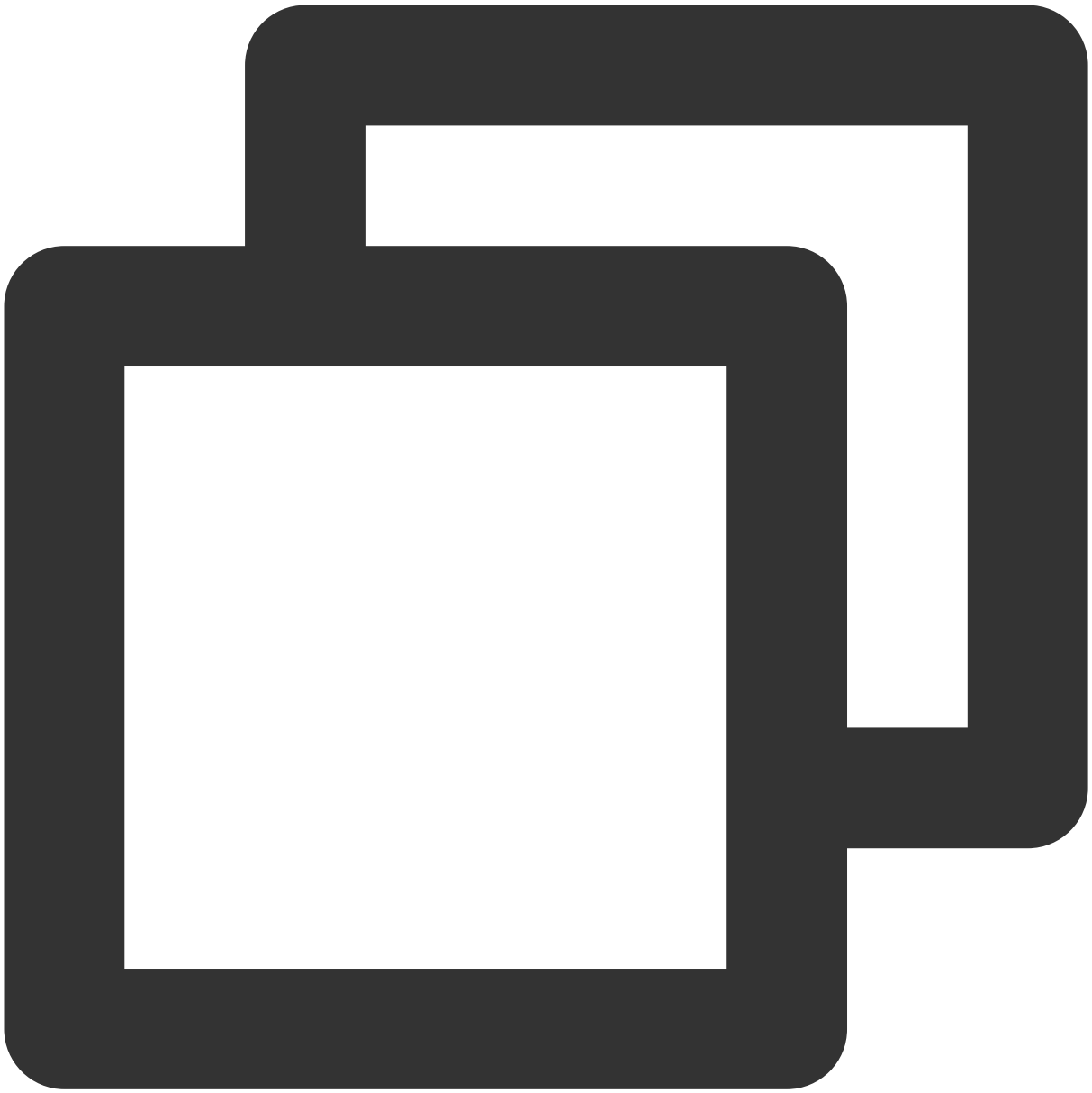
According to the rules above, the canonical request string obtained in the example is as follows:



```
POST
/

content-type:application/json; charset=utf-8
host:cvm.tencentcloudapi.com

content-type;host
35e9c5b0e3ae67532d3c9f17ead6c90222632e5b1ff7f6e89887f1398934f064
```

2. Concatenate the string to sign

Concatenate the string to sign in the following format:



```
StringToSign =
    Algorithm + \\n +
    RequestTimestamp + \\n +
    CredentialScope + \\n +
    HashedCanonicalRequest
```

Field	Description
Algorithm	Signature algorithm, which is always `TC3-HMAC-SHA256` currently.

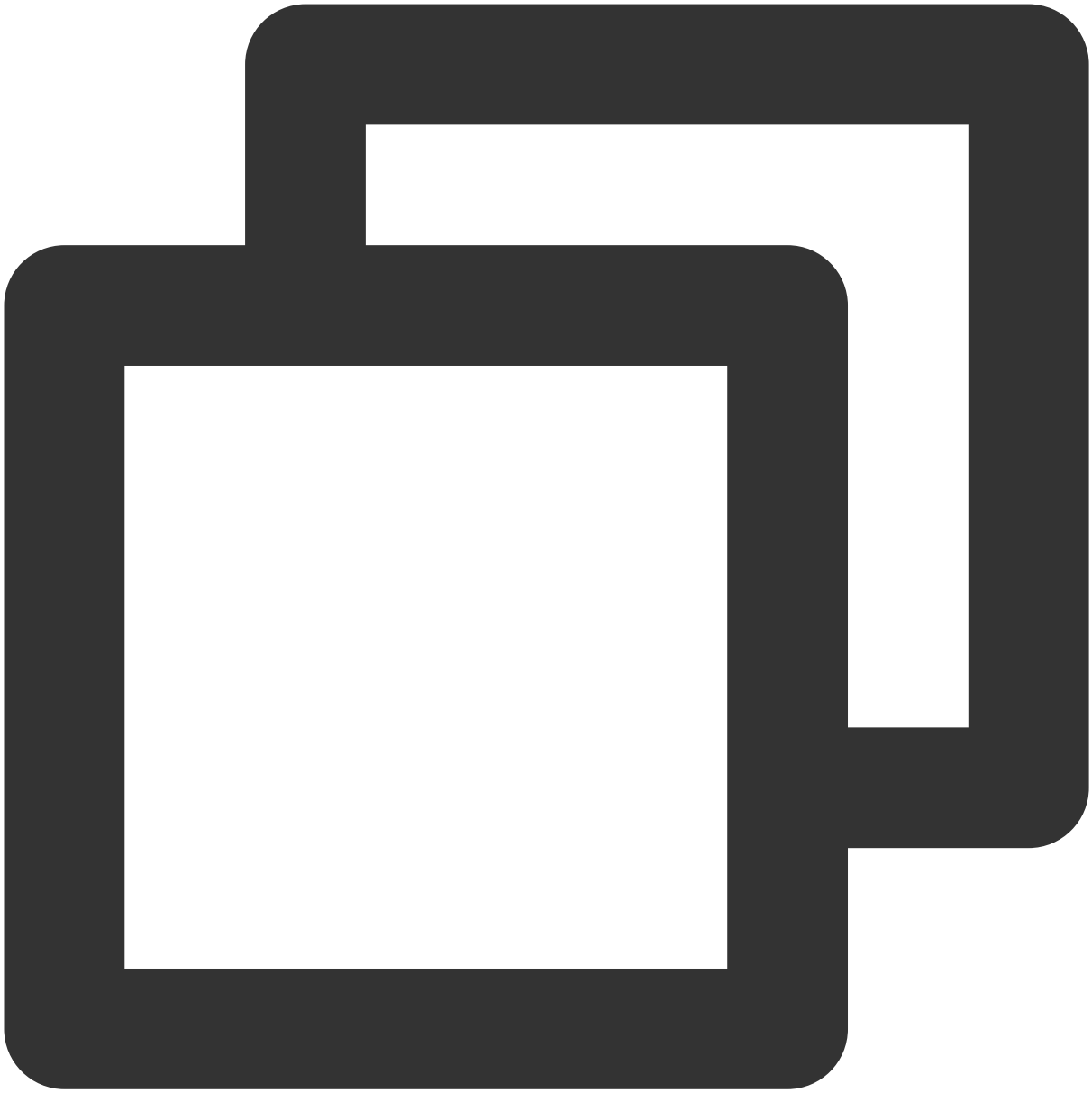
RequestTimestamp	Request timestamp, i.e., the value of the common parameter `X-TC-Timestamp` in request header. It is the UNIX timestamp of the current time in seconds, such as `1551113065` in this example.
CredentialScope	Scope of the credential in the format of `Date/service/tc3_request`, including date, requested service, and termination string (tc3_request). **`Date` indicates a U date, which should match the UTC date converted by the common parameter `TC-Timestamp`.** `service` is the service name, which should match the domain n of the service called. The calculation result in this example is `2019-02-25/cvm/tc3_request`.
HashedCanonicalRequest	Hash value of the canonical request string concatenated in the steps above. The pseu code for calculation is `Lowercase(HexEncode(Hash.SHA256(CanonicalRequest)))`. The calcu result in this example is `5ffe6a04c0664d6b969fab9a13bdab201d63ee709638e2749d62a09ca18d`

Note:

`Date` must be calculated from the timestamp `X-TC-Timestamp` and the time zone is UTC+0. If you add the local time zone information (such as UTC+8) in the system, calls can succeed both day and night but will definitely fail at 00:00. For example, if the timestamp is 1551113065 and the time in UTC+8 is 2019-02-26 00:44:25, the UTC+0 date in the calculated `Date` value should be 2019-02-25 instead of 2019-02-26.

`Timestamp` must be the same as your current system time, and your system time must be in sync with the UTC time. If the difference between the timestamp and your current system time is greater than five minutes, the request will fail. If your system time is out of sync with the UTC time for a prolonged period, the request will fail, and a signature expiration error will be returned.

According to the rules above, the string to sign obtained in the example is as follows:

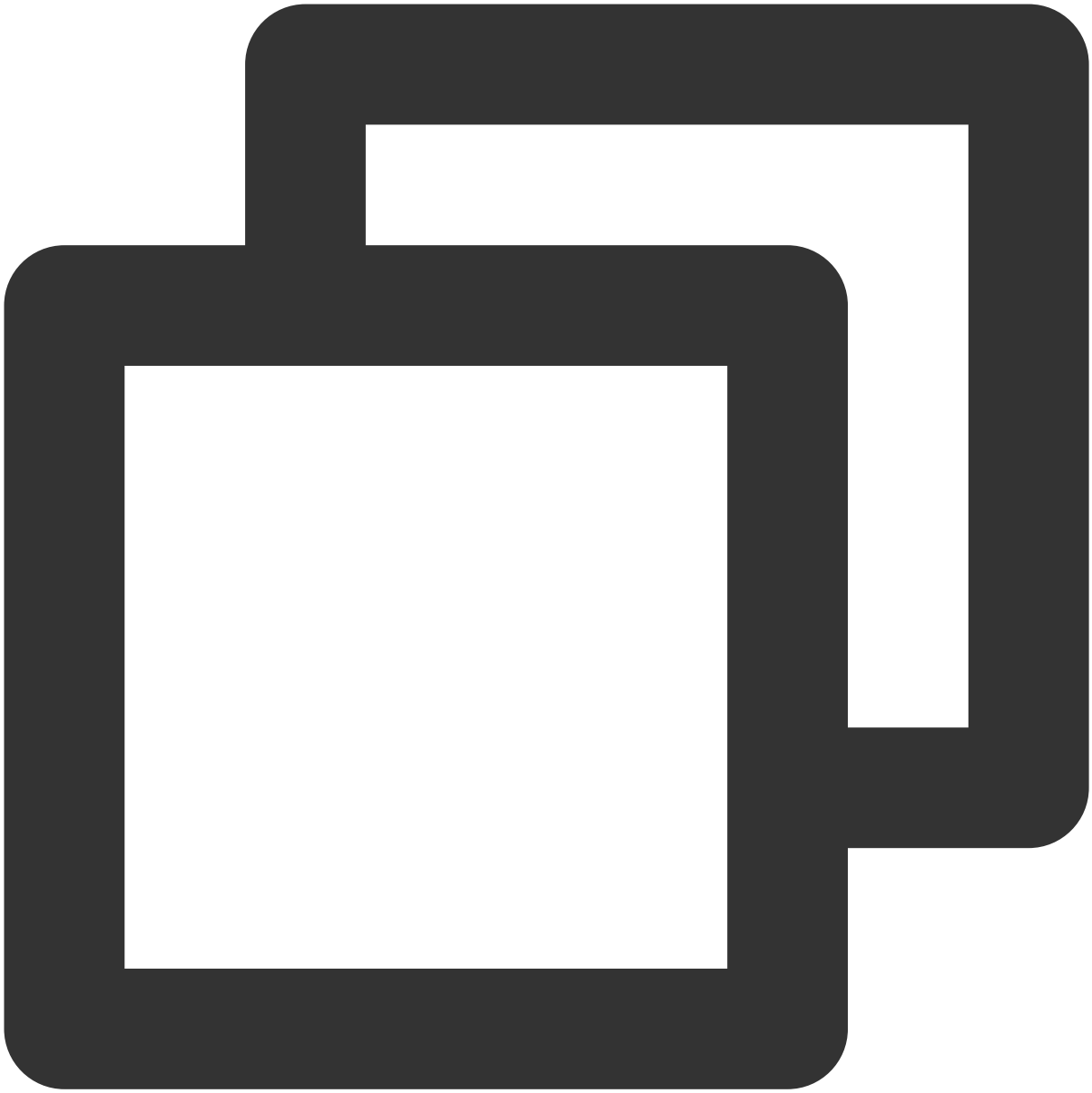


```
TC3-HMAC-SHA256
1551113065
2019-02-25/cvm/tc3_request
5ffe6a04c0664d6b969fab9a13bdab201d63ee709638e2749d62a09ca18d7031
```

3. Calculate the signature (pseudocode)

Please see the following sample code:

1. Calculate the derived signature key with the following pseudocode:



```
#include <tencentcloud/core/Sign.h>
#include <tencentcloud/core/utils/Utils.h>


using namespace TencentCloud;
using namespace std;

string Sign::Tc3Sign(const string &credDate, const string &serviceName, const strin
{
    string kKey = "TC3"+this->m_secretKey;
    string kDate = Utils::HmacSha256(kKey, credDate);
```

```
    string kService = Utils::HmacSha256(kDate, serviceName);
    string kSigning = Utils::HmacSha256(kService, "tc3_request");
    return Utils::HexEncode(Utils::HmacSha256(kSigning, signStr));
}
```

The derived key `SecretDate` , `SecretService` , and `SecretSigning` are binary data and may contain non-printable characters. Intermediate results are not displayed here.

Note:

The order of the parameters in the HMAC library functions may vary by programming language. In the pseudo code here, key parameters are at the second half, and the actual requirement of the used programming language shall prevail. Generally, standard library functions will provide calculated values in binary format, which is also used here. They will also provide print-friendly calculated values in hexadecimal format, which will be used in calculating the signature result below.

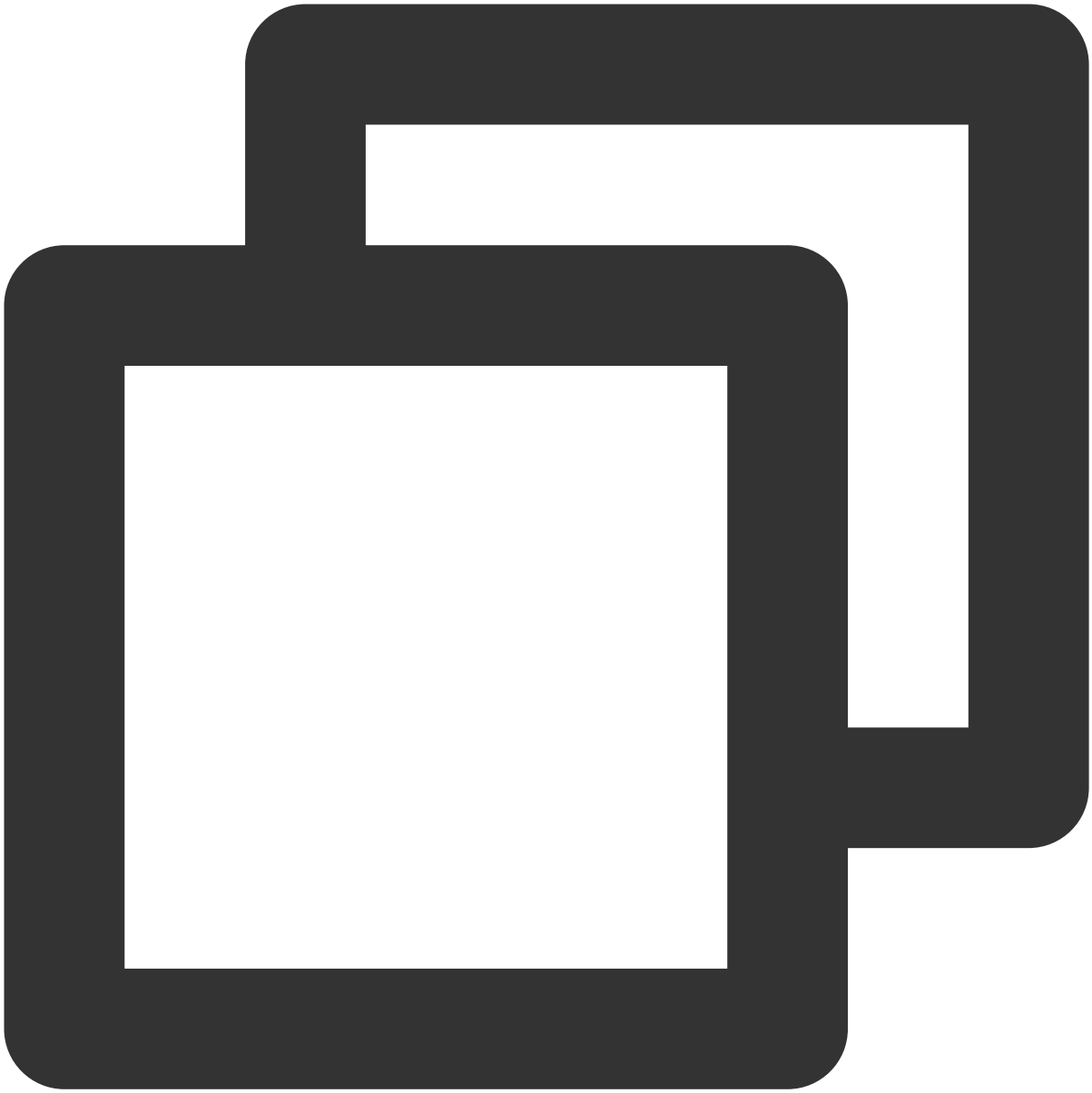
Field	Description
m_secretKey	Original `SecretKey` , i.e., `Gu5t9xGAR**********EXAMPLE` .
credDate	Value of the `Date` field in `Credential` , such as `2019-02-25` in this example.
serviceName	Value of the `Service` field in `Credential` , such as `cvm` in this example.
signStr	String to be signed.

2. Calculate the signature

The calculation result in this example is `72e494ea8****************************************a96525168` .

4. Concatenate the Authorization string

Concatenate the `Authorization` string in the following format:

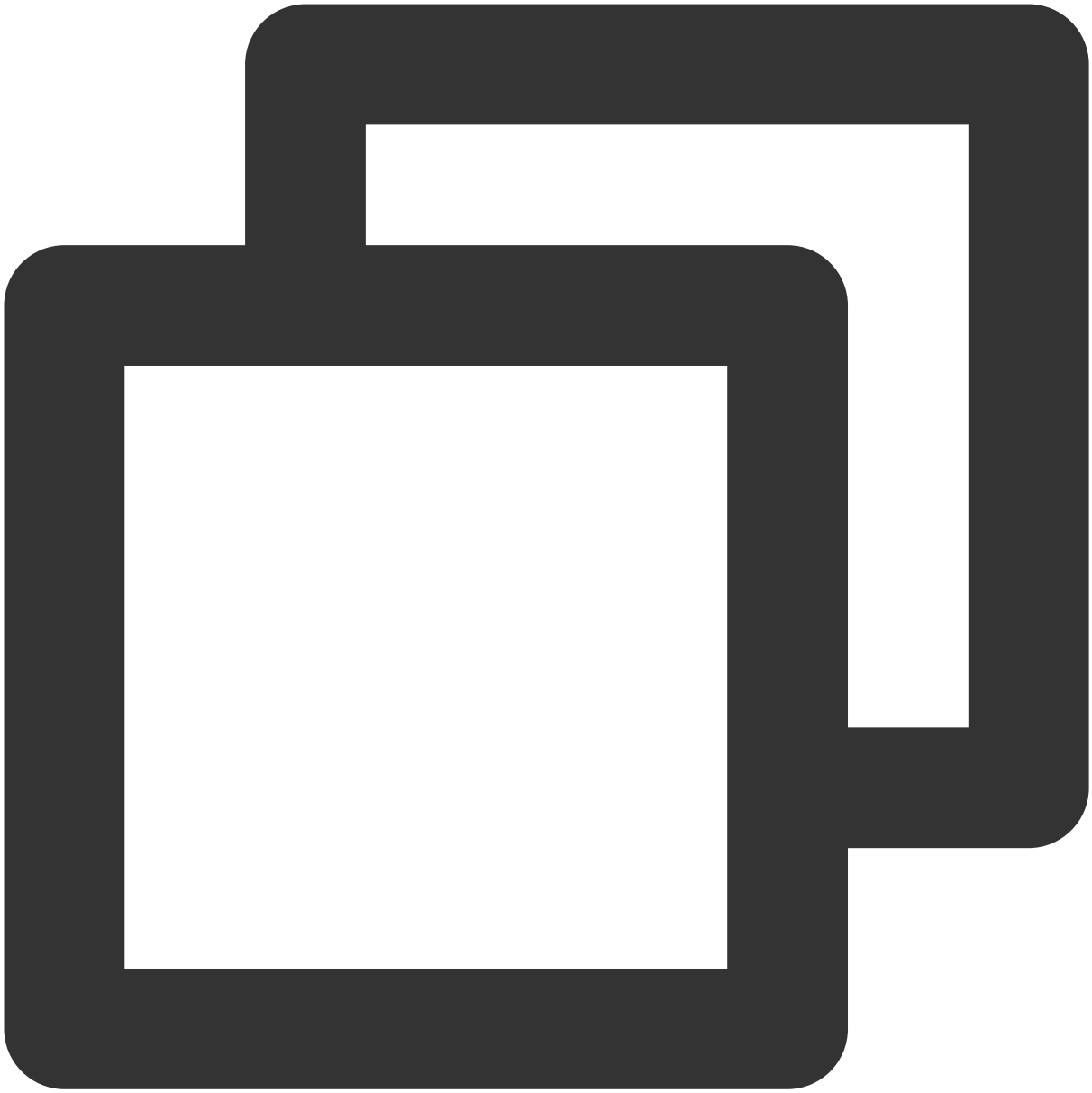


```
Authorization =
    Algorithm + ' ' +
    'Credential=' + SecretId + '/' + CredentialScope + ', ' +
    'SignedHeaders=' + SignedHeaders + ', ' +
    'Signature=' + Signature
```

Field	Description
Algorithm	Signature algorithm, which is always `TC3-HMAC-SHA256` .
SecretId	

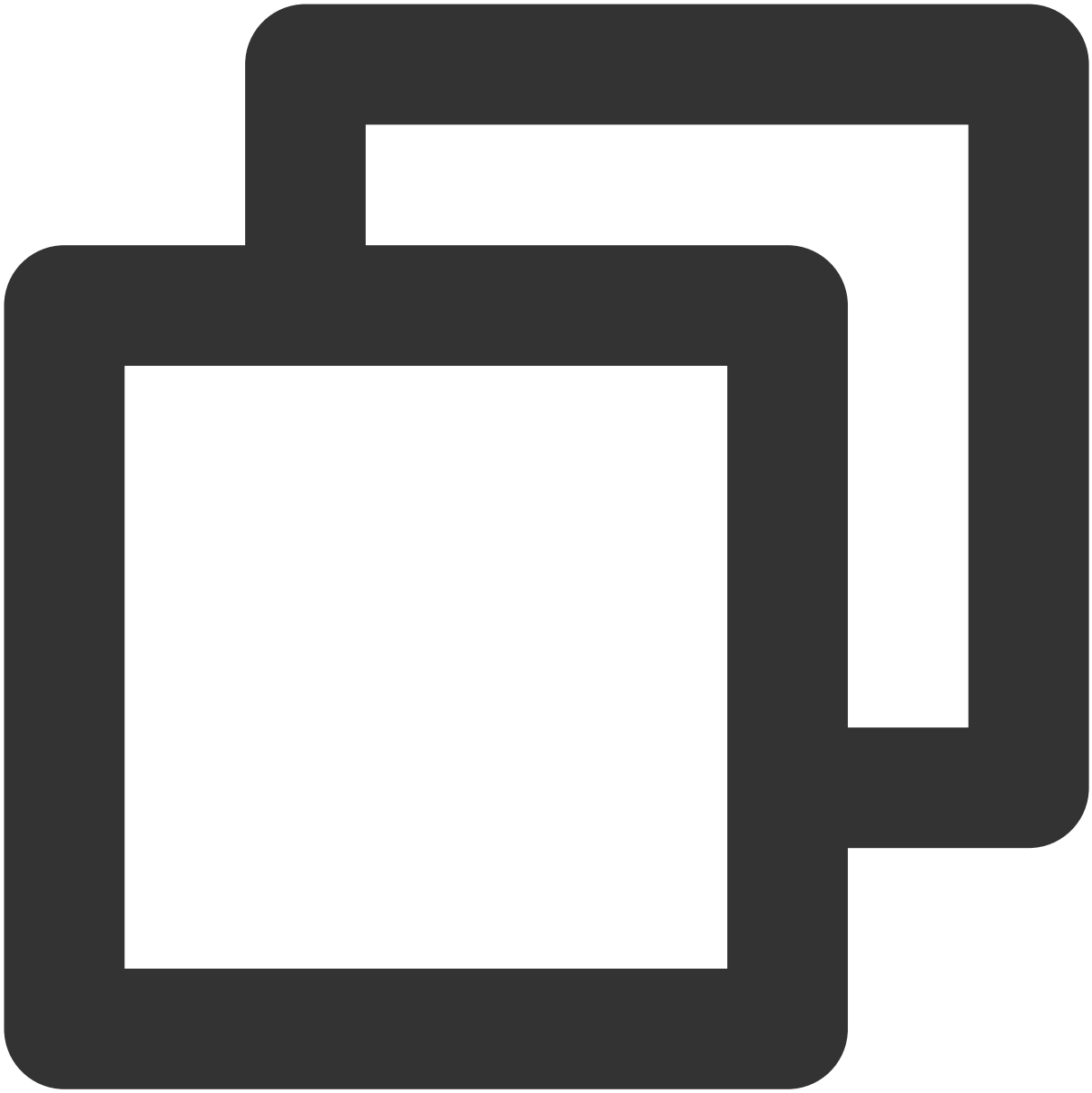
	`SecretId` in the key pair, i.e., `AKIDz8krbsJ5*********mLPx3EXAMPLE` .
CredentialScope	Credential scope (see above). The calculation result in this example is `2019-02-25/cvm/tc3_request` .
SignedHeaders	Header information for signature calculation (see above), such as `content-type;host` in this example.
Signature	Signature value. The calculation result in this example is `72e494ea8****************************************a96525168` .

According to the rules above, the values obtained in this example are:



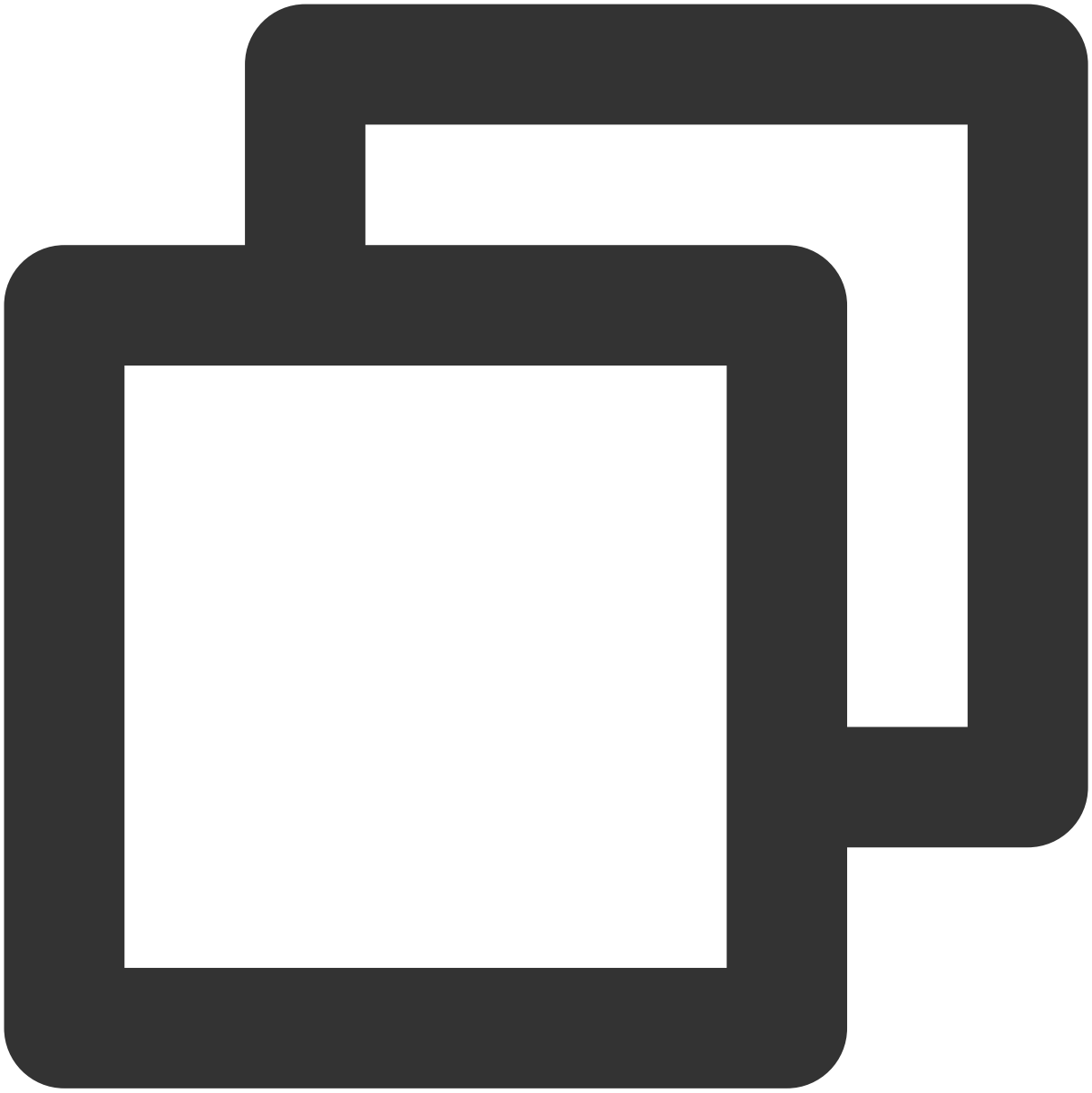
```
TC3-HMAC-SHA256 Credential=AKIDz8krbsJ5**********mLPx3EXAMPLE/2019-02-25/cvm/tc3_re
```

The complete call information is as follows:



```
POST https://cvm.tencentcloudapi.com/
Authorization: TC3-HMAC-SHA256 Credential=AKIDz8krbsJ5**********mLPx3EXAMPLE/2019-0
Content-Type: application/json; charset=utf-8
Host: cvm.tencentcloudapi.com
X-TC-Action: DescribeInstances
X-TC-Version: 2017-03-12
X-TC-Timestamp: 1551113065
X-TC-Region: ap-guangzhou

{"Limit": 1, "Filters": [{"Values": ["\\u672a\\u547d\\u540d"], "Name": "instance-na
```

5. Sample API 3.0 signature v3



```
#include <iostream>
#include <iomanip>
#include <sstream>
#include <string>
#include <stdio.h>
#include <time.h>
#include <openssl/sha.h>
#include <openssl/hmac.h>
```

```
using namespace std;

string get_data(int64_t &timestamp)
{
    string utcDate;
    char buff[20] = {0};
    // time_t timenow;
    struct tm sttime;
    sttime = *gmtime(&timestamp);
    strftime(buff, sizeof(buff), "%Y-%m-%d", &sttime);
    utcDate = string(buff);
    return utcDate;
}
string int2str(int64_t n)
{
    std::stringstream ss;
    ss << n;
    return ss.str();
}
string sha256Hex(const string &str)
{
    char buf[3];
    unsigned char hash[SHA256_DIGEST_LENGTH];
    SHA256_CTX sha256;
    SHA256_Init(&sha256);
    SHA256_Update(&sha256, str.c_str(), str.size());
    SHA256_Final(hash, &sha256);
    std::string NewString = "";
    for(int i = 0; i < SHA256_DIGEST_LENGTH; i++)
    {
        snprintf(buf, sizeof(buf), "%02x", hash[i]);
        NewString = NewString + buf;
    }
    return NewString;
}
string HmacSha256(const string &key, const string &input)
{
    unsigned char hash[32];

    HMAC_CTX *h;
#if OPENSSL_VERSION_NUMBER < 0x10100000L
    HMAC_CTX hmac;
    HMAC_CTX_init(&hmac);
    h = &hmac;
#else
    h = HMAC_CTX_new();
#endif
```

```
    HMAC_Init_ex(h, &key[0], key.length(), EVP_sha256(), NULL);
    HMAC_Update(h, ( unsigned char* )&input[0], input.length());
    unsigned int len = 32;
    HMAC_Final(h, hash, &len);

#if OPENSSL_VERSION_NUMBER < 0x10100000L
    HMAC_CTX_cleanup(h);
#else
    HMAC_CTX_free(h);
#endif

    std::stringstream ss;
    ss << std::setfill('0');
    for (int i = 0; i < len; i++)
    {
        ss  << hash[i];
    }

    return (ss.str());
}
string HexEncode(const string &input)
{
    static const char* const lut = "0123456789abcdef";
    size_t len = input.length();

    string output;
    output.reserve(2 * len);
    for (size_t i = 0; i < len; ++i)
    {
        const unsigned char c = input[i];
        output.push_back(lut[c >> 4]);
        output.push_back(lut[c & 15]);
    }
    return output;
}

int main()
{
    // Key parameter
    string SECRET_ID = "AKIDz8krbsJ5**********mLPx3EXAMPLE";
    string SECRET_KEY = "Gu5t9xGAR***********EXAMPLE";

    string service = "cvm";
    string host = "cvm.tencentcloudapi.com";
    string region = "ap-guangzhou";
    string action = "DescribeInstances";
```

```
    string version = "2017-03-12";
    int64_t timestamp = 1551113065;
    string date = get_data(timestamp);

    // ************* Step 1. Concatenate the canonical request string *************
    string httpRequestMethod = "POST";
    string canonicalUri = "/";
    string canonicalQueryString = "";
    string canonicalHeaders = "content-type:application/json; charset=utf-8\\nhost:
    string signedHeaders = "content-type;host";
    string payload = "{\\"Limit\\": 1, \\"Filters\\": [{\\"Values\\": [\\"\\\\u672a
    string hashedRequestPayload = sha256Hex(payload);
    string canonicalRequest = httpRequestMethod + "\\n" + canonicalUri + "\\n" + ca
            + canonicalHeaders + "\\n" + signedHeaders + "\\n" + hashedRequestPaylo
    cout << canonicalRequest << endl;
    cout << "-----------------------" << endl;

    // ************* Step 2. Concatenate the string to sign *************
    string algorithm = "TC3-HMAC-SHA256";
    string RequestTimestamp = int2str(timestamp);
    string credentialScope = date + "/" + service + "/" + "tc3_request";
    string hashedCanonicalRequest = sha256Hex(canonicalRequest);
    string stringToSign = algorithm + "\\n" + RequestTimestamp + "\\n" + credential
    cout << stringToSign << endl;
    cout << "-----------------------" << endl;

    // ************* Step 3. Calculate the signature ***************
    string kKey = "TC3" + SECRET_KEY;
    string kDate = HmacSha256(kKey, date);
    string kService = HmacSha256(kDate, service);
    string kSigning = HmacSha256(kService, "tc3_request");
    string signature = HexEncode(HmacSha256(kSigning, stringToSign));
    cout << signature << endl;
    cout << "-----------------------" << endl;

    // ************* Step 4. Concatenate the `Authorization` string *************
    string authorization = algorithm + " " + "Credential=" + SECRET_ID + "/" + cred
            + "SignedHeaders=" + signedHeaders + ", " + "Signature=" + signature;
    cout << authorization << endl;
    cout << "-----------------------" << endl;

    string headers = "curl -X POST https://" + host + "\\n"
                   + " -H \\"Authorization: " + authorization + "\\n"
                   + " -H \\"Content-Type: application/json; charset=utf-8\\"" + "\
                   + " -H \\"Host: " + host + "\\n"
                   + " -H \\"X-TC-Action: " + action + "\\n"
                   + " -H \\"X-TC-Timestamp: " + RequestTimestamp + "\\n"
```

```
                    + " -H \\"X-TC-Version: " + version + "\\n"
                    + " -H \\"X-TC-Region: " + region + "\\n"
                    + " -d '" + payload;
    cout << headers << endl;
    return 0;
};
```

API 2.0 Signature

This signature version has been disused. We recommend you use **API 3.0 signature** with better performance. If you still need to use it, please go to API Explorer > **Signature Generation** and select **API 2.0 Signature** as the signature version.

Signature Failure

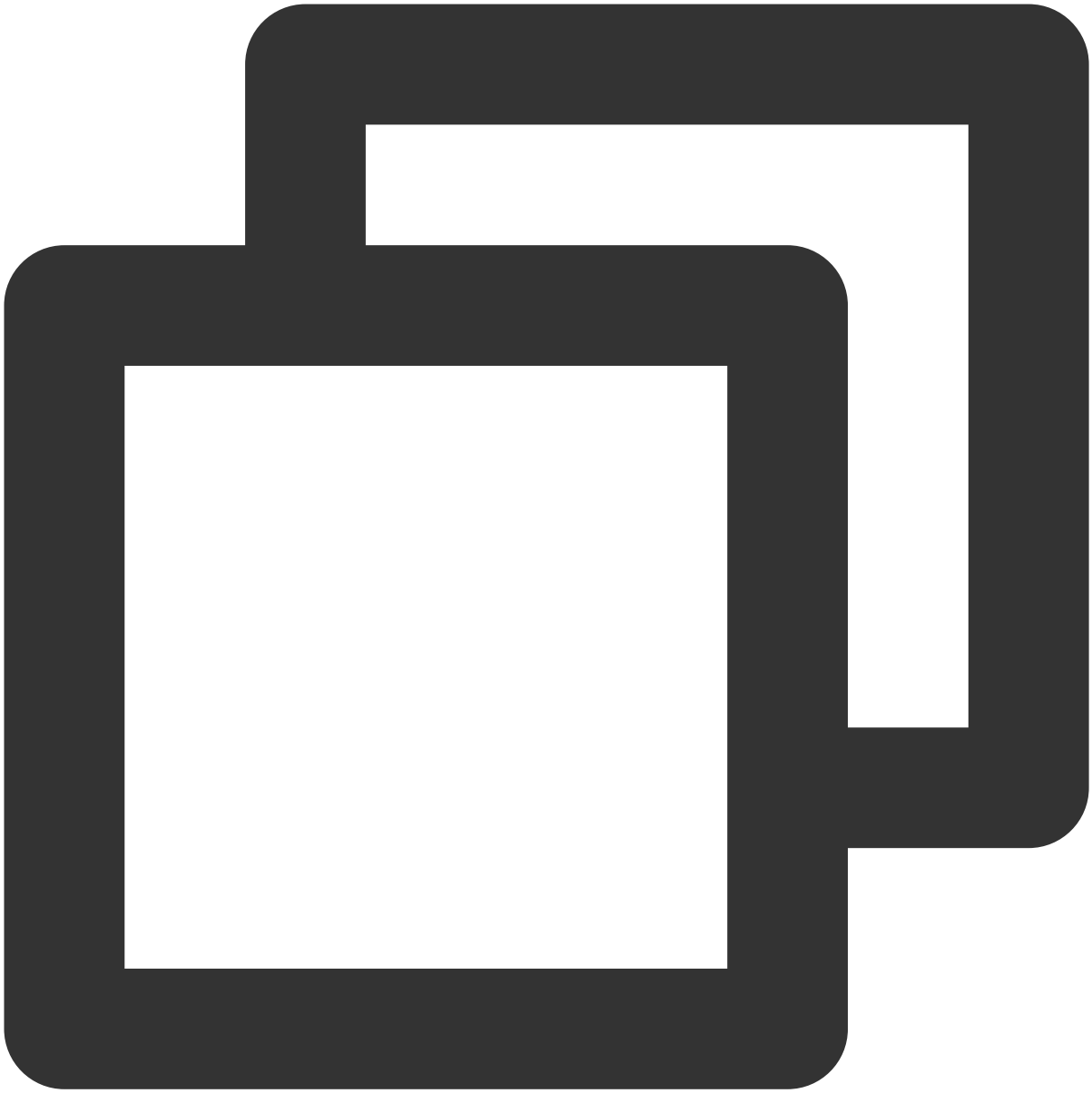
The following error codes may be returned for signature failure. Please resolve the errors accordingly.

Error Code	Error Description
AuthFailure.SignatureExpire	The signature expired. The difference between the `Timestamp` and the server time cannot be greater than five minutes.
AuthFailure.SecretIdNotFound	The key does not exist. Log in to the console and check whether it is disabled or you copied fewer or more characters.
AuthFailure.SignatureFailure	Signature error. It is possible that the signature is calculated incorrectly, the signature does not match the content that is actually sent, or the `SecretKey` is incorrect.
AuthFailure.TokenFailure	Temporary credential token error.
AuthFailure.InvalidSecretId	Invalid key (not TencentCloud API key type).

Returned Result

Successful response

For example, when calling the CVM API `DescribeInstancesStatus` (version: 2017-03-12) to view the status of instances, if the request succeeds, you may see the following response:



```
{
    "Response": {``
        "TotalCount": 0,
        "InstanceStatusSet": [],
        "RequestId": "b5b41468-520d-4192-b42f-595cc34b6c1c"
    }
}
```

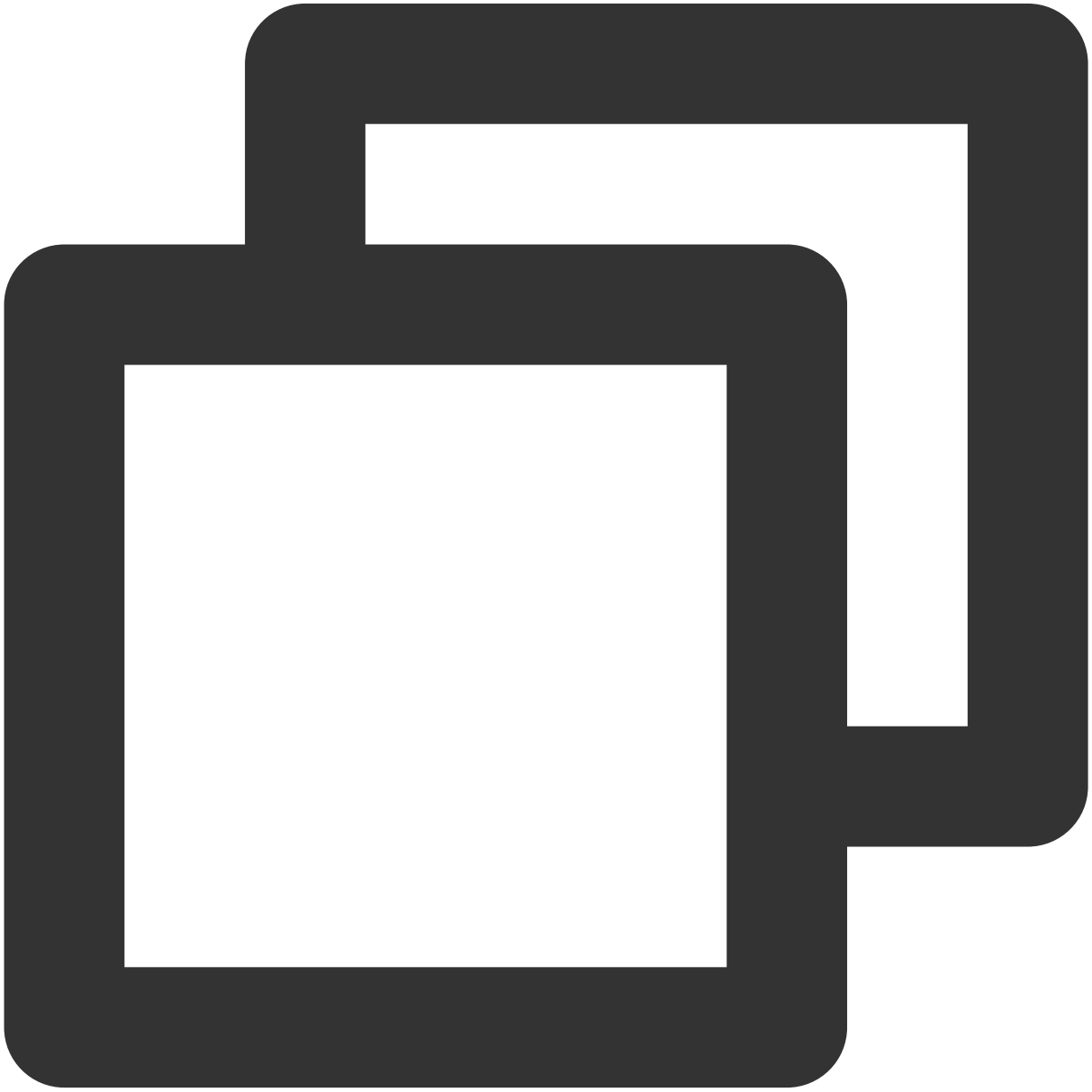
The API will return `Response` , which contains `RequestId` , as long as it processes the request, no matter whether the request is successful or not.

`RequestId` is the unique ID of an API request. It is required to troubleshoot issues.

Any fields other than the common fields are API-specific. For more information on such fields, please see the relevant API documentation. In this example, both `TotalCount` and `InstanceStatusSet` are specific to the `DescribeInstancesStatus` API. Since the user who initiated the request does not have a CVM instance yet, 0 is returned for `TotalCount` and `InstanceStatusSet` is empty.

Error response

If the call fails, you may see the following response:



```
{
```

```
    "Response": {
        "Error": {
            "Code": "AuthFailure.SignatureFailure",
            "Message": "The provided credentials could not be validated. Please che
        },
        "RequestId": "ed93f3cb-f35e-473f-b9f3-0d451b8b79c6"
    }
}
```

`Error` indicates that the request failed. A response for a failed request will always include the `Error`, `Code`, and `Message` fields.

`Code` indicates the specific error code, which is returned when an API request failed. You can use this code to locate the cause and solution of the error in the common or API-specific error code list.

`Message` explains the cause of the error. Note that the returned messages are subject to service updates. The information the messages provide may not be up-to-date and should not be the only source of reference.

`RequestId` is the unique ID of an API request. It is required to troubleshoot issues.

Common error codes

The `Error` field in a response indicates that the API call failed. The `Code` field in `Error` indicates the error code. The following table lists the common error codes that any services may return.

Error Code	Error Description
AuthFailure.InvalidSecretId	Invalid key (not TencentCloud API key type).
AuthFailure.MFAFailure	MFA failure.
AuthFailure.SecretIdNotFound	The key does not exist.
AuthFailure.SignatureExpire	The signature expired.
AuthFailure.SignatureFailure	Signature error.
AuthFailure.TokenFailure	Token error.
AuthFailure.UnauthorizedOperation	No CAM authorization.
DryRunOperation	DryRun Operation. It means that the request would have succeeded, but the DryRun parameter was used.
FailedOperation	The operation failed.
InternalError	Internal error.
InvalidAction	The API does not exist.

InvalidParameter	Incorrect parameter.
InvalidParameterValue	Invalid parameter value.
LimitExceeded	The quota limit is exceeded.
MissingParameter	A parameter is missing.
NoSuchVersion	The API version does not exist.
RequestLimitExceeded	The request rate limit is exceeded.
ResourceInUse	The resource is in use.
ResourceInsufficient	Insufficient resource.
ResourceNotFound	The resource does not exist.
ResourceUnavailable	The resource is unavailable.
UnauthorizedOperation	Unauthorized operation.
UnknownParameter	Unknown parameter error.
UnsupportedOperation	Unsupported operation.
UnsupportedProtocol	Unsupported HTTPS request method. Only GET and POST requests are supported.
UnsupportedRegion	Unsupported region.

API for .NET

Last updated：2023-03-07 18:16:40

TencentCloud API has been upgraded to v3.0. This version is optimized for performance and deployed in all regions. It supports nearby access and access by region for significantly reduced access latency. In addition, it features more detailed API descriptions and error codes and API-level comments for SDKs, enabling you to use Tencent Cloud services more conveniently and quickly. This document describes how to call APIs for .NET.
This version currently supports various Tencent Cloud services such as CVM, CBS, VPC, and TencentDB and will support more services in the future.

Request Structure

1. Service address (endpoint)

TencentCloud API supports access from either a nearby region (such as `cvm.tencentcloudapi.com` for CVM) or a specified region (such as `cvm.ap-guangzhou.tencentcloudapi.com` for CVM in the Guangzhou region). For values of the region parameter, please see the region list in the "Common Parameters" section below. To check whether a region is supported by a specific Tencent Cloud service, please see its "Request Structure" document.

Note:

For latency-sensitive businesses, we recommend you specify a domain name with a region.

2. Communications protocol

All TencentCloud APIs communicate over HTTPS, providing highly secure communications tunnels.

3. Request method

Supported HTTP request methods:
POST (recommended)
GET
`Content-Type` types supported by POST request:
application/json (recommended). The signature algorithm v3 (TC3-HMAC-SHA256) must be used.
application/x-www-form-urlencoded. The signature algorithm v1 (HmacSHA1 or HmacSHA256) must be used.
multipart/form-data (only supported by certain APIs). The signature algorithm v3 (TC3-HMAC-SHA256) must be used.
The size of a GET request packet cannot exceed 32 KB. The size of a POST request cannot exceed 1 MB for the signature algorithm v1 (HmacSHA1 or HmacSHA256) or 10 MB for the signature algorithm v3 (TC3-HMAC-SHA256).

4. Character encoding

UTF-8 encoding is always used.

Common Parameters

Note:
The common parameters are used to identity the user and API signature. They should be carried by each request to initiate properly.

Signature algorithm v3

The signature algorithm v3 (sometimes referred to as "TC3-HMAC-SHA256") is more secure than the signature algorithm v1 (referred to as signature algorithm in certain documents), supports larger request packets and POST JSON format, and has a higher performance. We recommend you use it to calculate signatures. For more information on how to use it, please see below.

Parameter Name	Type	Required	Description
X-TC-Action	String	**Yes**	Name of the API for the desired operation. For the specific value, please see the description of common parameter `Action` in the input parameters in the related API document. For example, the API for querying CVM instance list is `DescribeInstances` .
X-TC-Region	String	-	Region parameter, which is used to identify the region where the data you want to manipulate resides. For values supported for an API, please see the description of common parameter `Region` in the input parameters in related API documentation. **Note: this parameter is not required for some APIs (which will be indicated in related API documentation) and will not take effect even if it is passed.**
X-TC-Timestamp	Integer	**Yes**	The current UNIX timestamp that records the time when the API request was initiated, such as 1529223702. **Note: if the difference between the UNIX timestamp and the server time is greater than 5 minutes, a signature expiration error may occur.**
X-TC-Version	String	**Yes**	Version of the API for the desired operation, such as 2017-03-12 for CVM. For the specific value, please see the description of common parameter `Version` in the input parameters in related API documentation.
Authorization	String	**Yes**	HTTP authentication request header, such as TC3-HMAC-SHA256 Credential=AKIDEXAMPLE/Date/service/tc3_request,

			SignedHeaders=content-type;host, Signature=72e494ea8**********a96525168 Here, TC3-HMAC-SHA256: signature algorithm, currently fixed as this value. Credential: signature credential. `AKIDEXAMPLE` indicates the `SecretId`. `Date` indicates a UTC date which must match the value of `X-TC-Timestamp` (a common parameter) in UTC format. `service` indicates the name of the service and is generally a domain name prefix; for example, the domain name `cvm.tencentcloudapi.com` means the CVM service, and the value for this service is `cvm`. SignedHeaders: the headers that contain the authentication information. `content-type` and `host` are required. Signature: signature digest. For the calculation process, please see below.
X-TC-Token	String	No	Token used for temporary credentials. It must be used with a temporary key. You can get the temporary key and token by calling a CAM API. No token is required for a long-term key.

Signature algorithm v1

When the signature algorithm v1 (sometimes referred to as "HmacSHA256" or "HmacSHA1") is used, the common parameters should be uniformly placed in the request string.

Parameter Name	Type	Required	Description
Action	String	Yes	Name of the API for the desired operation. For the specific value, please see the description of common parameter `Action` in the input parameters in the related API document. For example, the API for querying CVM instance list is `DescribeInstances`.
Region	String	-	Region parameter, which is used to identify the region where the data you want to manipulate resides. For values supported for an API, please see the description of common parameter `Region` in the input parameters in related API documentation. **Note: this parameter is not required for some APIs (which will be indicated in related API documentation) and will not take effect even if it is passed.**
Timestamp	Integer	Yes	The current UNIX timestamp that records the time when the API request was initiated, such as 1529223702. If the

			difference between the UNIX timestamp and the current time is too large, a signature expiration error may occur.
Nonce	Integer	Yes	A random positive integer used in conjunction with `Timestamp` to prevent replay attacks.
SecretId	String	Yes	The identifying `SecretId` obtained on the TencentCloud API Key page. A `SecretId` corresponds to a unique `SecretKey` which is used to generate the request signature (`Signature`).
Signature	String	Yes	Request signature, which is used to verify the validity of the request. It is generated based on input parameters. For more information on how to calculate the signature, please see below.
Version	String	Yes	Version of the API for the desired operation, such as 2017-03-12 for CVM. For the specific value, please see the description of common parameter `Version` in the input parameters in related API documentation.
SignatureMethod	String	No	Signature algorithm. Currently, only HmacSHA256 and HmacSHA1 are supported. The HmacSHA256 algorithm is used to verify the signature only when this parameter is specified as HmacSHA256. In other cases, the signature is verified with HmacSHA1.
Token	String	No	Token used for temporary credentials. It must be used with a temporary key. You can get the temporary key and token by calling a CAM API. No token is required for a long-term key.

Region list

As the supported regions vary by service, please refer to the region list in each service's product documentation for specific details.
For example, you can see the region list of CVM.

API Call Method for .NET

TencentCloud API authenticates every request, that is, the request must be signed with the security credentials in the designated steps. Each request must contain the signature information in the common request parameters and be sent in the specified way and format.

Suppose your `SecretId` and `SecretKey` are `AKIDz8krbsJ5*********mLPx3EXAMPLE` and `Gu5t9xGAR**********EXAMPLE`, respectively. If you want to view the status of an unnamed instance in the Guangzhou region and have only one data entry returned, the request may be:

```
curl -X POST https://cvm.tencentcloudapi.com \\
-H "Authorization: TC3-HMAC-SHA256 Credential=AKIDz8krbsJ5*********mLPx3EXAMPLE/20
-H "Content-Type: application/json; charset=utf-8" \\
-H "Host: cvm.tencentcloudapi.com" \\
-H "X-TC-Action: DescribeInstances" \\
-H "X-TC-Timestamp: 1551113065" \\
-H "X-TC-Version: 2017-03-12" \\
```

```
-H "X-TC-Region: ap-guangzhou" \\
-d '{"Limit": 1, "Filters": [{"Values": ["\\u672a\\u547d\\u540d"], "Name": "instanc
```

Step 1. Apply for security credentials

In this document, the security credential used is a key pair, which consists of a `SecretId` and a `SecretKey`.

Each user can have up to two key pairs.

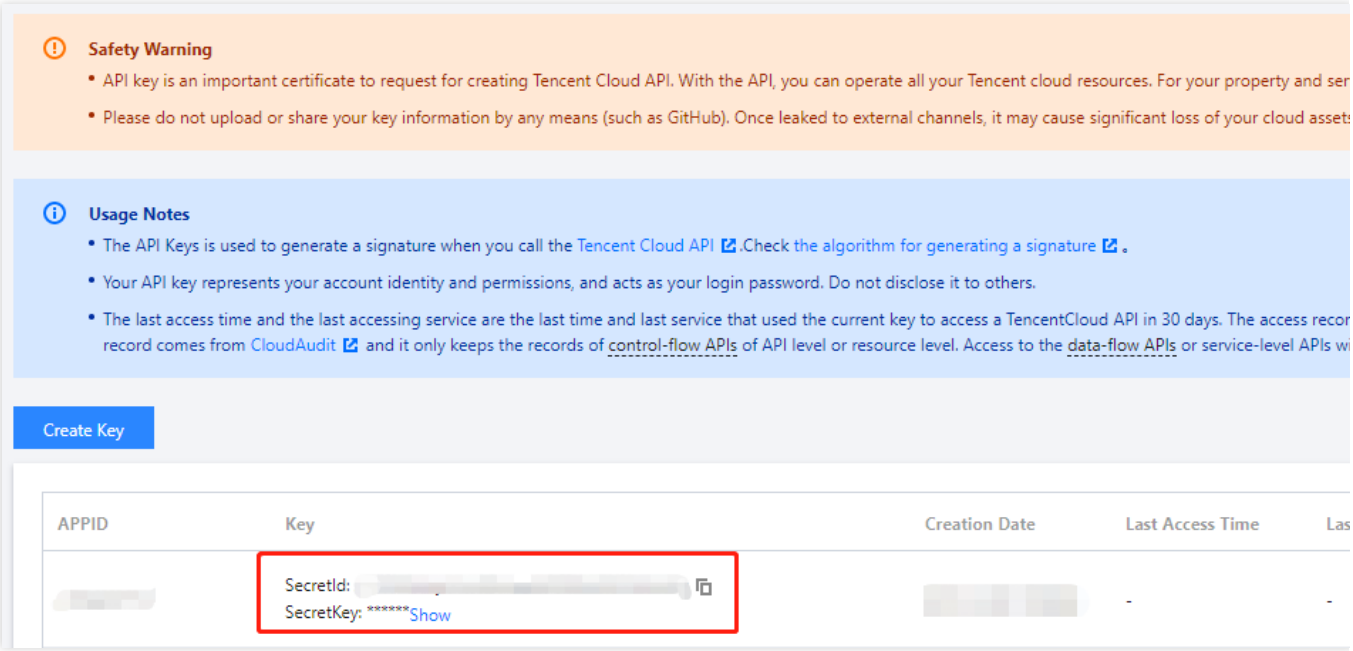
SecretId: identifies the user that calls an API, which is similar to a username.

SecretKey: authenticates the user that calls the API, which is similar to a password.

Note:

You must keep your security credentials private and avoid disclosure; otherwise, your assets may be compromised. If they are disclosed, please disable them as soon as possible.

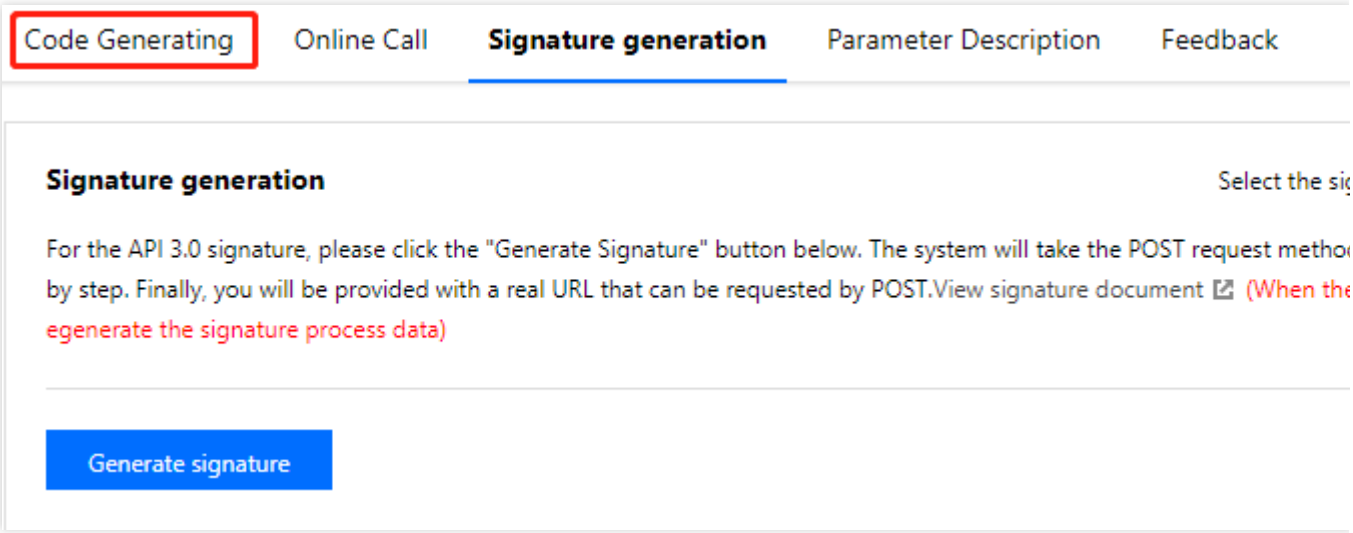
Go to the API key management page to get API keys as shown below:



Step 2

1. Get an API 3.0 signature v3

The signature algorithm v3 (TC3-HMAC-SHA256) is compatible with the previous signature algorithm v1 and more secure, supports larger request packets and POST JSON format, and has a higher performance. We recommend you use it to calculate signatures as shown below:



Note:

If you are using the signature algorithm for the first time, we recommend you use the "signature string generation" feature in API Explorer and select "API 3.0 signature v3" as the signature version, which can generate a signature for demonstration and verification. Plus, it can also generate SDK code directly. Seven common open-source programming language SDKs are available for TencentCloud API, including Python, Java, PHP, Go, Node.js, .NET, and C++.

TencentCloud API supports both GET and POST requests.

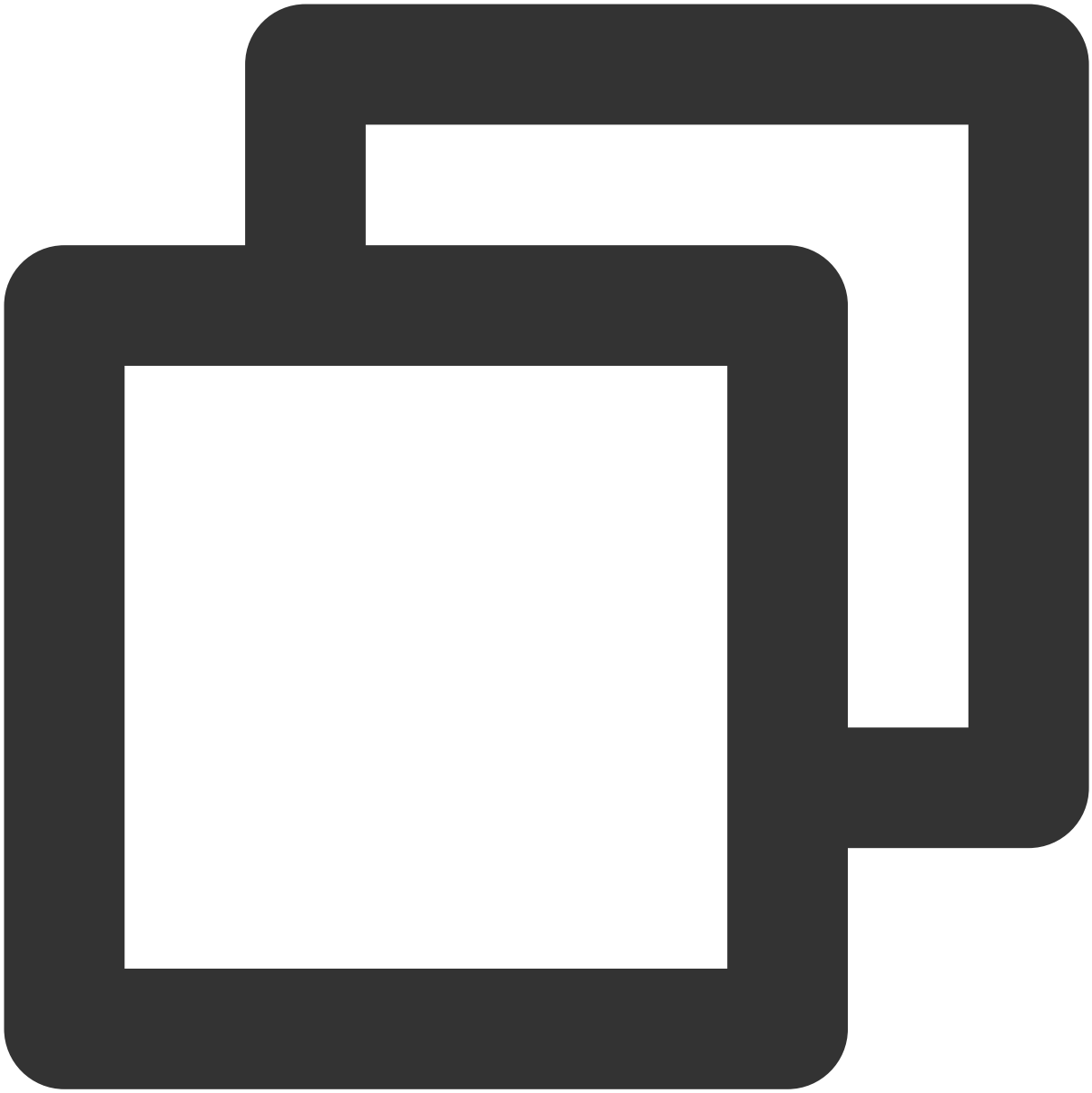
For the GET method, only the `Content-Type: application/x-www-form-urlencoded` protocol format is supported.

For the POST method, `Content-Type: application/json` and `Content-Type: multipart/form-data` are supported. The JSON format is supported by all business APIs, while the multipart format is supported only by specific APIs (in this case, an API cannot be called in JSON format). For more information, please see the specific business API document. We recommend you use the POST method because the two methods generate the same results, but the GET method only supports request packets below 32 KB in size.

The following describes how to calculate a signature by calling the DescribeInstances API. This API is chosen because:

1. The CVM API is enabled by default, and this API is often used.
2. It is read-only and does not change the status of existing resources.
3. It covers many types of parameters so that it is easy to show how to use an array that contains data structures.

1. Concatenate the canonical request string

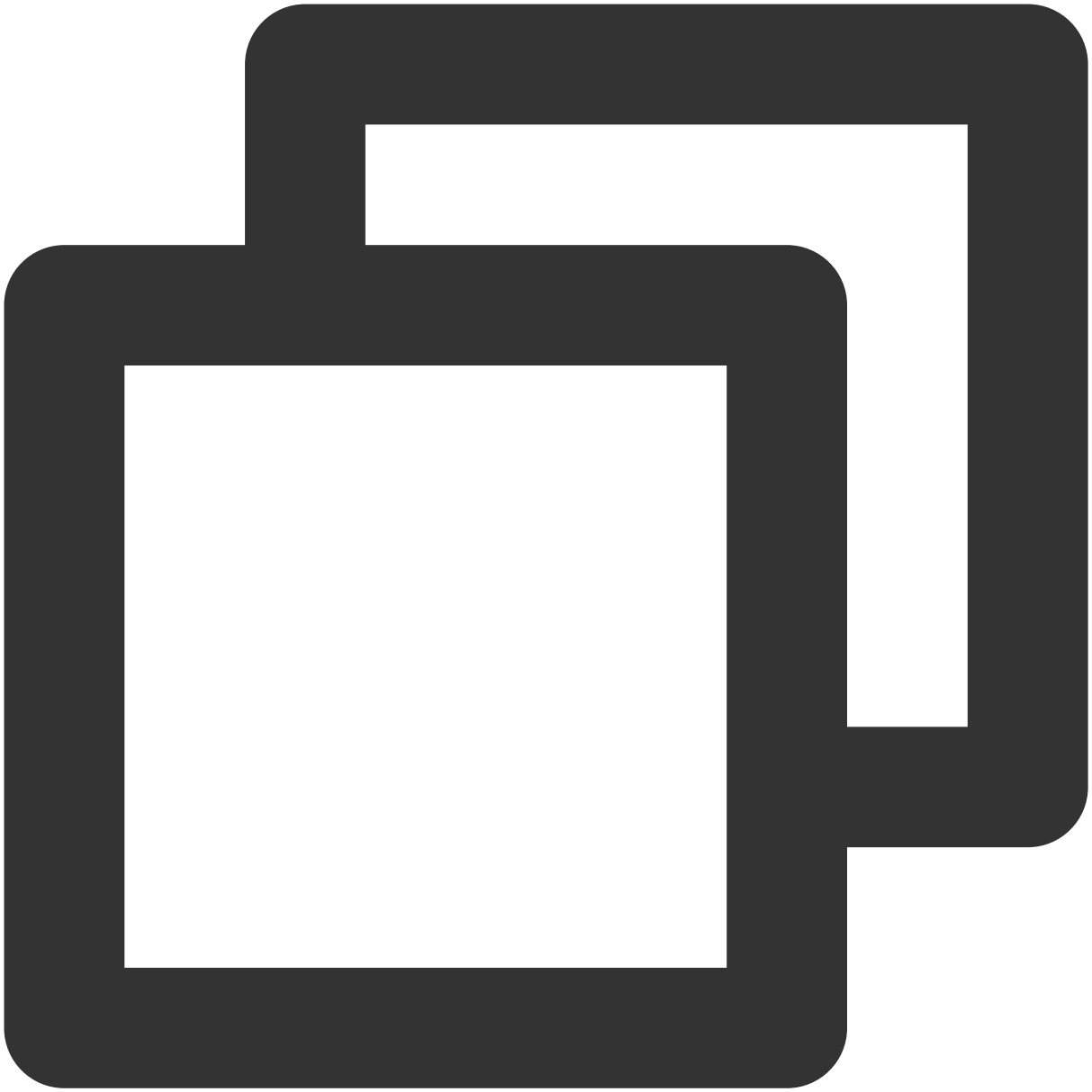


```
CanonicalRequest =
    HTTPRequestMethod + '\\n' +
    CanonicalURI + '\\n' +
    CanonicalQueryString + '\\n' +
    CanonicalHeaders + '\\n' +
    SignedHeaders + '\\n' +
    HashedRequestPayload
```

Field	Description

HTTPRequestMethod	HTTP request method (GET or POST). This example uses `POST` .
CanonicalURI	URI parameter. Slash ("/") is used for API 3.0.
CanonicalQueryString	Query string in the URL of the originating HTTP request. This is always an empty string POST requests and is the string after the question mark (?) for GET requests such as `Limit=10&Offset=0` . Note: `CanonicalQueryString` must be URL-encodec instructed in RFC 3986 with the UTF-8 character set. The applicable standard program language library is recommended. All special characters must be encoded and capitaliz
CanonicalHeaders	Header information for signature calculation, including at least `host` and `content type` . Custom headers can also be added to the signature process to improve the uniqueness and security of the request. Concatenation rules: both the key and value of header should be converted to lowercase with the leading and trailing spaces removed that they are concatenated in the `key:value\\n` format. If there are multiple heade they should be sorted in ASCII ascending order by header key (lowercase). The calcula result in this example is `content-type:application/json; charset=utf-8\\nhost:cvm.tencentcloudapi.com\\n` . Note: `content-type` must mat the content that is actually sent. In some programming languages, a `charset` value automatically added even if it is not specified. In this case, the request sent will be differ from the one signed, and the sever will return a signature verification failure.
SignedHeaders	Header information for signature calculation, indicating the request headers that are inv in the signature process. The request headers must correspond to the headers in `CanonicalHeaders` . `Content-type` and `host` are required headers. Concatenation rules: both the key and value of a header should be converted to lowerca there are multiple headers, they should be sorted in ASCII ascending order by header k (lowercase) and separated by semicolons (;). The value in this example is `content-type;host` .
HashedRequestPayload	Hash value of `Requestpayload` (i.e., the request body, such as `{"Limit": 1, "Filters": [{"Values": ["\\u672a\\u547d\\u540d"], "Name": "instance-name"}]}` in this example). The pseudo-code for calculation is `Lowercase(HexEncode(Hash.SHA256(RequestPayload)))` , which means t SHA256 hashing is performed on the payload of the HTTP request, then hexadecimal encoding is performed, and finally the encoded string is converted to lowercase letters. GET requests, `RequestPayload` is always an empty string. The calculation result i example is `35e9c5b0e3ae67532d3c9f17ead6c90222632e5b1ff7f6e89887f1398934f(`

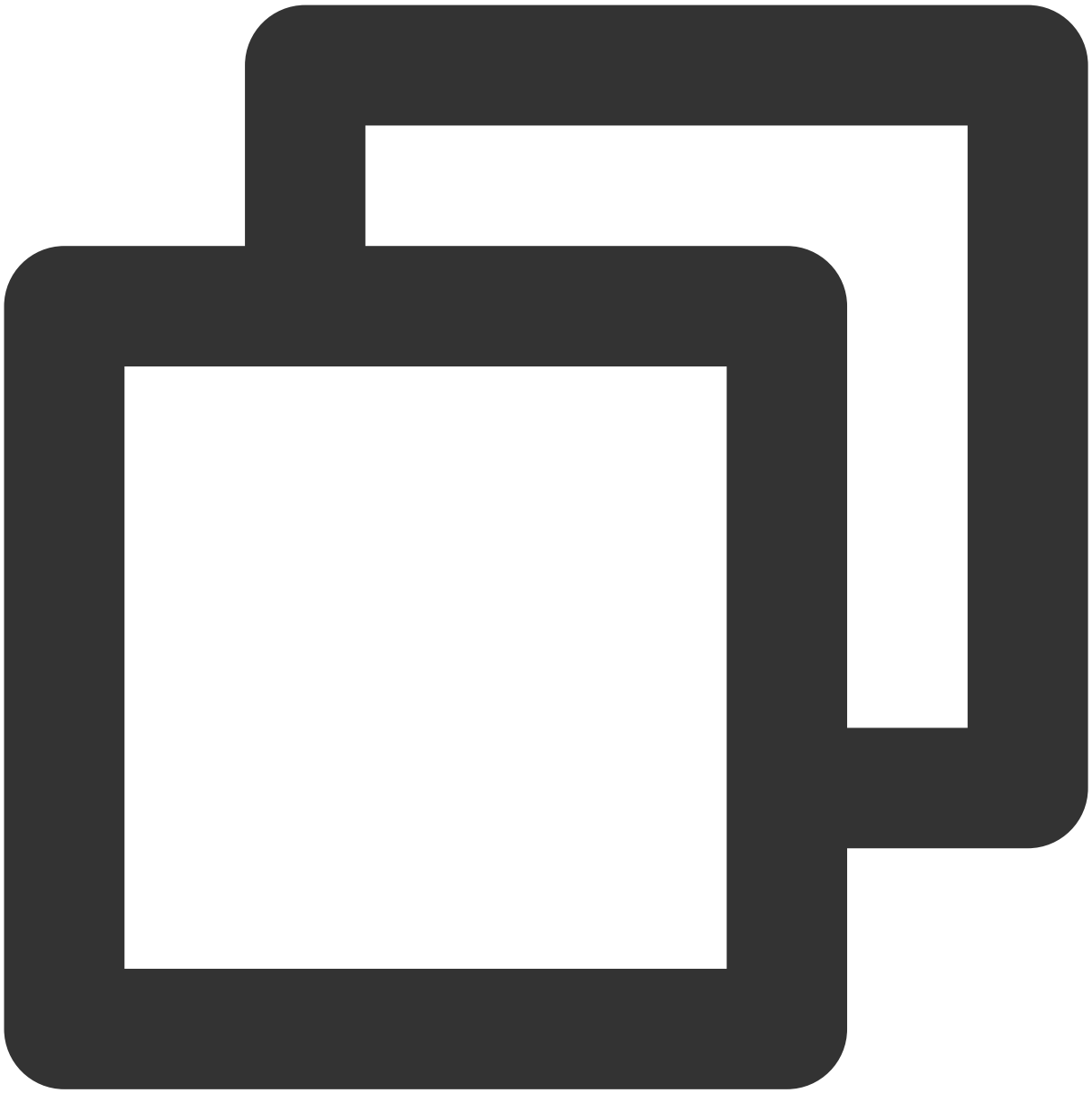
According to the rules above, the canonical request string obtained in the example is as follows:



```
POST
/

content-type:application/json; charset=utf-8
host:cvm.tencentcloudapi.com

content-type;host
35e9c5b0e3ae67532d3c9f17ead6c90222632e5b1ff7f6e89887f1398934f064
```

2. Concatenate the string to sign

Concatenate the string to sign in the following format:



```
StringToSign =
    Algorithm + \\n +
    RequestTimestamp + \\n +
    CredentialScope + \\n +
    HashedCanonicalRequest
```

Field	Description
Algorithm	Signature algorithm, which is always `TC3-HMAC-SHA256` currently.

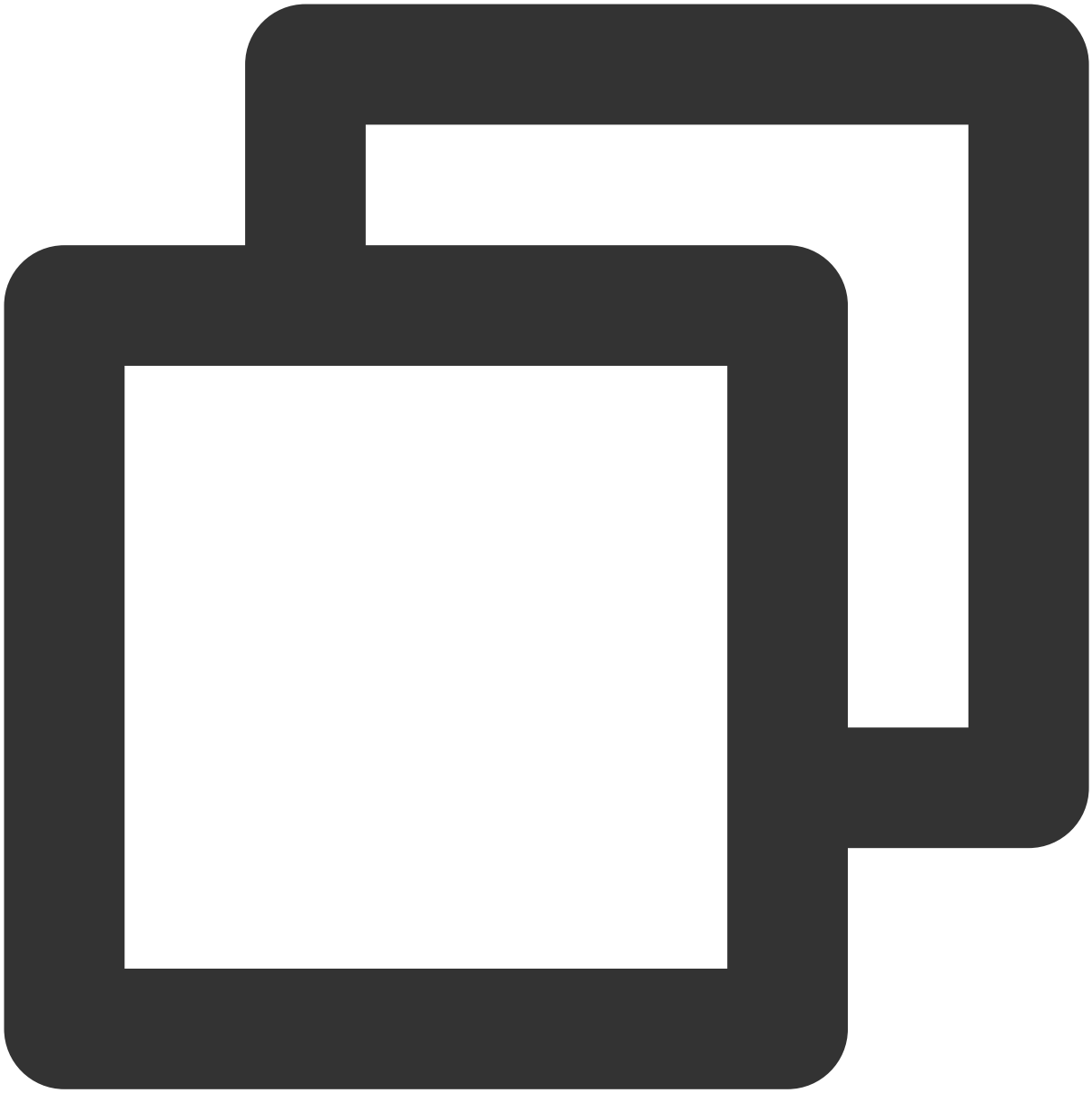
RequestTimestamp	Request timestamp, i.e., the value of the common parameter `X-TC-Timestamp` in request header. It is the UNIX timestamp of the current time in seconds, such as `1551113065` in this example.
CredentialScope	Scope of the credential in the format of `Date/service/tc3_request`, including date, requested service, and termination string (tc3_request). **`Date` indicates a U date, which should match the UTC date converted by the common parameter `TC-Timestamp`.** `service` is the service name, which should match the domain n of the service called. The calculation result in this example is `2019-02-25/cvm/tc3_request`.
HashedCanonicalRequest	Hash value of the canonical request string concatenated in the steps above. The pseu code for calculation is `Lowercase(HexEncode(Hash.SHA256(CanonicalRequest)))`. The calcu result in this example is `5ffe6a04c0664d6b969fab9a13bdab201d63ee709638e2749d62a09ca18d`

Note:

`Date` must be calculated from the timestamp `X-TC-Timestamp` and the time zone is UTC+0. If you add the local time zone information (such as UTC+8) in the system, calls can succeed both day and night but will definitely fail at 00:00. For example, if the timestamp is 1551113065 and the time in UTC+8 is 2019-02-26 00:44:25, the UTC+0 date in the calculated `Date` value should be 2019-02-25 instead of 2019-02-26.

`Timestamp` must be the same as your current system time, and your system time must be in sync with the UTC time. If the difference between the timestamp and your current system time is greater than five minutes, the request will fail. If your system time is out of sync with the UTC time for a prolonged period, the request will fail, and a signature expiration error will be returned.

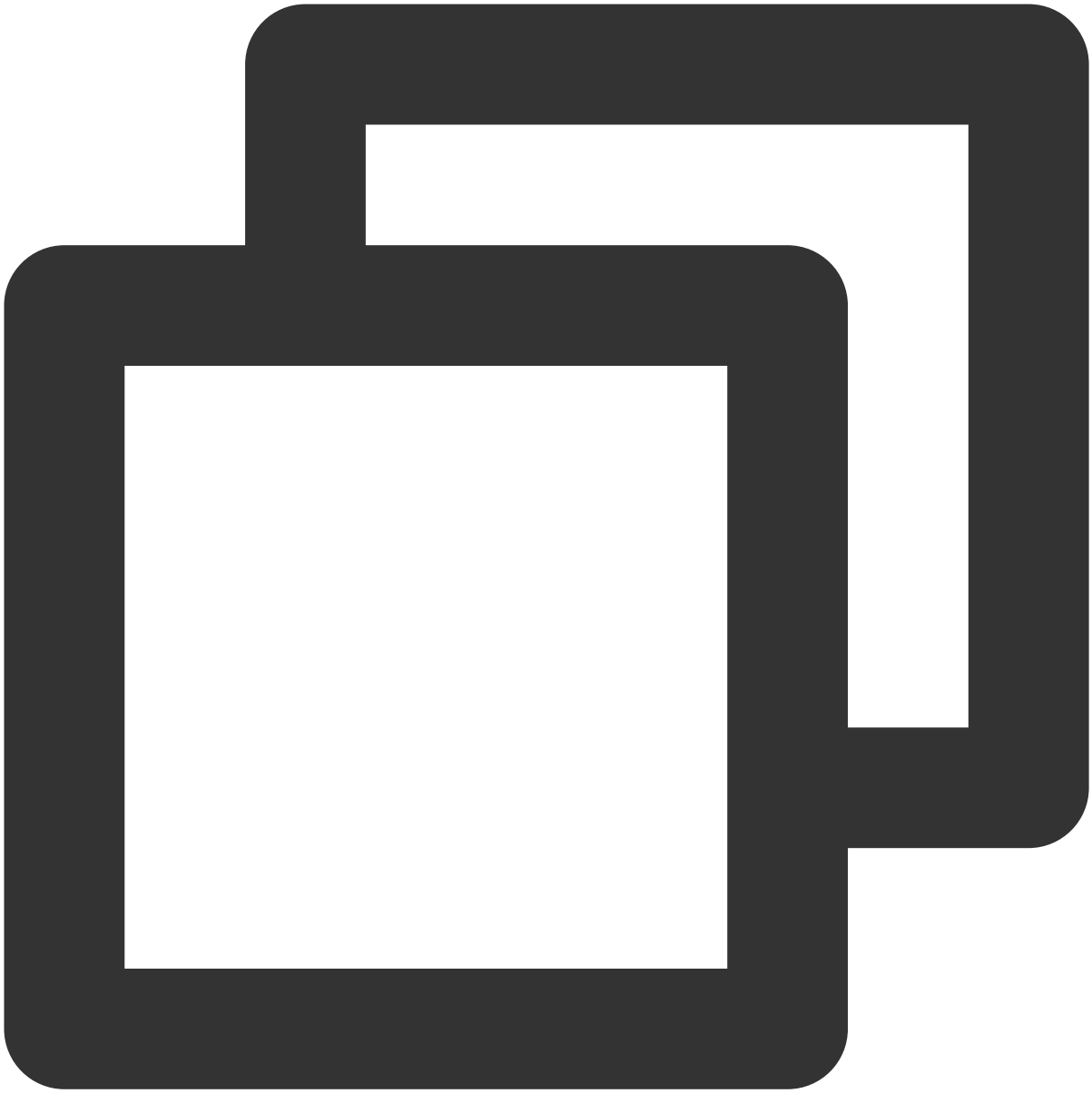
According to the rules above, the string to sign obtained in the example is as follows:



```
TC3-HMAC-SHA256
1551113065
2019-02-25/cvm/tc3_request
5ffe6a04c0664d6b969fab9a13bdab201d63ee709638e2749d62a09ca18d7031
```

3. Calculate the signature (pseudocode)

Please see the following sample code:



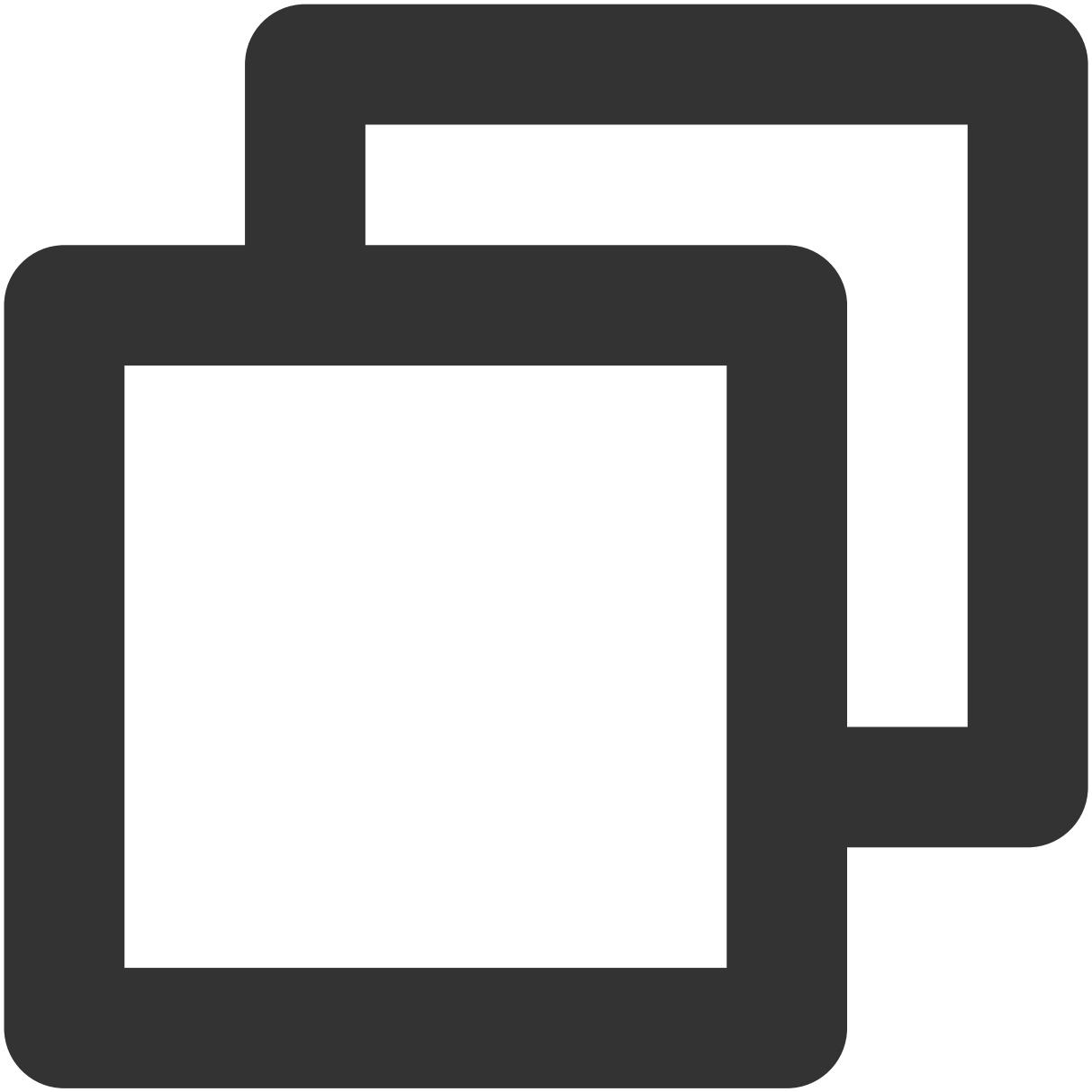
```
byte[] tc3SecretKey = Encoding.UTF8.GetBytes("TC3" + SECRET_KEY);
byte[] secretDate = HmacSHA256(tc3SecretKey, Encoding.UTF8.GetBytes(date));
byte[] secretService = HmacSHA256(secretDate, Encoding.UTF8.GetBytes(service));
byte[] secretSigning = HmacSHA256(secretService, Encoding.UTF8.GetBytes("tc3_reques
byte[] signatureBytes = HmacSHA256(secretSigning, Encoding.UTF8.GetBytes(stringToSi
string signature = BitConverter.ToString(signatureBytes).Replace("-", "").ToLower()
```

The calculation result in this example is

`72e494ea8*****************************************a96525168`.

4. Concatenate the Authorization string

Concatenate the `Authorization` string in the following format:

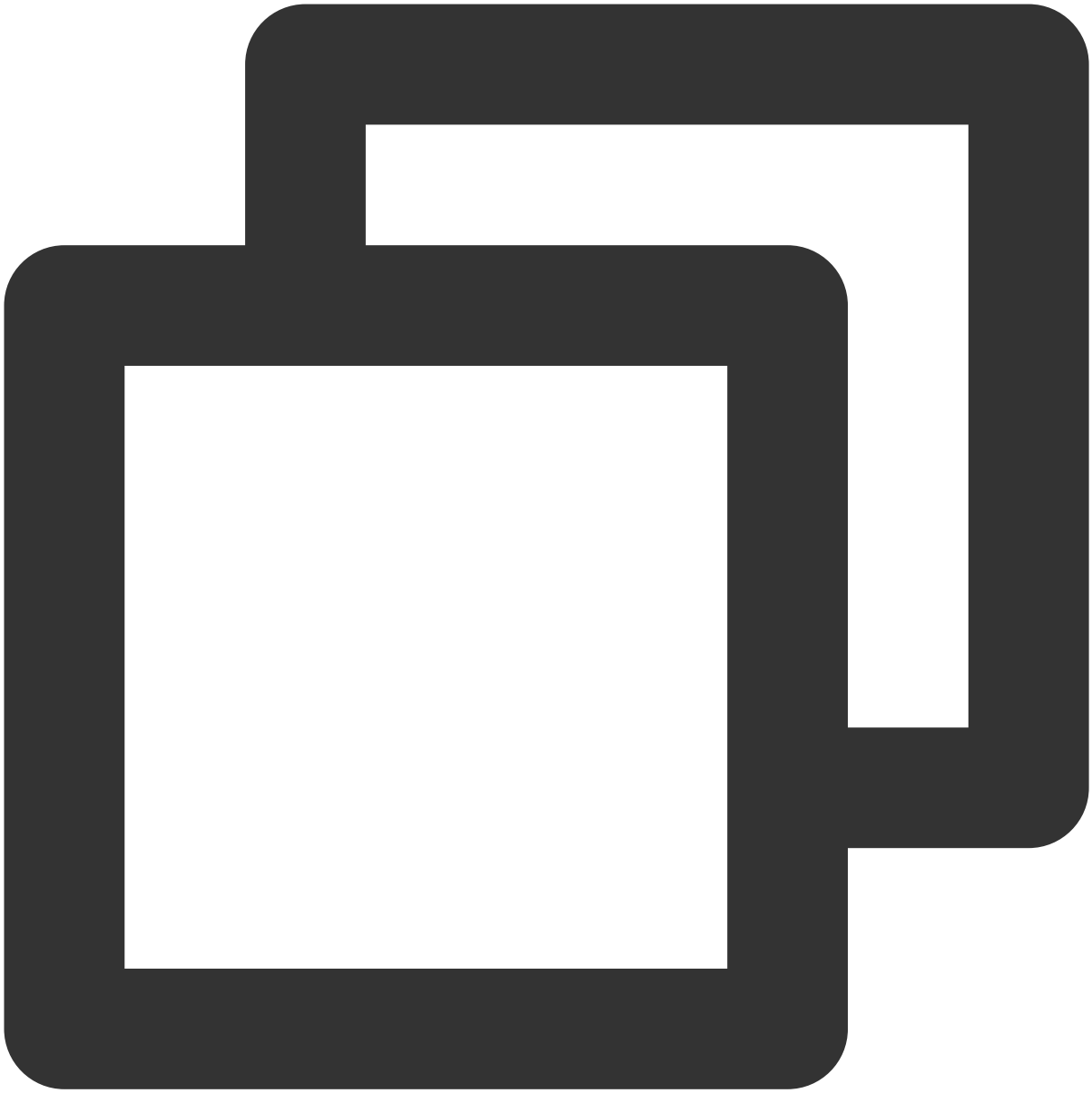


```
Authorization =
    Algorithm + ' ' +
    'Credential=' + SecretId + '/' + CredentialScope + ', ' +
    'SignedHeaders=' + SignedHeaders + ', ' +
    'Signature=' + Signature
```

Field	Description

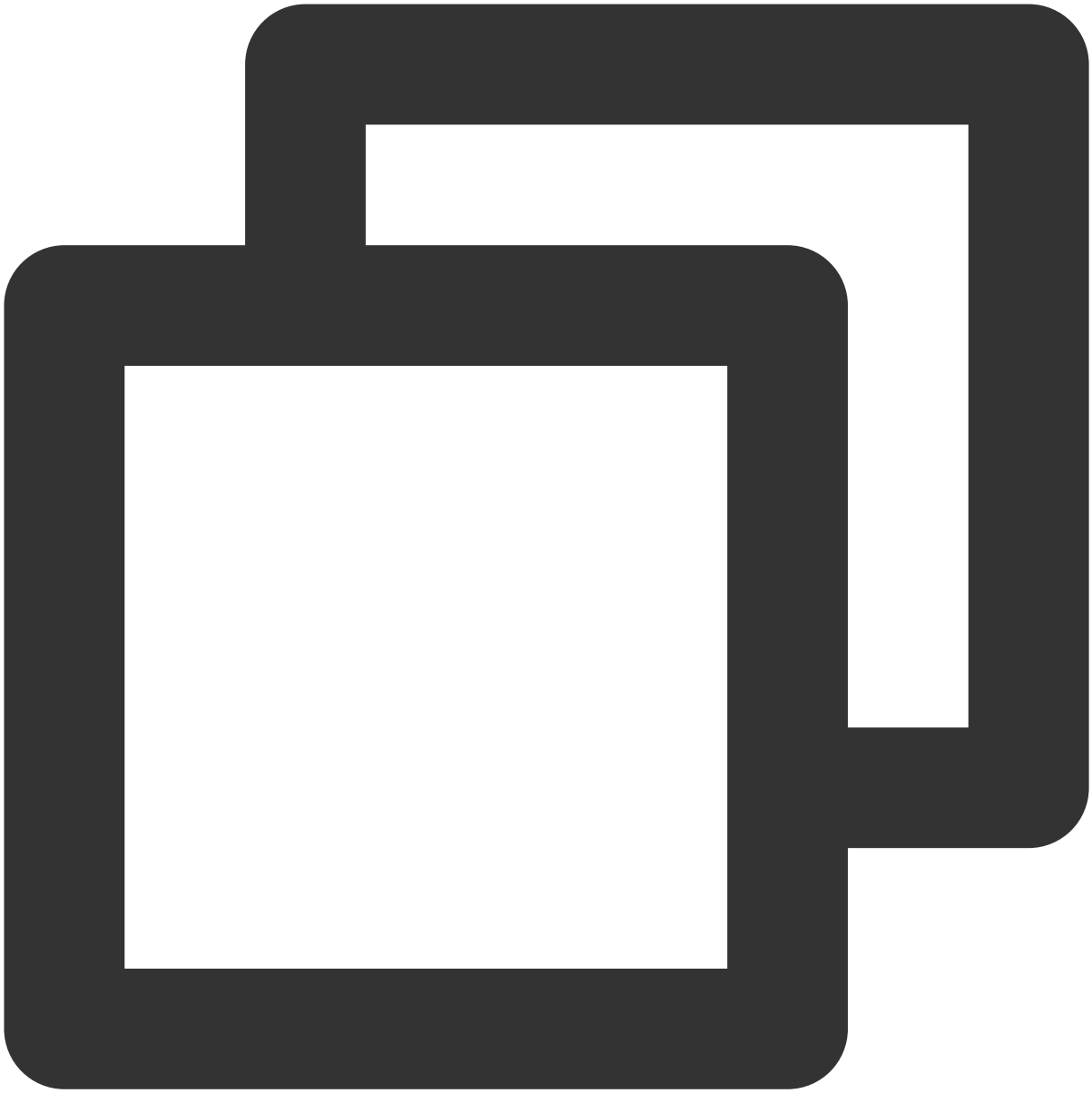
Algorithm	Signature algorithm, which is always `TC3-HMAC-SHA256` .
SecretId	`SecretId` in the key pair, i.e., `AKIDz8krbsJ5**********mLPx3EXAMPLE` .
CredentialScope	Credential scope (see above). The calculation result in this example is `2019-02-25/cvm/tc3_request` .
SignedHeaders	Header information for signature calculation (see above), such as `content-type;host` in this example.
Signature	Signature value. The calculation result in this example is `72e494ea8****************************************a96525168` .

According to the rules above, the values obtained in this example are:



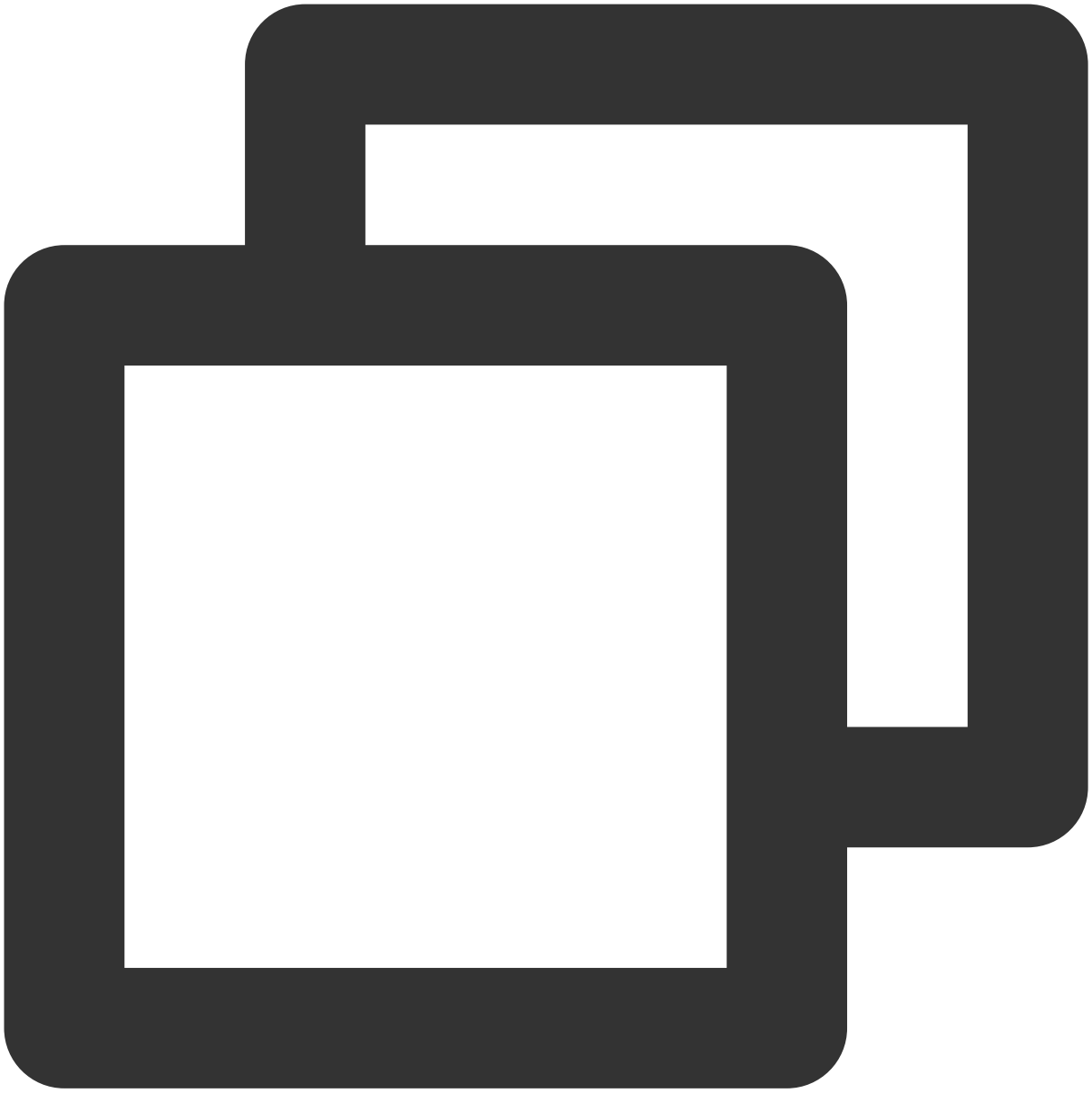
```
TC3-HMAC-SHA256 Credential=AKIDz8krbsJ5**********mLPx3EXAMPLE/2019-02-25/cvm/tc3_re
```

The complete call information is as follows:



```
POST https://cvm.tencentcloudapi.com/
Authorization: TC3-HMAC-SHA256 Credential=AKIDz8krbsJ5**********mLPx3EXAMPLE/2019-0
Content-Type: application/json; charset=utf-8
Host: cvm.tencentcloudapi.com
X-TC-Action: DescribeInstances
X-TC-Version: 2017-03-12
X-TC-Timestamp: 1551113065
X-TC-Region: ap-guangzhou

{"Limit": 1, "Filters": [{"Values": ["\\u672a\\u547d\\u540d"], "Name": "instance-na
```

5. Sample API 3.0 signature v3



```
using System;
using System.Collections.Generic;
using System.Security.Cryptography;
using System.Text;

public class Application {
    public static string SHA256Hex(string s)
    {
        using (SHA256 algo = SHA256.Create())
```

```
        {
            byte[] hashbytes = algo.ComputeHash(Encoding.UTF8.GetBytes(s));
            StringBuilder builder = new StringBuilder();
            for (int i = 0; i < hashbytes.Length; ++i)
            {
                builder.Append(hashbytes[i].ToString("x2"));
            }
            return builder.ToString();
        }
    }
    public static byte[] HmacSHA256(byte[] key, byte[] msg)
    {
        using (HMACSHA256 mac = new HMACSHA256(key))
        {
            return mac.ComputeHash(msg);
        }
    }
    public static void Main(string[] args)
    {

        // Key parameter
        string SECRET_ID = "AKIDz8krbsJ5**********mLPx3EXAMPLE";
        string SECRET_KEY = "Gu5t9xGAR***********EXAMPLE";


        string service = "cvm";
        string endpoint = "cvm.tencentcloudapi.com";
        string region = "ap-guangzhou";
        string action = "DescribeInstances";
        string version = "2017-03-12";
        string algorithm = "TC3-HMAC-SHA256";
        string contentType = "application/json";
        double RequestTimestamp = 1551113065;  // Timestamp: 2019-02-26 00:44:25. T
        // long timestamp = ToTimestamp() / 1000;
        // string requestTimestamp = timestamp.ToString();
        string date = new DateTime(1970, 1, 1, 0, 0, 0, 0, DateTimeKind.Utc).AddSec
        // Make sure that the time zone is correct

        // ************* Step 1. Concatenate the canonical request string *********
        string httpRequestMethod = "POST";
        string canonicalUri = "/";
        string canonicalQueryString = "";
        string canonicalHeaders = "content-type:" + contentType +"; charset=utf-8\\
        string signedHeaders = "content-type;host";
        string requestPayload = "{\\"Limit\\": 1, \\"Filters\\": [{\\"Values\\": [\
        string hashedRequestPayload = SHA256Hex(requestPayload);
        string canonicalRequest = httpRequestMethod + "\\n"
```

```
            + canonicalUri + "\\n"
            + canonicalQueryString + "\\n"
            + canonicalHeaders + "\\n"
            + signedHeaders + "\\n"
            + hashedRequestPayload;
        Console.WriteLine(canonicalRequest);
        Console.WriteLine("-----------------------------------");

        // ************* Step 2. Concatenate the string to sign *************
        string credentialScope = date + "/" + service + "/" + "tc3_request";
        string hashedCanonicalRequest = SHA256Hex(canonicalRequest);
        string stringToSign = algorithm + "\\n" + RequestTimestamp + "\\n" + creden
        Console.WriteLine(stringToSign);
        Console.WriteLine("-----------------------------------");

        // ************* Step 3. Calculate the signature *************
        byte[] tc3SecretKey = Encoding.UTF8.GetBytes("TC3" + SECRET_KEY);
        byte[] secretDate = HmacSHA256(tc3SecretKey, Encoding.UTF8.GetBytes(date));
        byte[] secretService = HmacSHA256(secretDate, Encoding.UTF8.GetBytes(servic
        byte[] secretSigning = HmacSHA256(secretService, Encoding.UTF8.GetBytes("tc
        byte[] signatureBytes = HmacSHA256(secretSigning, Encoding.UTF8.GetBytes(st
        string signature = BitConverter.ToString(signatureBytes).Replace("-", "").T
        Console.WriteLine(signature);
        Console.WriteLine("-----------------------------------");

        // ************* Step 4. Concatenate the `Authorization` string ***********
        string authorization = algorithm + " "
            + "Credential=" + SECRET_ID + "/" + credentialScope + ", "
            + "SignedHeaders=" + signedHeaders + ", "
            + "Signature=" + signature;
        Console.WriteLine(authorization);
        Console.WriteLine("-----------------------------------");

        Dictionary<string, string> headers = new Dictionary<string, string>();
        headers.Add("Authorization", authorization);
        headers.Add("Host", endpoint);
        headers.Add("Content-Type", contentType + "; charset=utf-8");
        headers.Add("X-TC-Timestamp", RequestTimestamp.ToString());
        headers.Add("X-TC-Version", version);
        headers.Add("X-TC-Action", action);
        headers.Add("X-TC-Region", region);
        Console.WriteLine("POST https://cvm.tencentcloudapi.com");
        foreach (KeyValuePair<string, string> kv in headers)
        {
            Console.WriteLine(kv.Key + ": " + kv.Value);
        }
        Console.WriteLine();
```

```
            Console.WriteLine(requestPayload);
        }
    }
```

2. Get an API 3.0 signature v1

The signature algorithm v1 (HmacSHA1 or HmacSHA256) is simple and easy to use, but its functionality and security are not as good as the signature algorithm v3 which is therefore recommended.
If you are using the signature algorithm for the first time, we recommend you use the "signature string generation" feature in API Explorer and select "API 3.0 signature v1" as the signature version, which can generate a signature for demonstration and verification and provides signing examples for certain programming languages. Plus, it can also generate SDK code directly. Seven common open-source programming language SDKs are available for TencentCloud API, including Python, Java, PHP, Go, Node.js, .NET, and C++.
For example, if you call the `DescribeInstances` API to query CVM instances, the request parameters may be as follows:

Parameter Name	Description	Value
Action	Method	DescribeInstances
SecretId	Key ID	`AKIDz8krbsJ5*********mLPx3EXAMPLE`
Timestamp	Current timestamp	1465185768
Nonce	Random positive integer	11886
Region	Instance region	ap-guangzhou
InstanceIds.0	ID of the instance to be queried	ins-09dx96dg
Offset	Offset	0
Limit	Allowed maximum number of output entries	20
Version	API version number	2017-03-12

1. Sort parameters

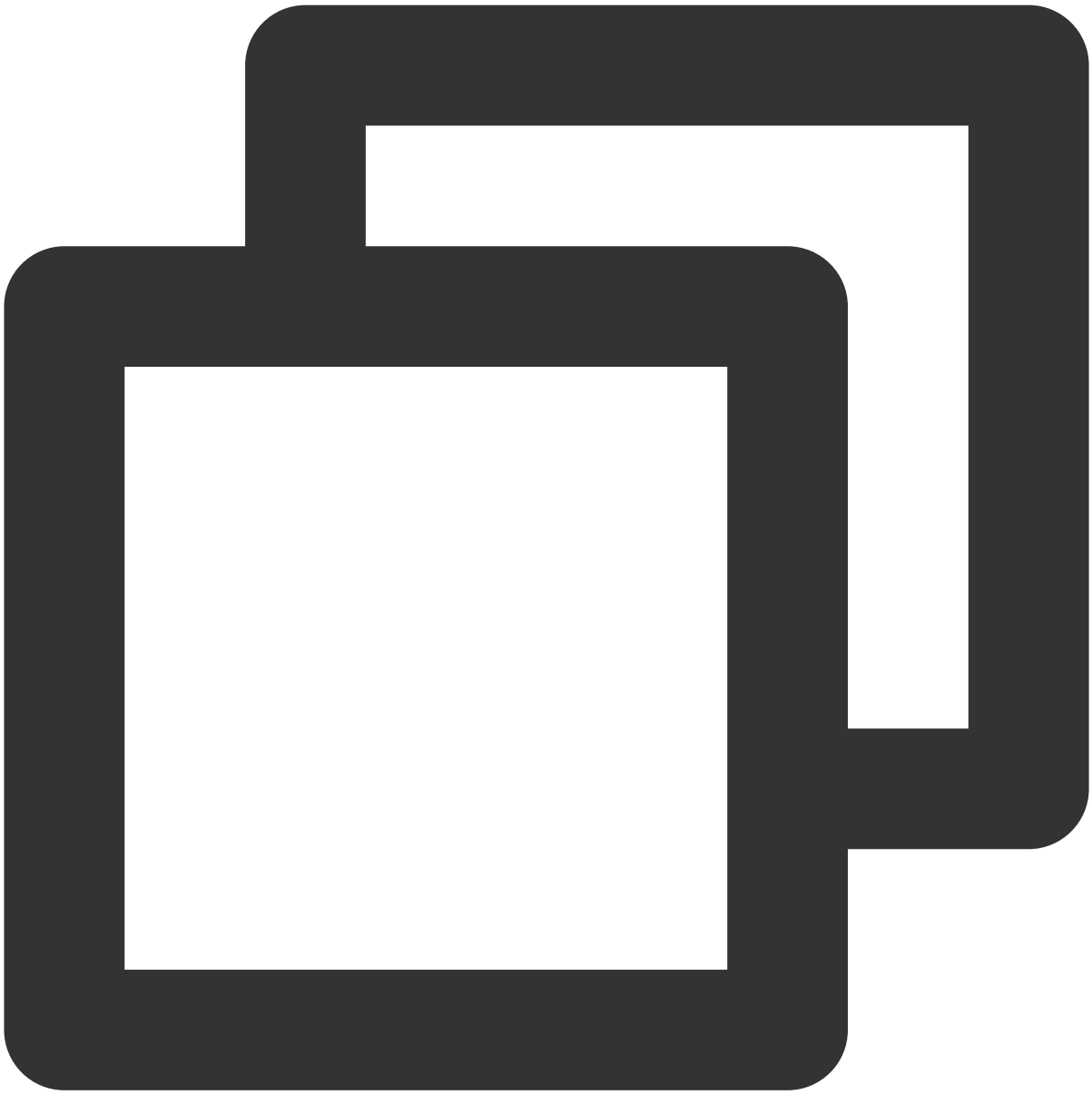
Sort all the request parameters in an ascending lexicographical order (ASCII code) by their names.

Note:

1. The parameters are sorted only by name but not by value.
2. The parameters are sorted based on ASCII code but not in an alphabetical order or by value. For example, InstanceIds.2 should be arranged behind InstanceIds.12. You can complete sorting by using a sorting function in a

programming language, such as the ksort function in PHP.

The parameters in the example are sorted as follows:



```
{
    'Action' : 'DescribeInstances',
    'InstanceIds.0' : 'ins-09dx96dg',
    'Limit' : 20,
    'Nonce' : 11886,
    'Offset' : 0,
    'Region' : 'ap-guangzhou',
    'SecretId' : 'AKIDz8krbsJ5**********mLPx3EXAMPLE',
```

```
    'Timestamp' : 1465185768,
    'Version': '2017-03-12',
}
```

Any other programming languages can be used to sort these parameters as long as the same result is produced.

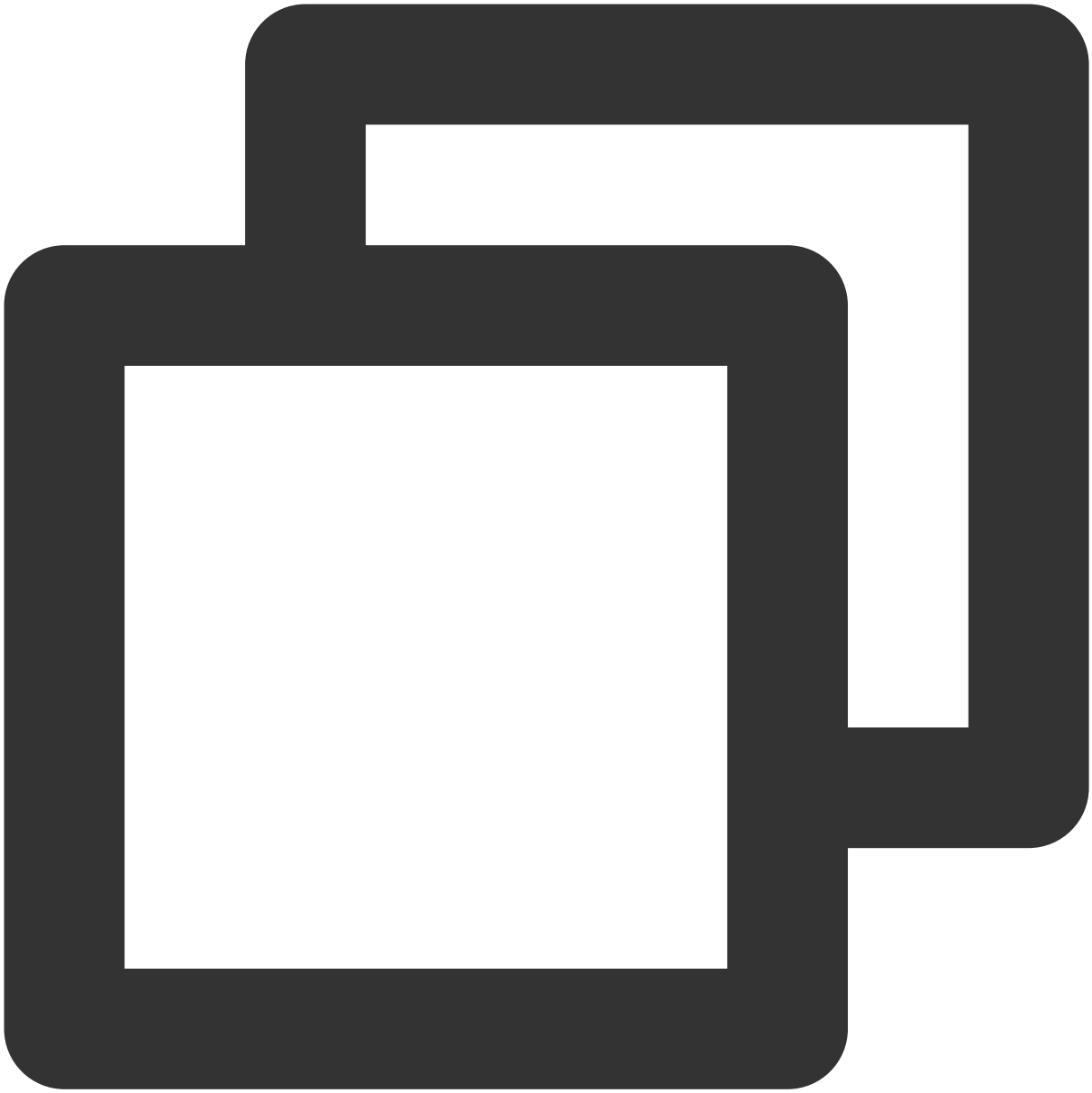
2. Concatenate the canonical request string

This step generates a request string. Format the request parameters sorted in the previous step into the form of `parameter=value`. For example, for the `Action` parameter, its parameter is `Action` and its value is `DescribeInstances`; therefore, the parameter will be formatted into `Action=DescribeInstances`.

Note:

The `value` is the original value instead of the URL-encoded value.

Then, concatenate the formatted parameters with `&`. The generated request string will be as follows:



```
Action=DescribeInstances&InstanceIds.0=ins-09dx96dg&Limit=20&Nonce=11886&Offset=0&R
```

3. Concatenate the string to sign

This step generates the original signature string. The original signature string consists of the following parameters:

1. Request method: POST and GET methods are supported. GET is used here for the request. Please note that the method name should be in all capital letters.

2. Request server: the domain name of the request for querying instances (DescribeInstances) is `cvm.tencentcloudapi.com` . The actual request domain name varies by the module to which the API belongs.

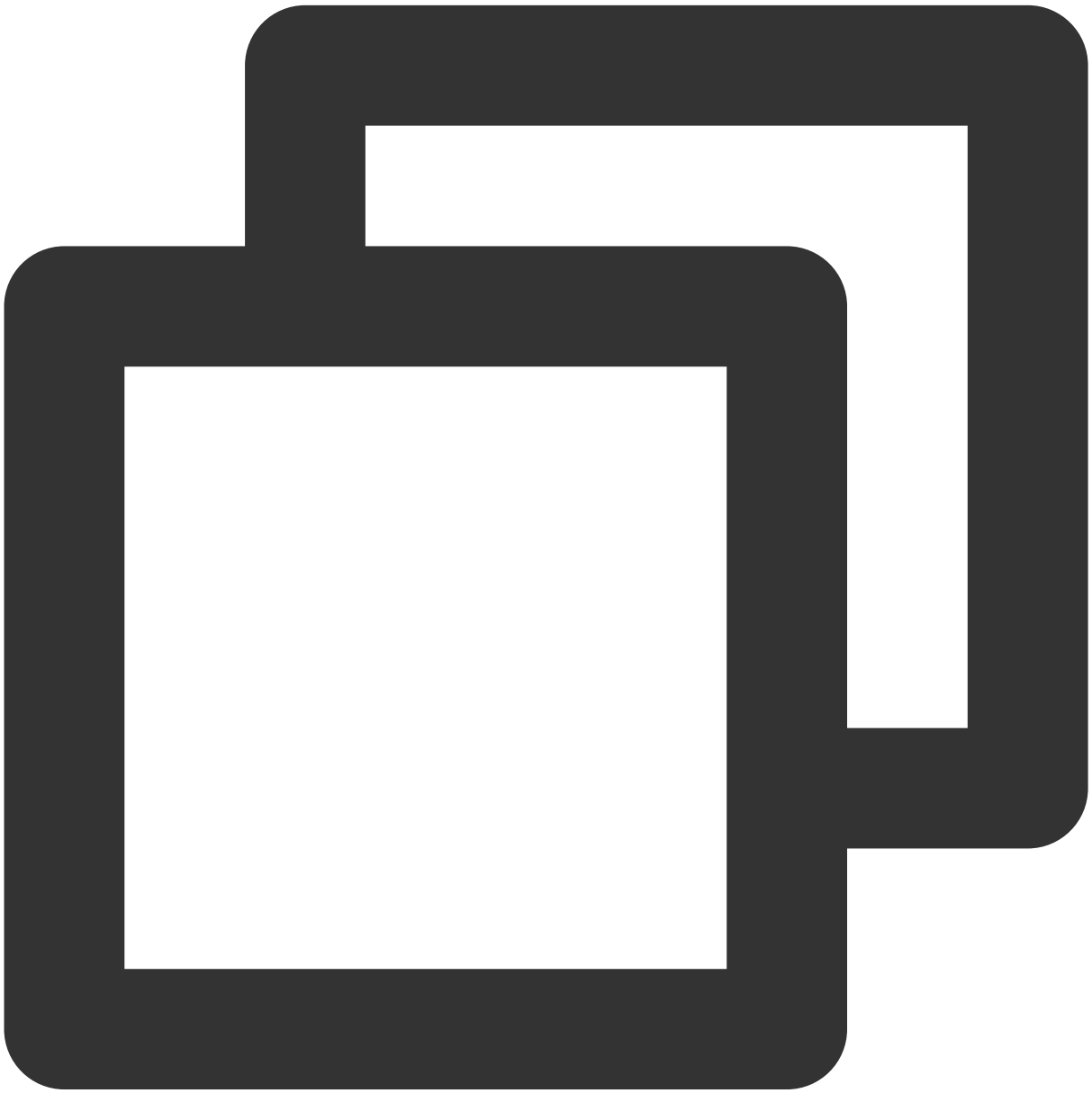
For more information, please see the specific API document.

3. Request path: the request path in the current version of TencentCloud API is fixed to `/` .

4. Request string: the request string generated in the previous step.

The rule for concatenating the original string of the signature is `request method + request server + request path + ? + request string` .

The concatenation result in the example is as follows:



```
GETcvm.tencentcloudapi.com/?Action=DescribeInstances&InstanceIds.0=ins-09dx96dg&Lim
```

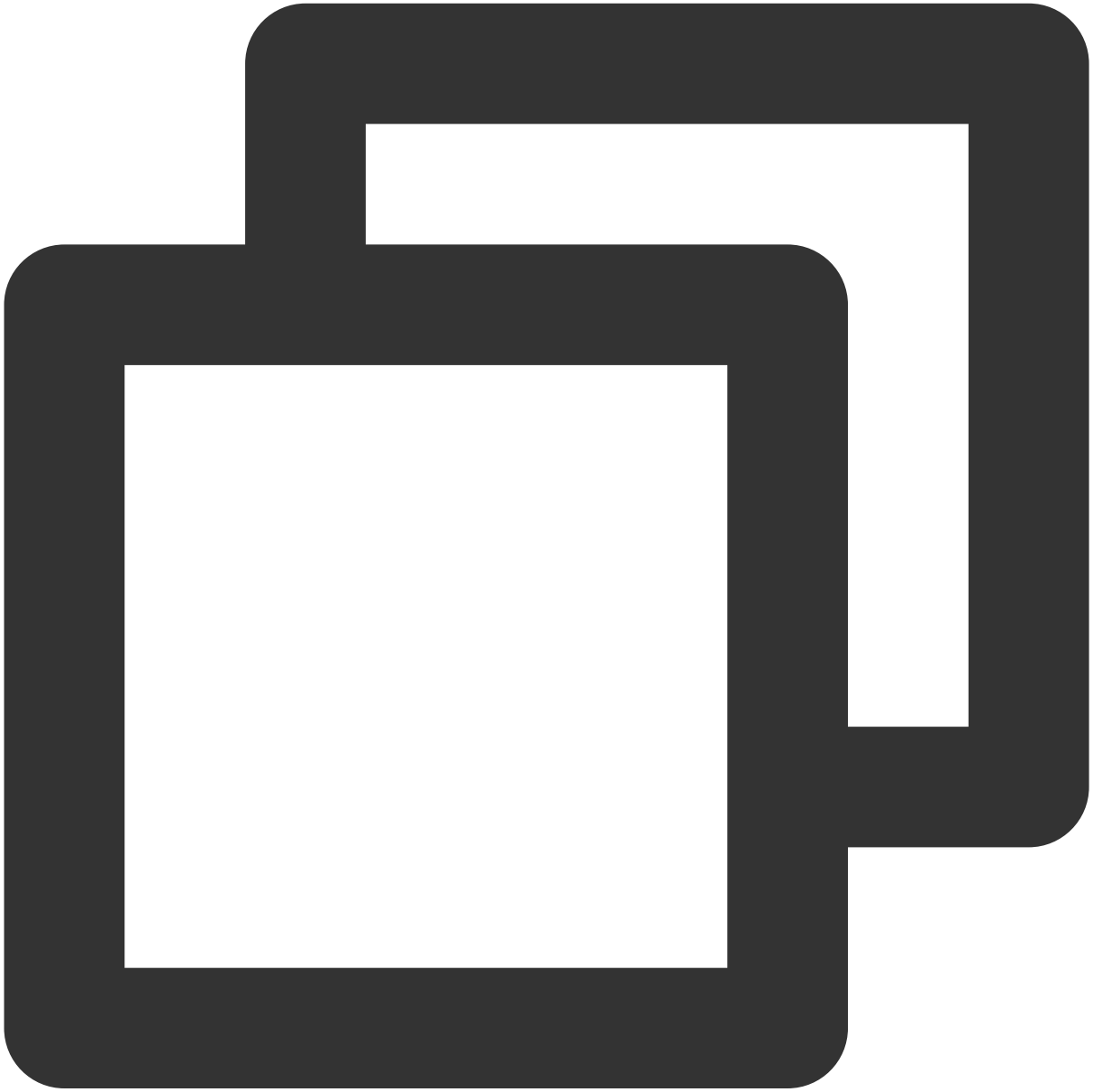
4. Calculate the signature (pseudocode)

This step generates a signature string. Use the HMAC-SHA1 algorithm to sign the **original signature string** obtained in the previous step, and then Base64-encode the generated signature to get the final signature.

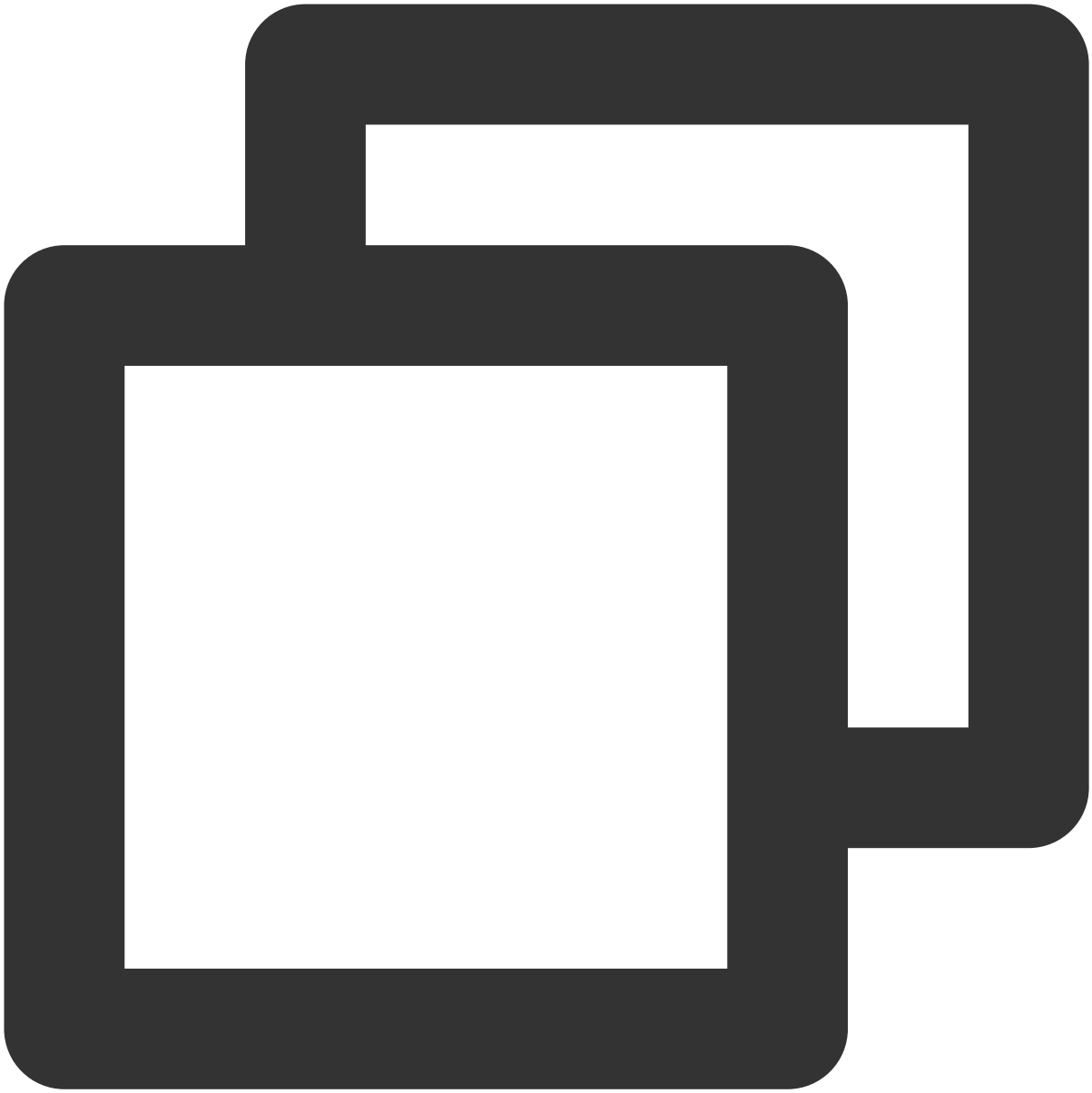
```
public static string Sign(string signKey, string secret, string SignatureMethod)
    {
        string signRet = string.Empty;
        using (HMACSHA1 mac = new HMACSHA1(Encoding.UTF8.GetBytes(signKey)))
        {
            byte[] hash = mac.ComputeHash(Encoding.UTF8.GetBytes(secret));
            signRet = Convert.ToBase64String(hash);
```

```
        }
        return signRet;
    }
```

5. Get the call information and send a request



```
# The API will be called actually, and fees will be incurred if it is a consumption
resp = requests.get("https://" + endpoint, params=data)
print(resp.url)
```



```
The obtained request string is as follows:
https://cvm.tencentcloudapi.com/?Action=DescribeInstances&InstanceIds.0=ins-09dx96d
```

Field	Description
endpoint	Service address, such as `cvm.tencentcloudapi.com` .
data	API parameter of the sample API 3.0 signature v1. **Note:** you should add the calculated signature in the format of key-value pair to `data` .

Note:

The key in the example is not real, and the timestamp is not the current system time. If you open this URL in the browser or call it by using commands such as `curl` , an authentication error `The signature expired` will be returned. To obtain a URL that works, you need to replace the `SecretId` and `SecretKey` in this example with your own credentials and use the current system time as the `Timestamp` .

To further explain the signing process, .NET is used as examples below to implement the process as described above. The request domain name, API, and parameter values in the above example are used here. The code below is for demonstration only. Please use the SDK for actual development.

6. Encode a signature string

The generated signature string cannot be directly used as a request parameter and needs to be URL-encoded. For example, if the signature string generated in the previous step is `Eli****************cGeI=` , the final value of the `Signature` request parameter will be `EliP**********************eI%3D` , which will be used to generate the final request URL.

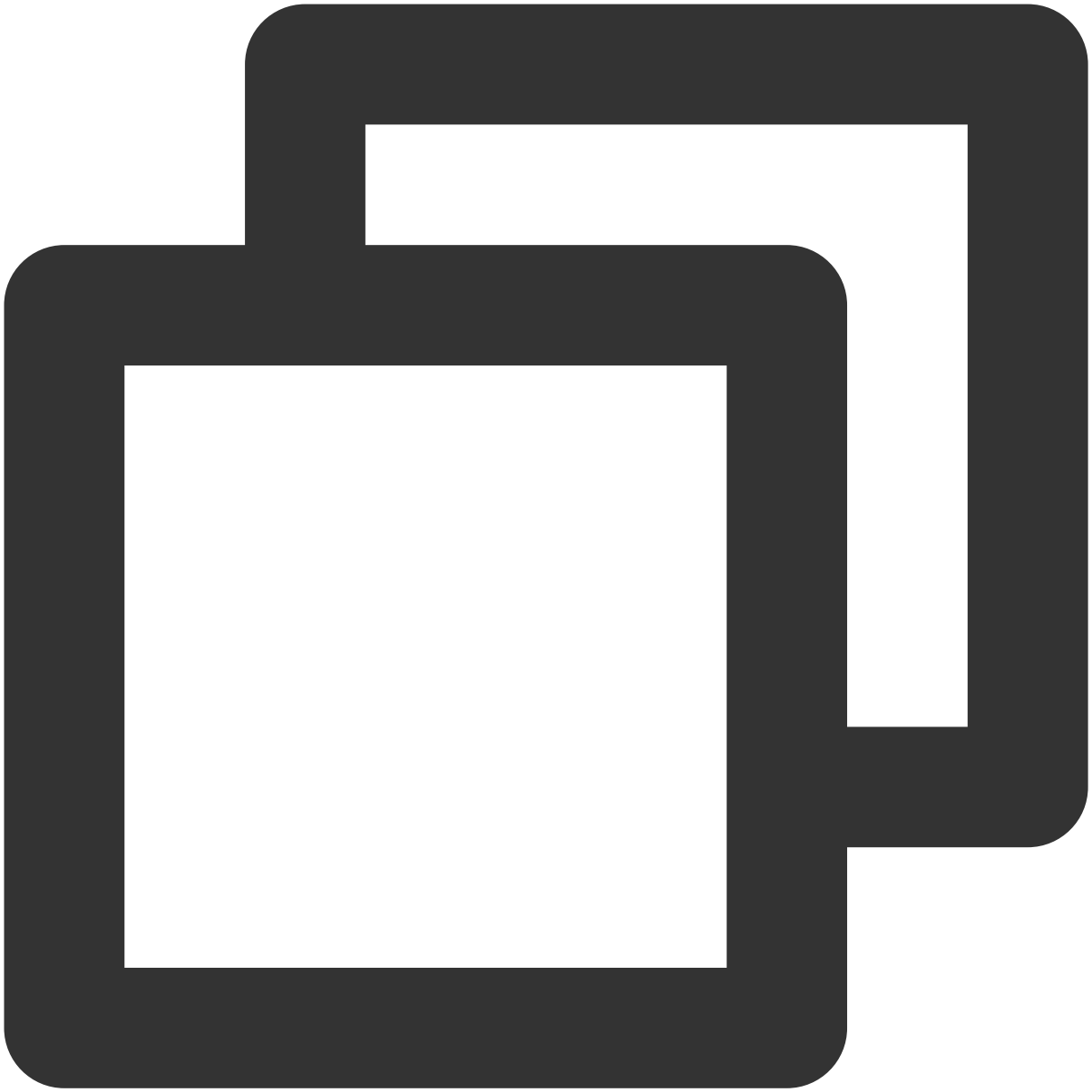
Note:

If you use the GET request method or use the POST request method with `Content-Type` of `application/x-www-form-urlencoded` , all the request parameter values must be URL-encoded (except the parameter key and the equal symbol (=)) before the request is sent. Non-ASCII characters must be encoded with UTF-8 before URL-encoding.

The network libraries of some programming languages automatically URL-encode all parameters. In this case, the signature string does not need to be URL-encoded again; otherwise, two rounds of URL-encoding will cause the signature to fail.

Other parameter values also need to be encoded with RFC 3986. Use %XY in percent-encoding for special characters such as Chinese characters, where "X" and "Y" are hexadecimal characters (0–9 and uppercase A–F). Using lowercase characters will cause an error.

7. Sample API 3.0 signature v1



```
using System;
using System.Collections.Generic;
using System.Net;
using System.Security.Cryptography;
using System.Text;

public class Application {
    public static string Sign(string signKey, string secret)
    {
        string signRet = string.Empty;
            using (HMACSHA1 mac = new HMACSHA1(Encoding.UTF8.GetBytes(signKey)))
```

```
            {
                byte[] hash = mac.ComputeHash(Encoding.UTF8.GetBytes(secret));
                signRet = Convert.ToBase64String(hash);
            }
        return signRet;
    }
    public static string MakeSignPlainText(SortedDictionary<string, string> request
    {
        string retStr = "";
        retStr += requestMethod;
        retStr += requestHost;
        retStr += requestPath;
        retStr += "?";
        string v = "";
        foreach (string key in requestParams.Keys)
        {
            v += string.Format("{0}={1}&", key, requestParams[key]);
        }
        retStr += v.TrimEnd('&');
        return retStr;
    }

    public static void Main(string[] args)
    {
        // Key parameter
        string SECRET_ID = "AKIDz8krbsJ5**********mLPx3EXAMPLE";
        string SECRET_KEY = "Gu5t9xGAR***********EXAMPLE";

        string endpoint = "cvm.tencentcloudapi.com";
        string region = "ap-guangzhou";
        string action = "DescribeInstances";
        string version = "2017-03-12";
        double RequestTimestamp = 1465185768;  // Timestamp: 2019-02-26 00:44:25. T
        // long timestamp = ToTimestamp() / 1000;
        // string requestTimestamp = timestamp.ToString();
        Dictionary<string, string> param = new Dictionary<string, string>();
        param.Add("Limit", "20");
        param.Add("Offset", "0");
        param.Add("InstanceIds.0", "ins-09dx96dg");
        param.Add("Action", action);
        param.Add("Nonce", "11886");
        // param.Add("Nonce", Math.Abs(new Random().Next()).ToString());

        param.Add("Timestamp", RequestTimestamp.ToString());
        param.Add("Version", version);

        param.Add("SecretId", SECRET_ID);
```

```
        param.Add("Region", region);
        SortedDictionary<string, string> headers = new SortedDictionary<string, str
        string sigInParam = MakeSignPlainText(headers, "GET", endpoint, "/");
        Console.WriteLine(sigInParam);
        string sigOutParam = Sign(SECRET_KEY, sigInParam);


        Console.WriteLine("GET https://cvm.tencentcloudapi.com");
        foreach (KeyValuePair<string, string> kv in headers)
        {
            Console.WriteLine(kv.Key + ": " + kv.Value);
        }
        Console.WriteLine("Signature" + ": " + WebUtility.UrlEncode(sigOutParam));
        Console.WriteLine();

        string result = "https://cvm.tencentcloudapi.com/?";
        foreach (KeyValuePair<string, string> kv in headers)
        {
            result += WebUtility.UrlEncode(kv.Key) + "=" + WebUtility.UrlEncode(kv.
        }
        result += WebUtility.UrlEncode("Signature") + "=" + WebUtility.UrlEncode(si
        Console.WriteLine("GET " + result);
    }
}
```

API 2.0 Signature

This signature version has been disused. We recommend you use **API 3.0 signature** with better performance. If you still need to use it, please go to API Explorer > **Signature Generation** and select **API 2.0 Signature** as the signature version.

Signature Failure

The following error codes may be returned for signature failure. Please resolve the errors accordingly.

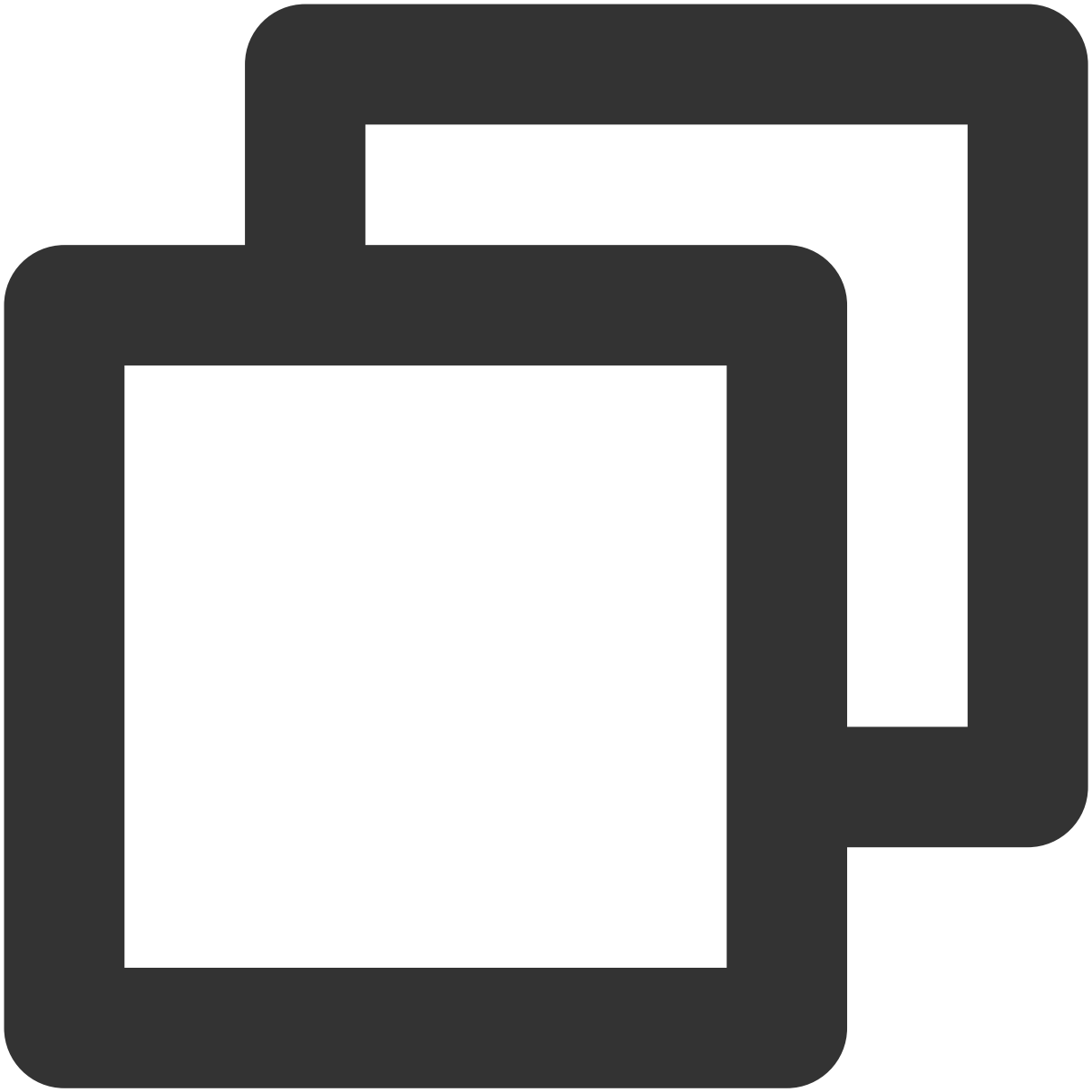
Error Code	Error Description
AuthFailure.SignatureExpire	The signature expired. The difference between the `Timestamp` and the server time cannot be greater than five minutes.
AuthFailure.SecretIdNotFound	The key does not exist. Log in to the console and check whether it is disabled or you copied fewer or more characters.

AuthFailure.SignatureFailure	Signature error. It is possible that the signature is calculated incorrectly, the signature does not match the content that is actually sent, or the `SecretKey` is incorrect.
AuthFailure.TokenFailure	Temporary credential token error.
AuthFailure.InvalidSecretId	Invalid key (not TencentCloud API key type).

Returned Result

Successful response

For example, when calling the CVM API `DescribeInstancesStatus` (version: 2017-03-12) to view the status of instances, if the request succeeds, you may see the following response:



```
{
    "Response": {
        "TotalCount": 0,
        "InstanceStatusSet": [],
        "RequestId": "b5b41468-520d-4192-b42f-595cc34b6c1c"
    }
}
```

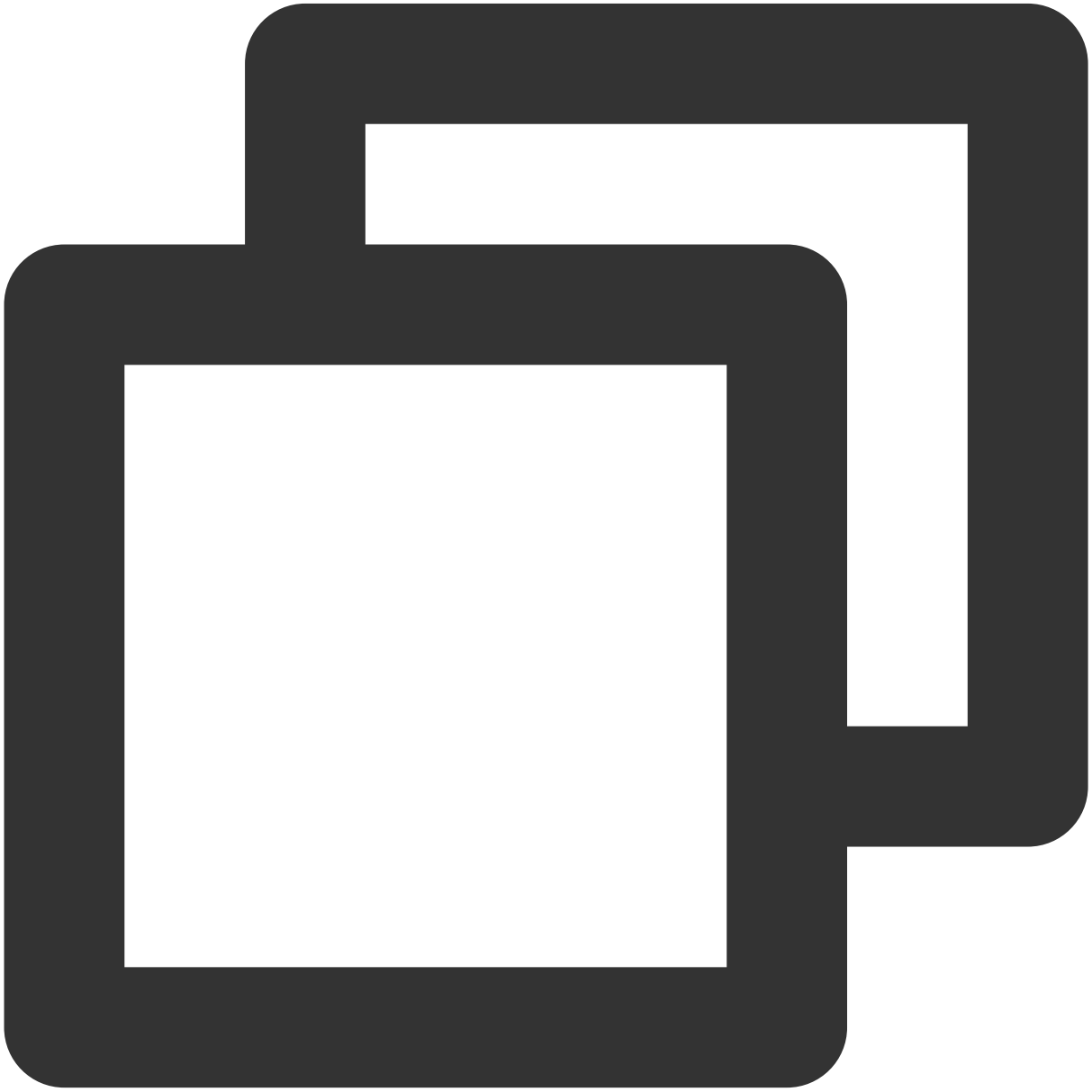
The API will return `Response` , which contains `RequestId` , as long as it processes the request, no matter whether the request is successful or not.

`RequestId` is the unique ID of an API request. It is required to troubleshoot issues.

Any fields other than the common fields are API-specific. For more information on such fields, please see the relevant API documentation. In this example, both `TotalCount` and `InstanceStatusSet` are specific to the `DescribeInstancesStatus` API. Since the user who initiated the request does not have a CVM instance yet, 0 is returned for `TotalCount` and `InstanceStatusSet` is empty.

Error response

If the call fails, you may see the following response:



```
{
```

```
    "Response": {
        "Error": {
            "Code": "AuthFailure.SignatureFailure",
            "Message": "The provided credentials could not be validated. Please che
        },
        "RequestId": "ed93f3cb-f35e-473f-b9f3-0d451b8b79c6"
    }
}
```

`Error` indicates that the request failed. A response for a failed request will always include the `Error`, `Code`, and `Message` fields.

`Code` indicates the specific error code, which is returned when an API request failed. You can use this code to locate the cause and solution of the error in the common or API-specific error code list.

`Message` explains the cause of the error. Note that the returned messages are subject to service updates. The information the messages provide may not be up-to-date and should not be the only source of reference.

`RequestId` is the unique ID of an API request. It is required to troubleshoot issues.

Common error codes

The `Error` field in a response indicates that the API call failed. The `Code` field in `Error` indicates the error code. The following table lists the common error codes that any services may return.

Error Code	Error Description
AuthFailure.InvalidSecretId	Invalid key (not TencentCloud API key type).
AuthFailure.MFAFailure	MFA failure.
AuthFailure.SecretIdNotFound	The key does not exist.
AuthFailure.SignatureExpire	The signature expired.
AuthFailure.SignatureFailure	Signature error.
AuthFailure.TokenFailure	Token error.
AuthFailure.UnauthorizedOperation	No CAM authorization.
DryRunOperation	DryRun Operation. It means that the request would have succeeded, but the DryRun parameter was used.
FailedOperation	The operation failed.
InternalError	Internal error.
InvalidAction	The API does not exist.

InvalidParameter	Incorrect parameter.
InvalidParameterValue	Invalid parameter value.
LimitExceeded	The quota limit is exceeded.
MissingParameter	A parameter is missing.
NoSuchVersion	The API version does not exist.
RequestLimitExceeded	The request rate limit is exceeded.
ResourceInUse	The resource is in use.
ResourceInsufficient	Insufficient resource.
ResourceNotFound	The resource does not exist.
ResourceUnavailable	The resource is unavailable.
UnauthorizedOperation	Unauthorized operation.
UnknownParameter	Unknown parameter error.
UnsupportedOperation	Unsupported operation.
UnsupportedProtocol	Unsupported HTTPS request method. Only GET and POST requests are supported.
UnsupportedRegion	Unsupported region.

API for Go

Last updated：2023-03-07 18:16:40

TencentCloud API has been upgraded to v3.0. This version is optimized for performance and deployed in all regions. It supports nearby access and access by region for significantly reduced access latency. In addition, it features more detailed API descriptions and error codes and API-level comments for SDKs, enabling you to use Tencent Cloud services more conveniently and quickly. This document describes how to call APIs for Go.
This version currently supports various Tencent Cloud services such as CVM, CBS, VPC, and TencentDB and will support more services in the future.

Request Structure

1. Service address (endpoint)

TencentCloud API supports access from either a nearby region (such as `cvm.tencentcloudapi.com` for CVM) or a specified region (such as `cvm.ap-guangzhou.tencentcloudapi.com` for CVM in the Guangzhou region). For values of the region parameter, please see the region list in the "Common Parameters" section below. To check whether a region is supported by a specific Tencent Cloud service, please see its "Request Structure" document.

Note:

For latency-sensitive businesses, we recommend you specify a domain name with a region.

2. Communications protocol

All TencentCloud APIs communicate over HTTPS, providing highly secure communications tunnels.

3. Request method

Supported HTTP request methods:
POST (recommended)
GET
`Content-Type` types supported by POST request:
application/json (recommended). The signature algorithm v3 (TC3-HMAC-SHA256) must be used.
application/x-www-form-urlencoded. The signature algorithm v1 (HmacSHA1 or HmacSHA256) must be used.
multipart/form-data (only supported by certain APIs). The signature algorithm v3 (TC3-HMAC-SHA256) must be used.
The size of a GET request packet cannot exceed 32 KB. The size of a POST request cannot exceed 1 MB for the signature algorithm v1 (HmacSHA1 or HmacSHA256) or 10 MB for the signature algorithm v3 (TC3-HMAC-SHA256).

4. Character encoding

UTF-8 encoding is always used.

Common Parameters

Note:
The common parameters are used to identity the user and API signature. They should be carried by each request to initiate properly.

Signature algorithm v3

The signature algorithm v3 (sometimes referred to as "TC3-HMAC-SHA256") is more secure than the signature algorithm v1 (referred to as signature algorithm in certain documents), supports larger request packets and POST JSON format, and has a higher performance. We recommend you use it to calculate signatures. For more information on how to use it, please see below.

Parameter Name	Type	Required	Description
X-TC-Action	String	Yes	Name of the API for the desired operation. For the specific value, please see the description of common parameter `Action` in the input parameters in the related API document. For example, the API for querying CVM instance list is `DescribeInstances` .
X-TC-Region	String	-	Region parameter, which is used to identify the region where the data you want to manipulate resides. For values supported for an API, please see the description of common parameter `Region` in the input parameters in related API documentation. **Note: this parameter is not required for some APIs (which will be indicated in related API documentation) and will not take effect even if it is passed.**
X-TC-Timestamp	Integer	Yes	The current UNIX timestamp that records the time when the API request was initiated, such as 1529223702. **Note: if the difference between the UNIX timestamp and the server time is greater than 5 minutes, a signature expiration error may occur.**
X-TC-Version	String	Yes	Version of the API for the desired operation, such as 2017-03-12 for CVM. For the specific value, please see the description of common parameter `Version` in the input parameters in related API documentation.
Authorization	String	Yes	HTTP authentication request header, such as TC3-HMAC-SHA256 Credential=AKIDEXAMPLE/Date/service/tc3_request,

			SignedHeaders=content-type;host, Signature=72e494ea8**********a96525168 Here, TC3-HMAC-SHA256: signature algorithm, currently fixed as this value. Credential: signature credential. `AKIDEXAMPLE` indicates the `SecretId`. `Date` indicates a UTC date which must match the value of `X-TC-Timestamp` (a common parameter) in UTC format. `service` indicates the name of the service and is generally a domain name prefix; for example, the domain name `cvm.tencentcloudapi.com` means the CVM service, and the value for this service is `cvm`. SignedHeaders: the headers that contain the authentication information. `content-type` and `host` are required. Signature: signature digest. For the calculation process, please see below.
X-TC-Token	String	No	Token used for temporary credentials. It must be used with a temporary key. You can get the temporary key and token by calling a CAM API. No token is required for a long-term key.

Signature algorithm v1

When the signature algorithm v1 (sometimes referred to as "HmacSHA256" or "HmacSHA1") is used, the common parameters should be uniformly placed in the request string.

Parameter Name	Type	Required	Description
Action	String	Yes	Name of the API for the desired operation. For the specific value, please see the description of common parameter `Action` in the input parameters in the related API document. For example, the API for querying CVM instance list is `DescribeInstances`.
Region	String	-	Region parameter, which is used to identify the region where the data you want to manipulate resides. For values supported for an API, please see the description of common parameter `Region` in the input parameters in related API documentation. **Note: this parameter is not required for some APIs (which will be indicated in related API documentation) and will not take effect even if it is passed.**
Timestamp	Integer	Yes	The current UNIX timestamp that records the time when the API request was initiated, such as 1529223702. If the

			difference between the UNIX timestamp and the current time is too large, a signature expiration error may occur.
Nonce	Integer	Yes	A random positive integer used in conjunction with `Timestamp` to prevent replay attacks.
SecretId	String	Yes	The identifying `SecretId` obtained on the TencentCloud API Key page. A `SecretId` corresponds to a unique `SecretKey` which is used to generate the request signature (`Signature`).
Signature	String	Yes	Request signature, which is used to verify the validity of the request. It is generated based on input parameters. For more information on how to calculate the signature, please see below.
Version	String	Yes	Version of the API for the desired operation, such as 2017-03-12 for CVM. For the specific value, please see the description of common parameter `Version` in the input parameters in related API documentation.
SignatureMethod	String	No	Signature algorithm. Currently, only HmacSHA256 and HmacSHA1 are supported. The HmacSHA256 algorithm is used to verify the signature only when this parameter is specified as HmacSHA256. In other cases, the signature is verified with HmacSHA1.
Token	String	No	Token used for temporary credentials. It must be used with a temporary key. You can get the temporary key and token by calling a CAM API. No token is required for a long-term key.

Region list

As the supported regions vary by service, please refer to the region list in each service's product documentation for specific details.
For example, you can see the region list of CVM.

API Call Method for Go

TencentCloud API authenticates every request, that is, the request must be signed with the security credentials in the designated steps. Each request must contain the signature information in the common request parameters and be sent in the specified way and format.

Suppose your `SecretId` and `SecretKey` are `AKIDz8krbsJ5*********mLPx3EXAMPLE` and `Gu5t9xGAR**********EXAMPLE`, respectively. If you want to view the status of an unnamed instance in the Guangzhou region and have only one data entry returned, the request may be:

```
curl -X POST https://cvm.tencentcloudapi.com \\
-H "Authorization: TC3-HMAC-SHA256 Credential=AKIDz8krbsJ5*********mLPx3EXAMPLE/20
-H "Content-Type: application/json; charset=utf-8" \\
-H "Host: cvm.tencentcloudapi.com" \\
-H "X-TC-Action: DescribeInstances" \\
-H "X-TC-Timestamp: 1551113065" \\
-H "X-TC-Version: 2017-03-12" \\
```

```
-H "X-TC-Region: ap-guangzhou" \\
-d '{"Limit": 1, "Filters": [{"Values": ["\\u672a\\u547d\\u540d"], "Name": "instanc
```

Step 1. Apply for security credentials

In this document, the security credential used is a key pair, which consists of a `SecretId` and a `SecretKey`.

Each user can have up to two key pairs.

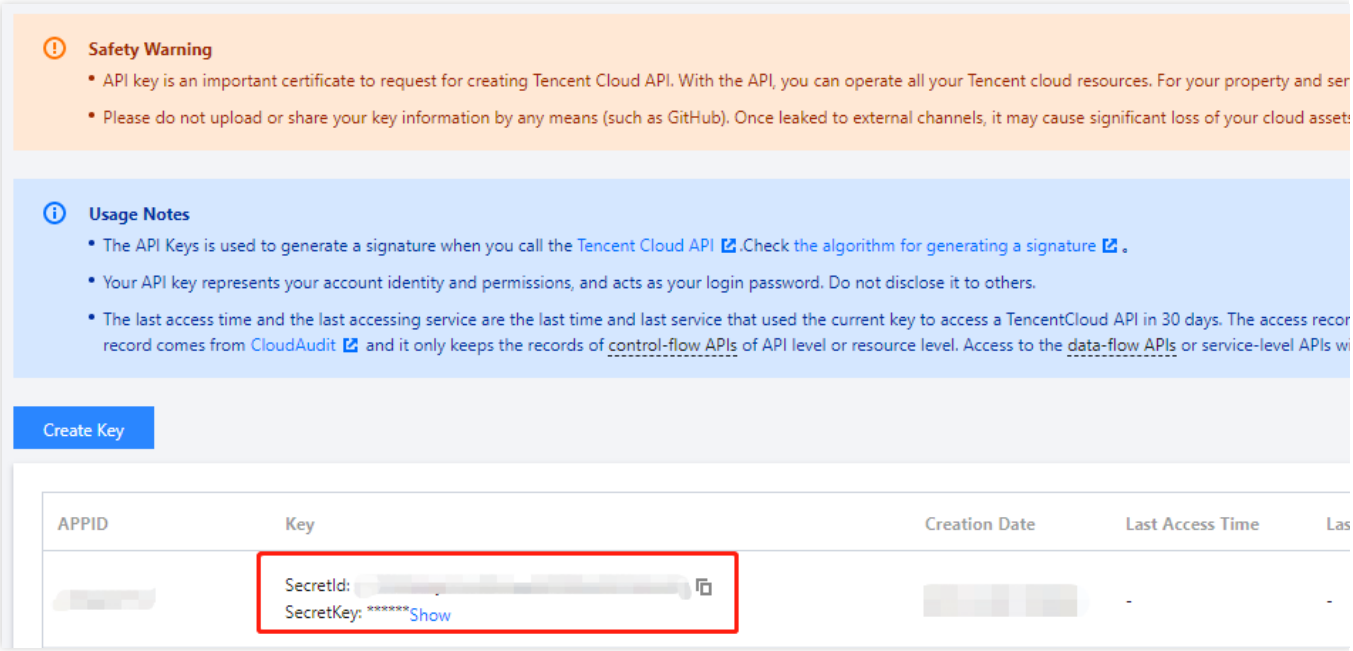
SecretId: identifies the user that calls an API, which is similar to a username.

SecretKey: authenticates the user that calls the API, which is similar to a password.

Note:

You must keep your security credentials private and avoid disclosure; otherwise, your assets may be compromised. If they are disclosed, please disable them as soon as possible.

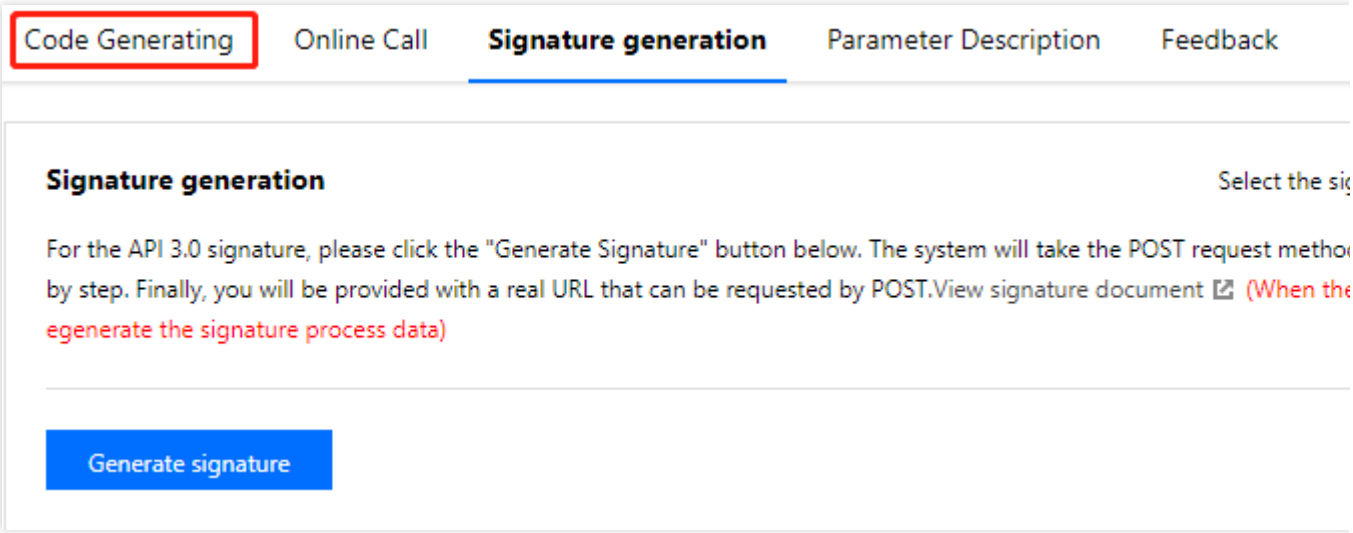
Go to the API key management page to get API keys as shown below:



Step 2

1. Get an API 3.0 signature v3

The signature algorithm v3 (TC3-HMAC-SHA256) is compatible with the previous signature algorithm v1 and more secure, supports larger request packets and POST JSON format, and has a higher performance. We recommend you use it to calculate signatures.



Note:

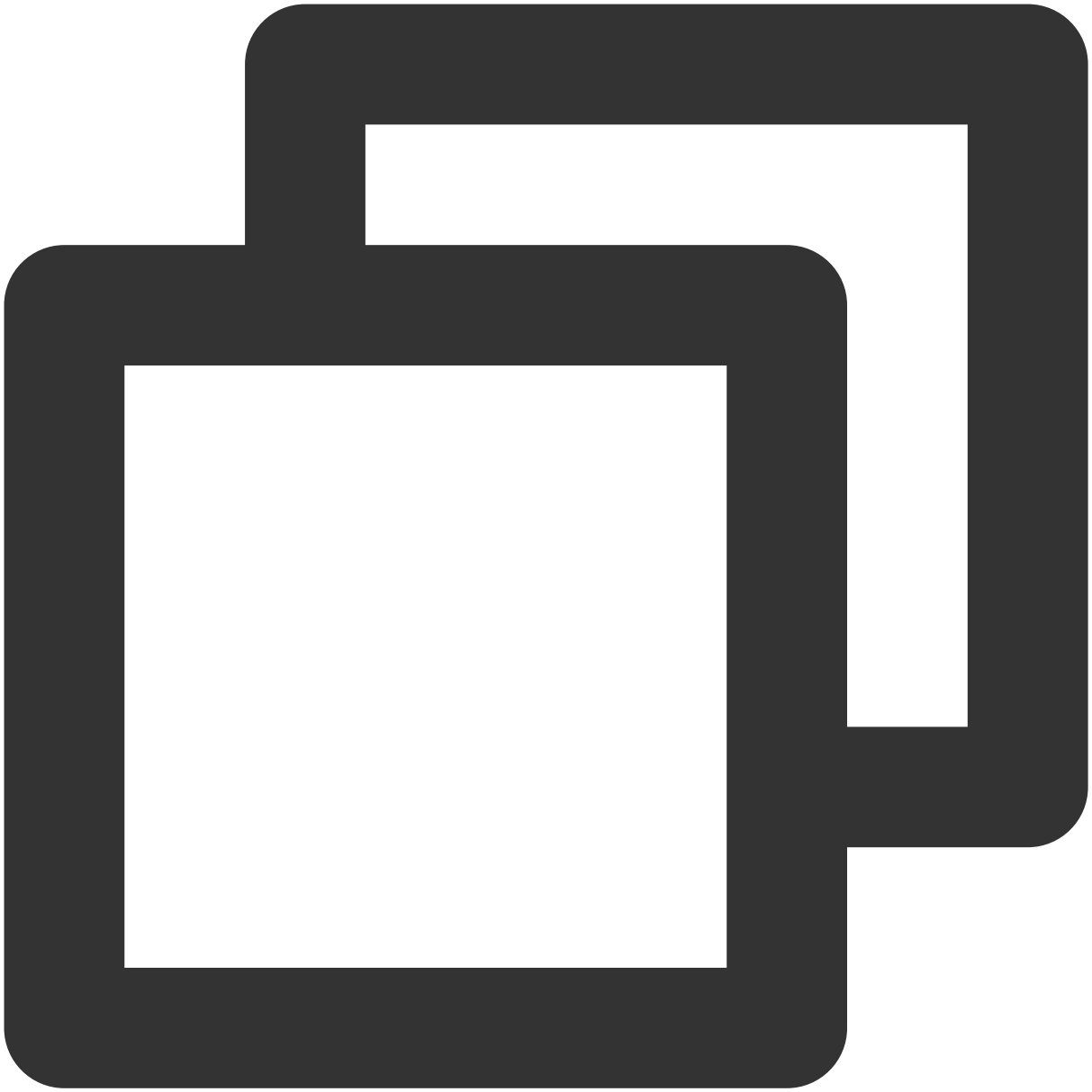
If you are using the signature algorithm for the first time, we recommend you use the "signature string generation" feature in API Explorer and select "API 3.0 signature v3" as the signature version, which can generate a signature for demonstration and verification. Plus, it can also generate SDK code directly. Seven common open-source programming language SDKs are available for TencentCloud API, including Python, Java, PHP, Go, Node.js, .NET, and C++.

TencentCloud API supports both GET and POST requests. For the GET method, only the `Content-Type: application/x-www-form-urlencoded` protocol format is supported. For the POST method, `Content-Type: application/json` and `Content-Type: multipart/form-data` are supported. The JSON format is supported by all business APIs, while the multipart format is supported only by specific APIs (in this case, an API cannot be called in JSON format). For more information, please see the specific business API document. We recommend you use the POST method because the two methods generate the same results, but the GET method only supports request packets below 32 KB in size.

The following describes how to calculate a signature by calling the DescribeInstances API. This API is chosen because:

1. The CVM API is enabled by default, and this API is often used.
2. It is read-only and does not change the status of existing resources.
3. It covers many types of parameters so that it is easy to show how to use an array that contains data structures.

1. Concatenate the canonical request string

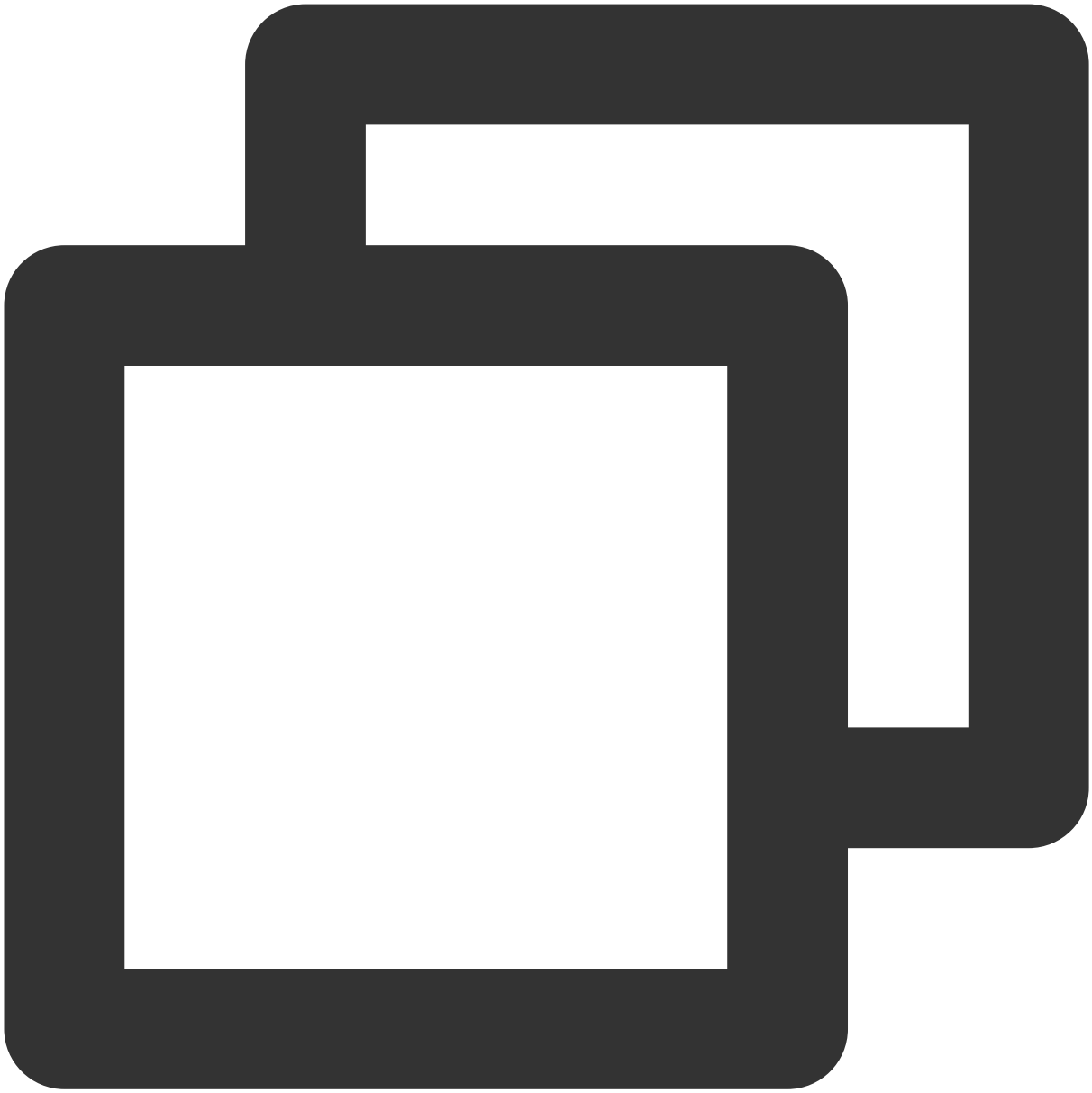


```
CanonicalRequest =
    HTTPRequestMethod + '\\n' +
    CanonicalURI + '\\n' +
    CanonicalQueryString + '\\n' +
    CanonicalHeaders + '\\n' +
    SignedHeaders + '\\n' +
    HashedRequestPayload
```

Field	Description

HTTPRequestMethod	HTTP request method (GET or POST). This example uses `POST` .
CanonicalURI	URI parameter. Slash ("/") is used for API 3.0.
CanonicalQueryString	Query string in the URL of the originating HTTP request. This is always an empty string POST requests and is the string after the question mark (?) for GET requests such as `Limit=10&Offset=0` . Note: `CanonicalQueryString` must be URL-encodec instructed in RFC 3986 with the UTF-8 character set. The applicable standard program language library is recommended. All special characters must be encoded and capitaliz
CanonicalHeaders	Header information for signature calculation, including at least `host` and `content type` . Custom headers can also be added to the signature process to improve the uniqueness and security of the request. Concatenation rules: both the key and value of header should be converted to lowercase with the leading and trailing spaces removed that they are concatenated in the `key:value\\n` format. If there are multiple heade they should be sorted in ASCII ascending order by header key (lowercase). The calcula result in this example is `content-type:application/json; charset=utf-8\\nhost:cvm.tencentcloudapi.com\\n` . Note: `content-type` must mat the content that is actually sent. In some programming languages, a `charset` value automatically added even if it is not specified. In this case, the request sent will be differ from the one signed, and the sever will return a signature verification failure.
SignedHeaders	Header information for signature calculation, indicating the request headers that are inv in the signature process. The request headers must correspond to the headers in `CanonicalHeaders` . `Content-type` and `host` are required headers. Concatenation rules: both the key and value of a header should be converted to lowerca there are multiple headers, they should be sorted in ASCII ascending order by header k (lowercase) and separated by semicolons (;). The value in this example is `content-type;host` .
HashedRequestPayload	Hash value of `Requestpayload` (i.e., the request body, such as `{"Limit": 1, "Filters": [{"Values": ["\\u672a\\u547d\\u540d"], "Name": "instance-name"}]}` in this example). The pseudo-code for calculation is `Lowercase(HexEncode(Hash.SHA256(RequestPayload)))` , which means t SHA256 hashing is performed on the payload of the HTTP request, then hexadecimal encoding is performed, and finally the encoded string is converted to lowercase letters. GET requests, `RequestPayload` is always an empty string. The calculation result i example is `35e9c5b0e3ae67532d3c9f17ead6c90222632e5b1ff7f6e89887f1398934f(`

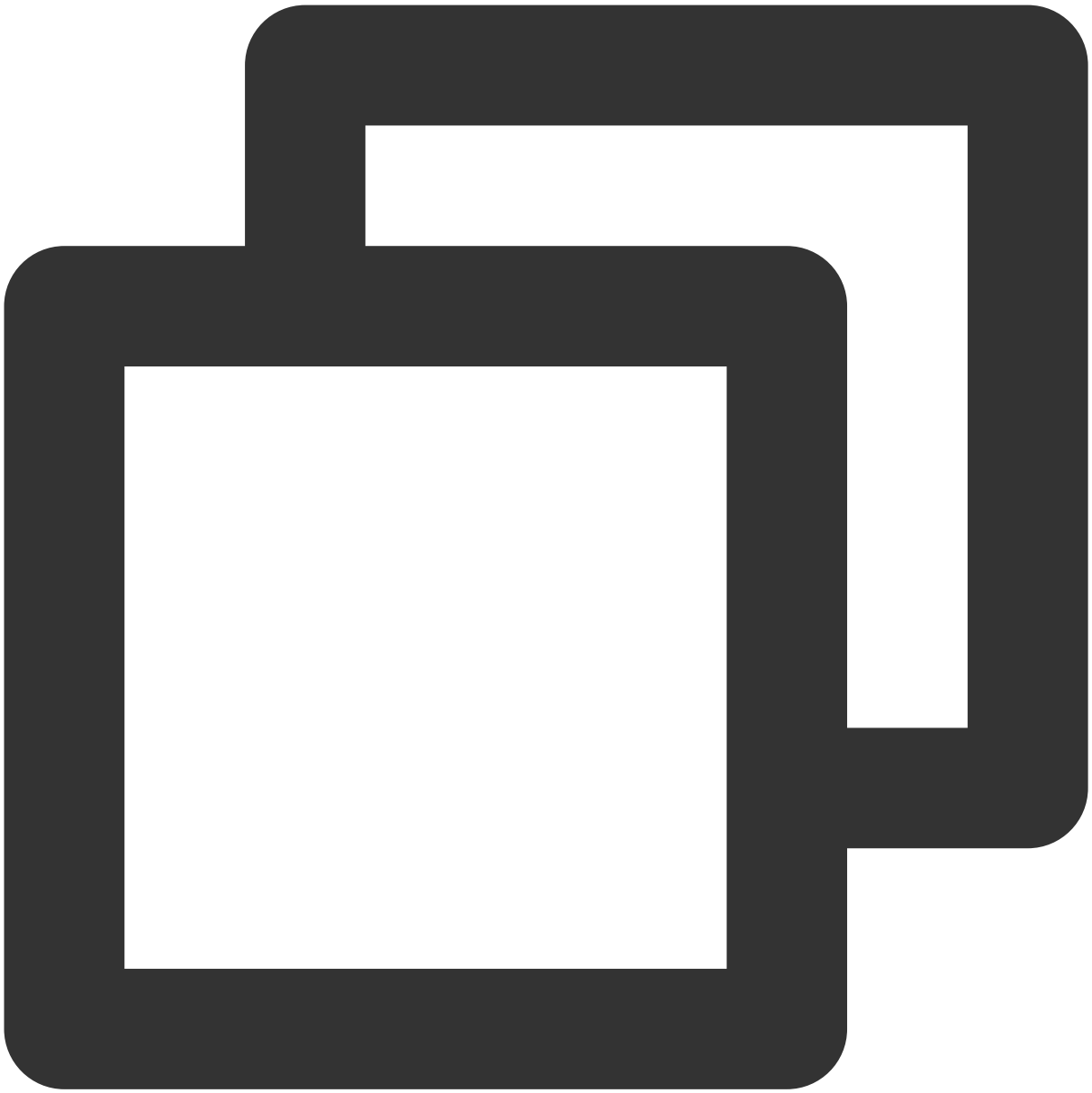
According to the rules above, the canonical request string obtained in the example is as follows:



```
POST
/

content-type:application/json; charset=utf-8
host:cvm.tencentcloudapi.com

content-type;host
35e9c5b0e3ae67532d3c9f17ead6c90222632e5b1ff7f6e89887f1398934f064
```

2. Concatenate the string to sign

Concatenate the string to sign in the following format:



```
StringToSign =
    Algorithm + \\n +
    RequestTimestamp + \\n +
    CredentialScope + \\n +
    HashedCanonicalRequest
```

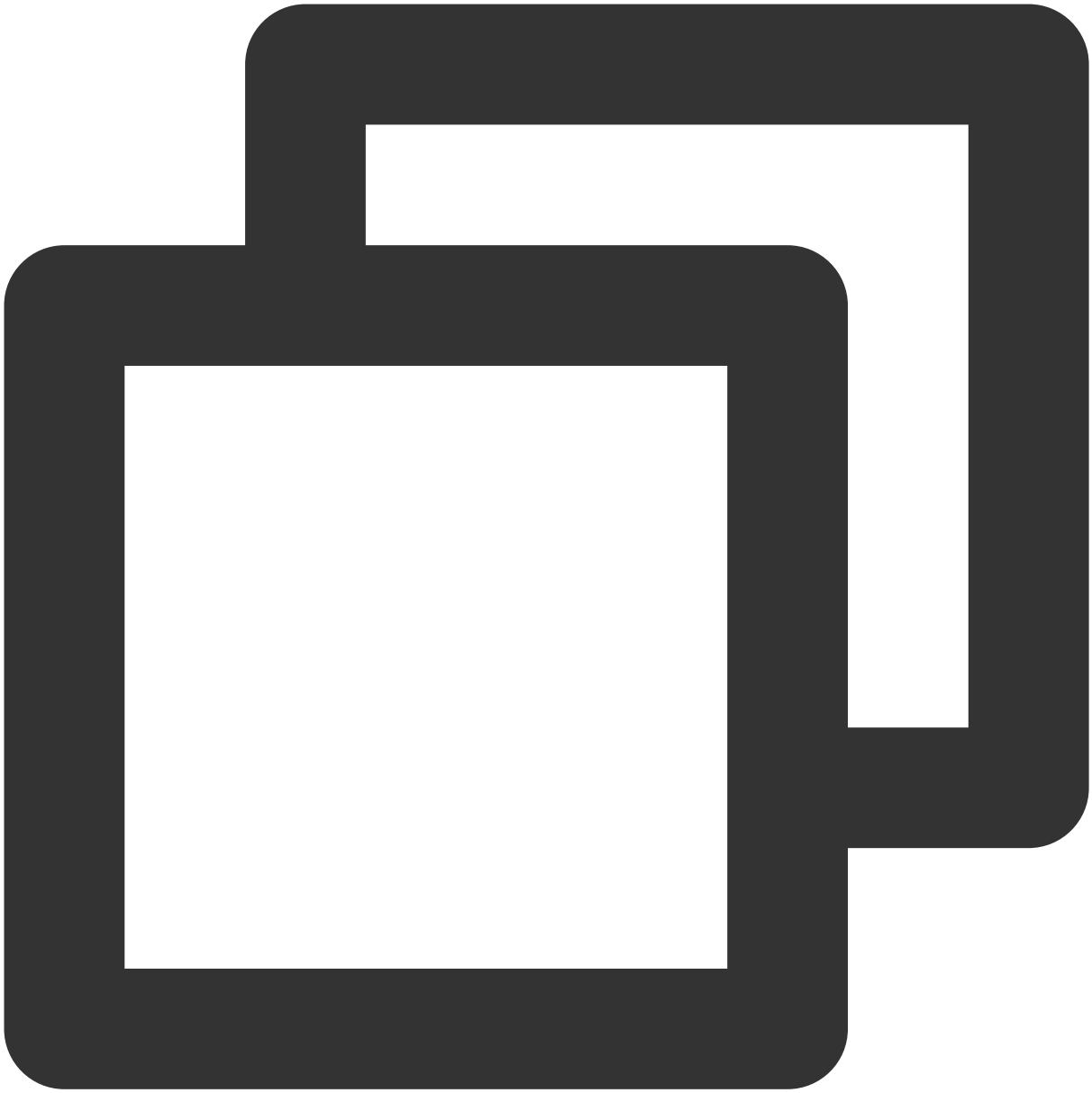
Field	Description
Algorithm	Signature algorithm, which is always `TC3-HMAC-SHA256` currently.

RequestTimestamp	Request timestamp, i.e., the value of the common parameter `X-TC-Timestamp` in request header. It is the UNIX timestamp of the current time in seconds, such as `1551113065` in this example.
CredentialScope	Scope of the credential in the format of `Date/service/tc3_request`, including date, requested service, and termination string (tc3_request). **`Date` indicates a U date, which should match the UTC date converted by the common parameter `TC-Timestamp`.** `service` is the service name, which should match the domain n of the service called. The calculation result in this example is `2019-02-25/cvm/tc3_request`.
HashedCanonicalRequest	Hash value of the canonical request string concatenated in the steps above. The pseu code for calculation is `Lowercase(HexEncode(Hash.SHA256(CanonicalRequest)))`. The calcu result in this example is `5ffe6a04c0664d6b969fab9a13bdab201d63ee709638e2749d62a09ca18d`

Note:

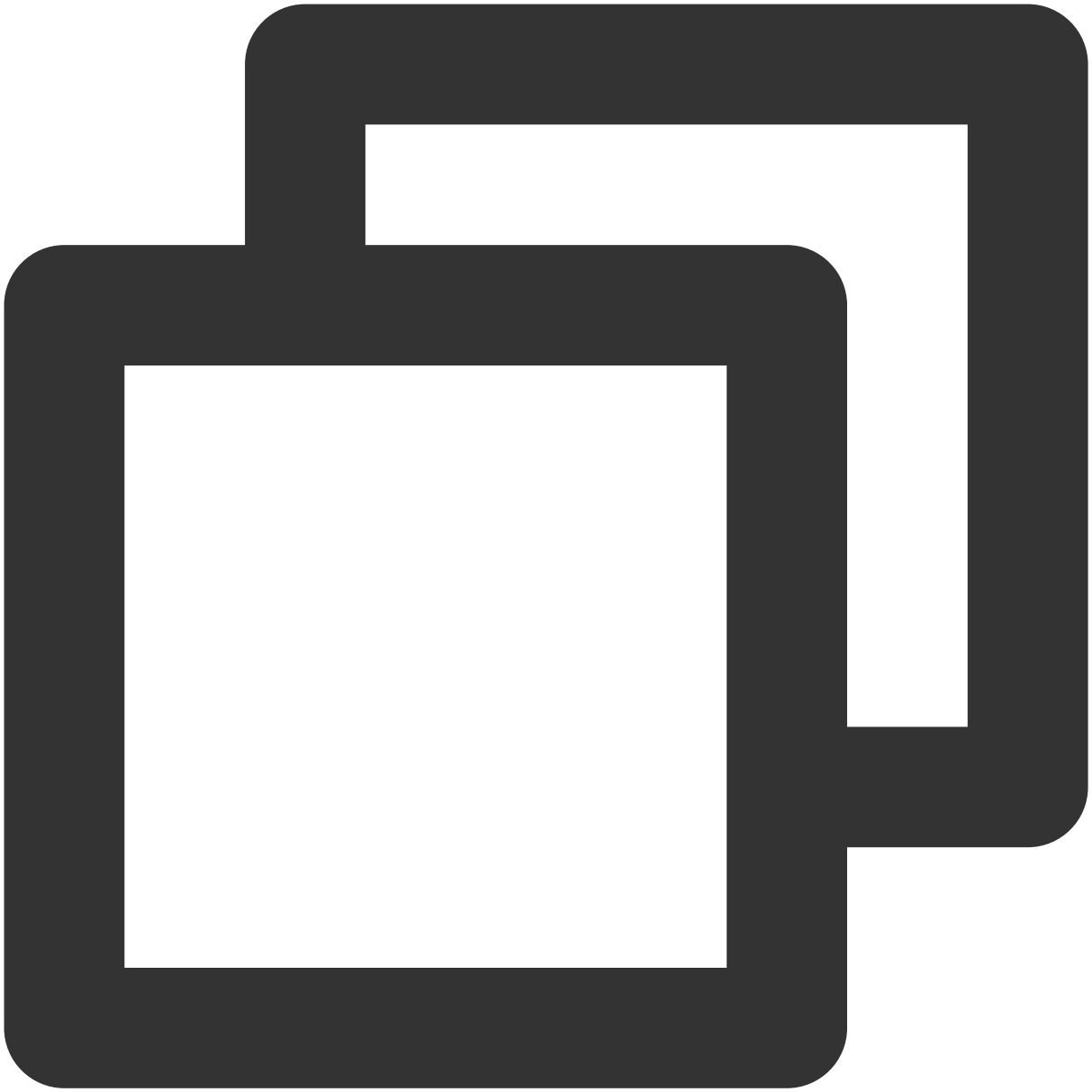
1. `Date` must be calculated from the timestamp `X-TC-Timestamp` and the time zone is UTC+0. If you add the local time zone information (such as UTC+8) in the system, calls can succeed both day and night but will definitely fail at 00:00. For example, if the timestamp is 1551113065 and the time in UTC+8 is 2019-02-26 00:44:25, the UTC+0 date in the calculated `Date` value should be 2019-02-25 instead of 2019-02-26.
2. `Timestamp` must be the same as your current system time, and your system time must be in sync with the UTC time. If the difference between the timestamp and your current system time is greater than five minutes, the request will fail. If your system time is out of sync with the UTC time for a prolonged period, the request will fail, and a signature expiration error will be returned.

According to the rules above, the string to sign obtained in the example is as follows:



```
TC3-HMAC-SHA256
1551113065
2019-02-25/cvm/tc3_request
5ffe6a04c0664d6b969fab9a13bdab201d63ee709638e2749d62a09ca18d7031
```

3. Calculate the signature (pseudocode)



```
secretDate := hmacsha256(date, "TC3"+secretKey)
secretService := hmacsha256(service, secretDate)
secretSigning := hmacsha256("tc3_request", secretService)
signature := hex.EncodeToString([]byte(hmacsha256(string2sign, secretSigning)))
fmt.Println(signature)
```

The derived key `SecretDate` , `SecretService` , and `SecretSigning` are binary data and may contain non-printable characters. Intermediate results are not displayed here.

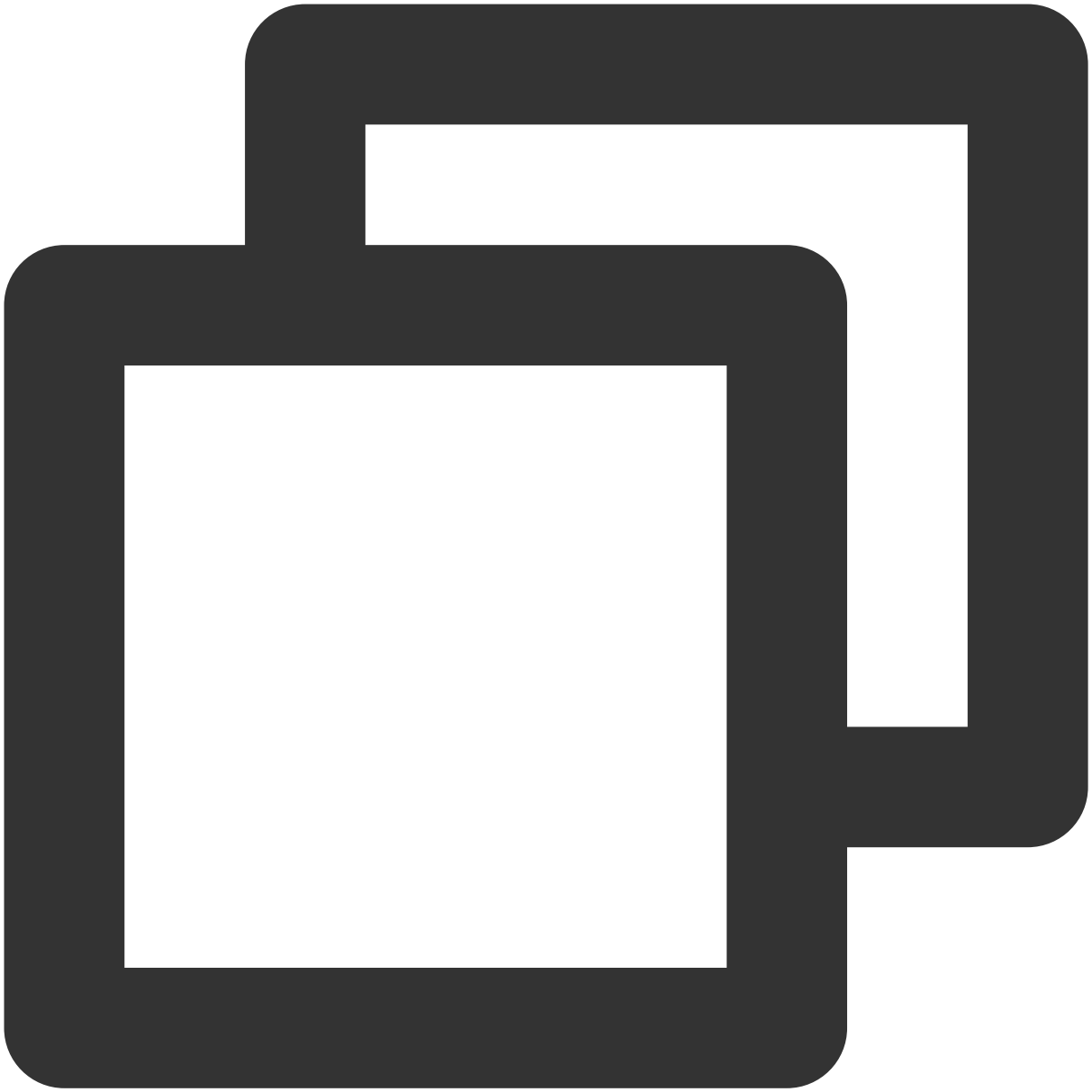
Field	Description

secretKey	Original `secretKey` , i.e., `Gu5t9xGAR**********EXAMPLE` .
date	Value of the `Date` field (UTC) in `Credential` , such as `2019-02-25` in this example.
service	Value of the `Service` field in `Credential` , such as `cvm` in this example.

The calculation result in this example is `72e494ea8****************************************a96525168` .

4. Concatenate the Authorization string

Concatenate the `Authorization` string in the following format:

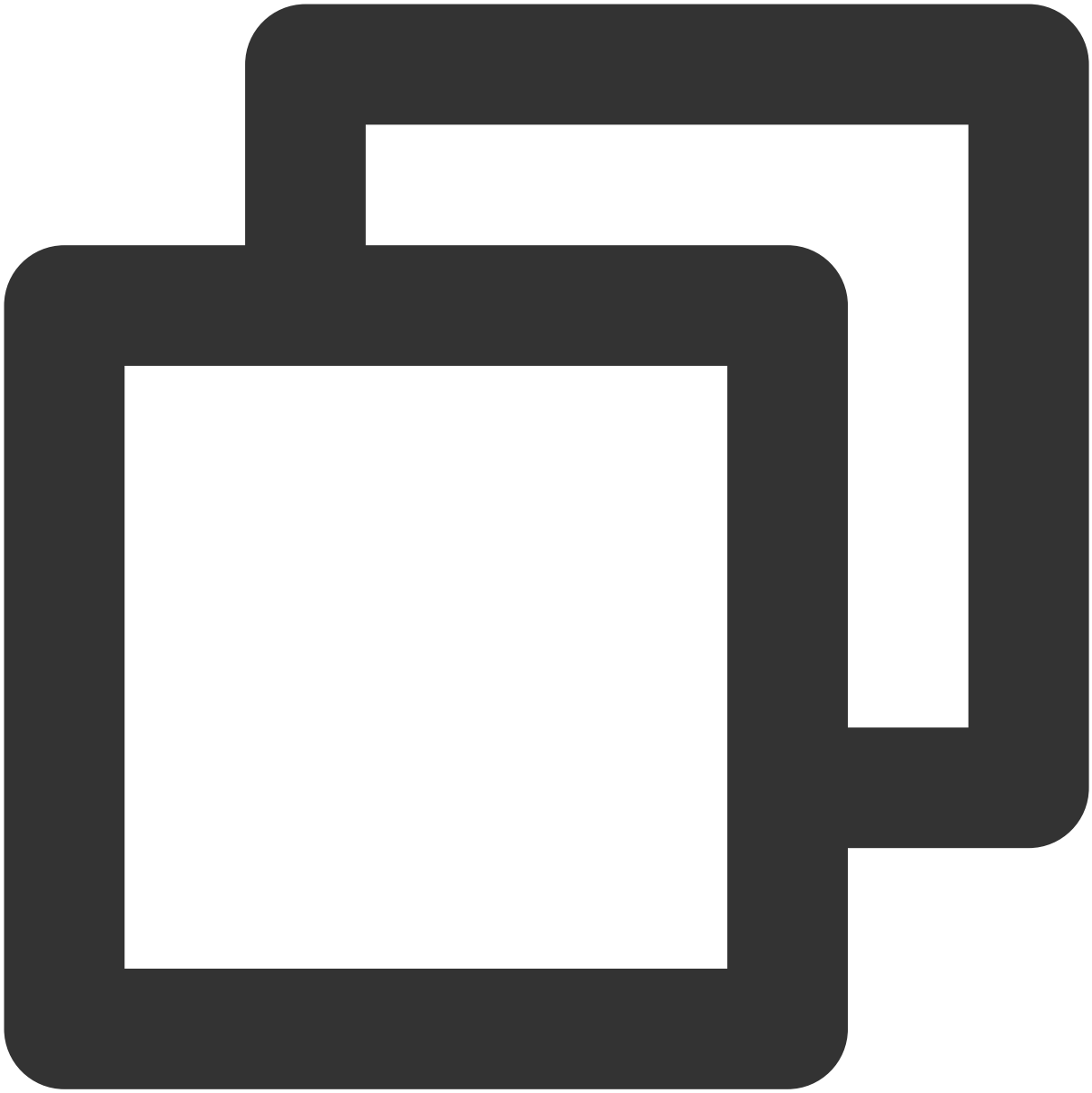


```
    authorization := fmt.Sprintf("%s Credential=%s/%s, SignedHeaders=%s, Signature=
        algorithm,
        secretId,
        credentialScope,
        signedHeaders,
        signature)
    fmt.Println(authorization)
```

Field	Description

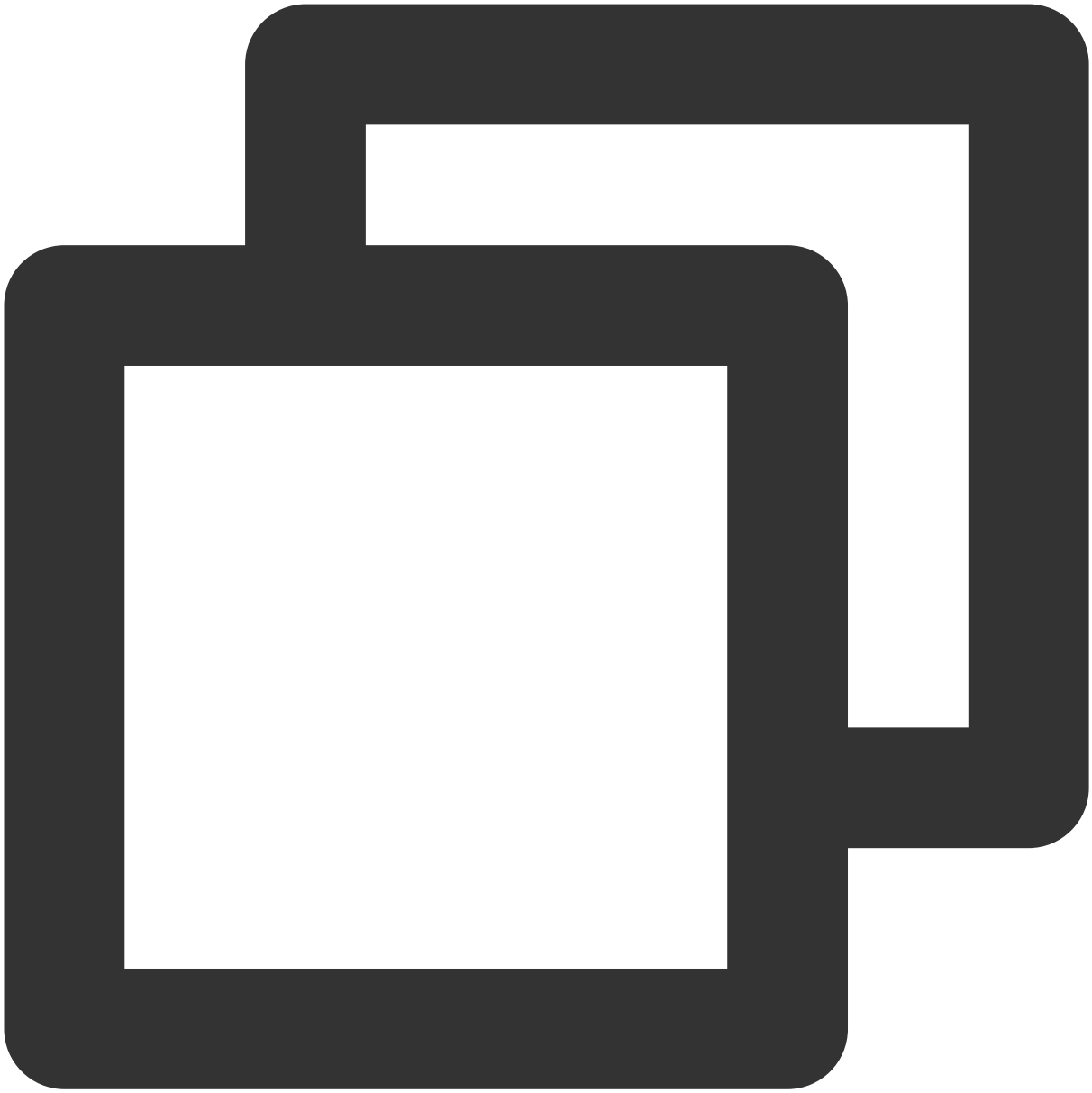
algorithm	Signature algorithm, which is always `TC3-HMAC-SHA256` .
secretId	`SecretId` in the key pair, i.e., `AKIDz8krbsJ5**********mLPx3EXAMPLE` .
credentialScope	Credential scope (see above). The calculation result in this example is `2019-02-25/cvm/tc3_request` .
signedHeaders	Header information for signature calculation (see above), such as `content-type;host` in this example.
signature	Signature value. The calculation result in this example is `72e494ea8****************************************a96525168` .

According to the rules above, the values obtained in this example are:



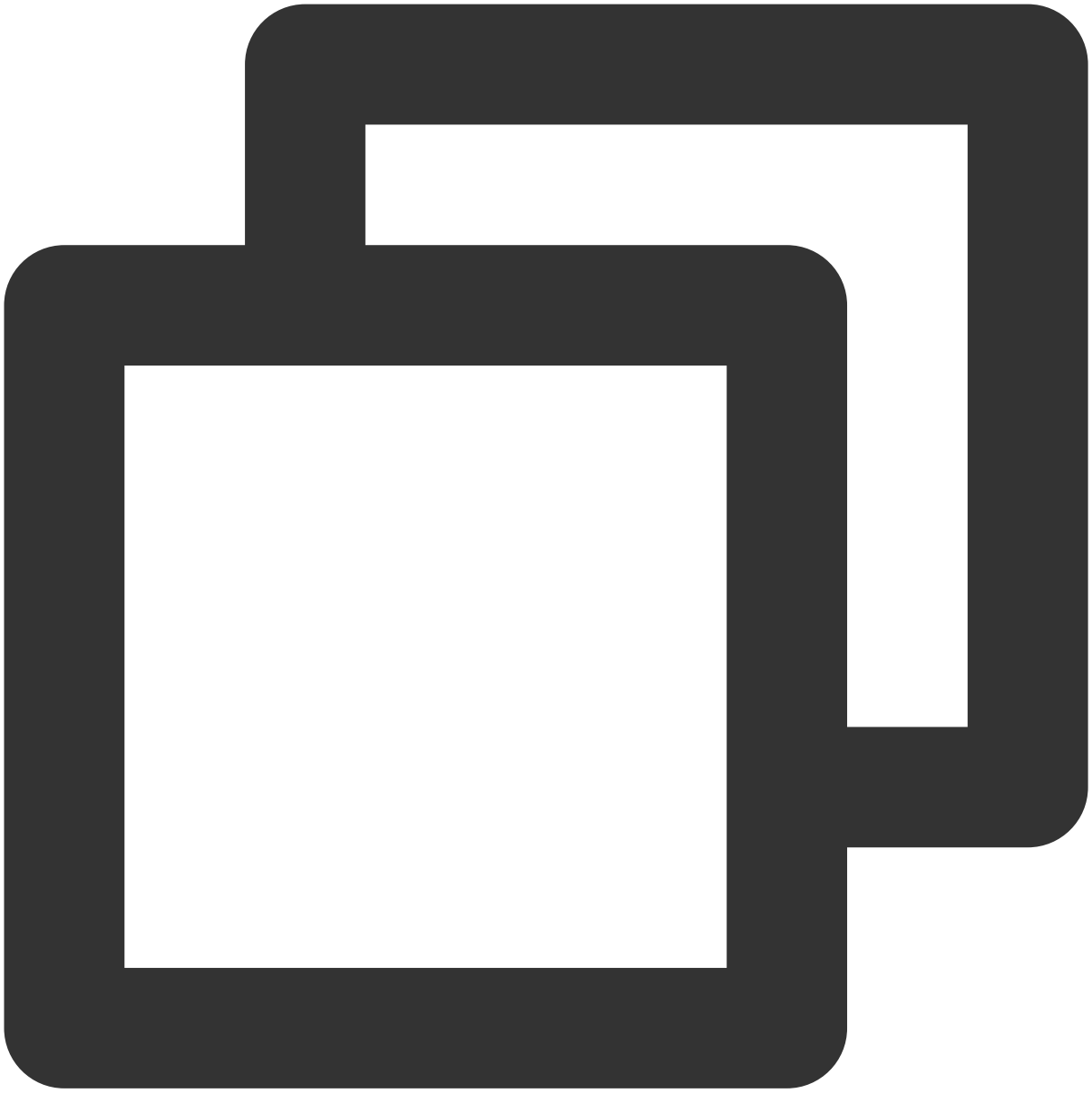
```
TC3-HMAC-SHA256 Credential=AKIDz8krbsJ5**********mLPx3EXAMPLE/2019-02-25/cvm/tc3_re
```

The complete call information is as follows:



```
POST https://cvm.tencentcloudapi.com/
Authorization: TC3-HMAC-SHA256 Credential=AKIDz8krbsJ5**********mLPx3EXAMPLE/2019-0
Content-Type: application/json; charset=utf-8
Host: cvm.tencentcloudapi.com
X-TC-Action: DescribeInstances
X-TC-Version: 2017-03-12
X-TC-Timestamp: 1551113065
X-TC-Region: ap-guangzhou

{"Limit": 1, "Filters": [{"Values": ["\\u672a\\u547d\\u540d"], "Name": "instance-na
```

5. Sample API 3.0 signature v3



```
package main

import (
    "crypto/hmac"
    "crypto/sha256"
    "encoding/hex"
    "fmt"
    "time"
)
```

```
func sha256hex(s string) string {
    b := sha256.Sum256([]byte(s))
    return hex.EncodeToString(b[:])
}

func hmacsha256(s, key string) string {
    hashed := hmac.New(sha256.New, []byte(key))
    hashed.Write([]byte(s))
    return string(hashed.Sum(nil))
}

func main() {
    secretId := "AKIDz8krbsJ5**********mLPx3EXAMPLE"
    secretKey := "Gu5t9xGAR***********EXAMPLE"
    host := "cvm.tencentcloudapi.com"
    algorithm := "TC3-HMAC-SHA256"
    service := "cvm"
    version := "2017-03-12"
    action := "DescribeInstances"
    region := "ap-guangzhou"
    //var timestamp int64 = time.Now().Unix()
    var timestamp int64 = 1551113065

    // step 1: build canonical request string
    httpRequestMethod := "POST"
    canonicalURI := "/"
    canonicalQueryString := ""
    canonicalHeaders := "content-type:application/json; charset=utf-8\\n" + "host:"
    signedHeaders := "content-type;host"
    payload := `{"Limit": 1, "Filters": [{"Values": ["\\u672a\\u547d\\u540d"], "Nam
    hashedRequestPayload := sha256hex(payload)
    canonicalRequest := fmt.Sprintf("%s\\n%s\\n%s\\n%s\\n%s\\n%s",
        httpRequestMethod,
        canonicalURI,
        canonicalQueryString,
        canonicalHeaders,
        signedHeaders,
        hashedRequestPayload)
    fmt.Println(canonicalRequest)

    // step 2: build string to sign
    date := time.Unix(timestamp, 0).UTC().Format("2006-01-02")
    credentialScope := fmt.Sprintf("%s/%s/tc3_request", date, service)
    hashedCanonicalRequest := sha256hex(canonicalRequest)
    string2sign := fmt.Sprintf("%s\\n%d\\n%s\\n%s",
        algorithm,
```

```
        timestamp,
        credentialScope,
        hashedCanonicalRequest)
    fmt.Println(string2sign)

    // step 3: sign string
    secretDate := hmacsha256(date, "TC3"+secretKey)
    secretService := hmacsha256(service, secretDate)
    secretSigning := hmacsha256("tc3_request", secretService)
    signature := hex.EncodeToString([]byte(hmacsha256(string2sign, secretSigning)))
    fmt.Println(signature)

    // step 4: build authorization
    authorization := fmt.Sprintf("%s Credential=%s/%s, SignedHeaders=%s, Signature=
        algorithm,
        secretId,
        credentialScope,
        signedHeaders,
        signature)
    fmt.Println(authorization)

    curl := fmt.Sprintf(`curl -X POST https://%s\\
 -H "Authorization: %s"\\
 -H "Content-Type: application/json; charset=utf-8"\\
 -H "Host: %s" -H "X-TC-Action: %s"\\
 -H "X-TC-Timestamp: %d"\\
 -H "X-TC-Version: %s"\\
 -H "X-TC-Region: %s"\\
 -d '%s'`, host, authorization, host, action, timestamp, version, region, payload)
    fmt.Println(curl)
}
```

2. Get an API 3.0 signature v1

The signature algorithm v1 is simple and easy to use, but its functionality and security are not as good as the signature algorithm v3 which is therefore recommended.

Note:

If you are using the signature algorithm for the first time, we recommend you use the "signature string generation" feature in API Explorer and select "API 3.0 signature v1" as the signature version, which can generate a signature for demonstration and verification and provides signing examples for certain programming languages. Plus, it can also generate SDK code directly. Seven common open-source programming language SDKs are available for TencentCloud API, including Python, Java, PHP, Go, Node.js, .NET, and C++.

For example, if you call the `DescribeInstances` API to query CVM instances, the request parameters may be as follows:

Parameter Name	Description	Value
Action	Method	DescribeInstances
SecretId	Key ID	`AKIDz8krbsJ5*********mLPx3EXAMPLE`
Timestamp	Current timestamp	1465185768
Nonce	Random positive integer	11886
Region	Instance region	ap-guangzhou
InstanceIds.0	ID of the instance to be queried	ins-09dx96dg
Offset	Offset	0
Limit	Allowed maximum number of output entries	20
Version	API version number	2017-03-12

1. Sort parameters

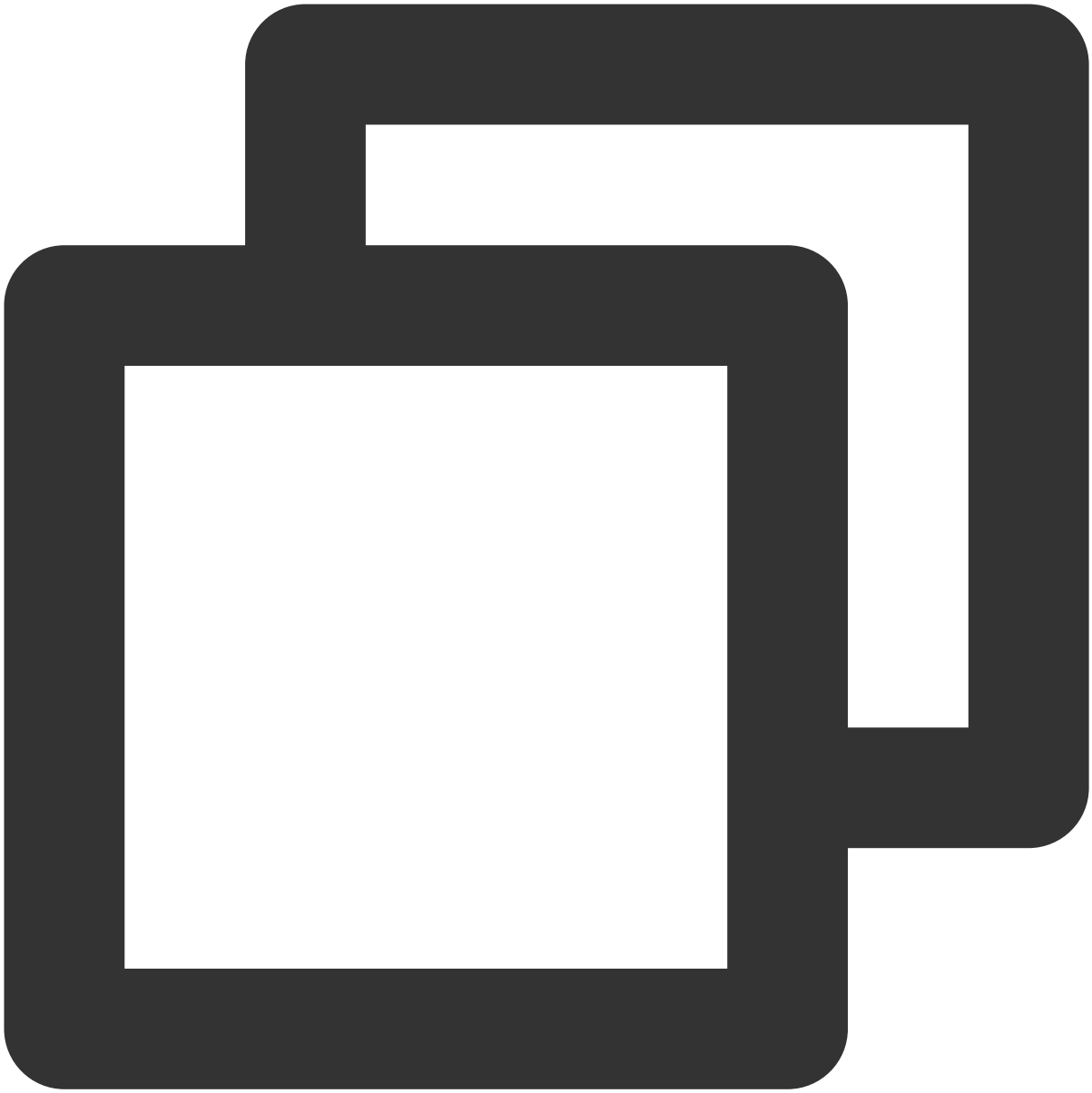
Sort all the request parameters in an ascending lexicographical order (ASCII code) by their names.

Note:

1. The parameters are sorted only by name but not by value.

2. The parameters are sorted based on ASCII code but not in an alphabetical order or by value. For example, InstanceIds.2 should be arranged behind InstanceIds.12. You can complete sorting by using a sorting function in a programming language, such as the ksort function in PHP.

The parameters in the example are sorted as follows:



```
{
    'Action' : 'DescribeInstances',
    'InstanceIds.0' : 'ins-09dx96dg',
    'Limit' : 20,
    'Nonce' : 11886,
    'Offset' : 0,
    'Region' : 'ap-guangzhou',
    'SecretId' : 'AKIDz8krbsJ5**********mLPx3EXAMPLE',
    'Timestamp' : 1465185768,
    'Version': '2017-03-12',
}
```

Any other programming languages can be used to sort these parameters as long as the same result is produced.

2. Concatenate the canonical request string

This step generates a request string. Format the request parameters sorted in the previous step into the form of `parameter=value`. For example, for the `Action` parameter, its parameter is `Action` and its value is `DescribeInstances`; therefore, the parameter will be formatted into `Action=DescribeInstances`.

Note:

The `value` is the original value instead of the URL-encoded value.

Then, concatenate the formatted parameters with `&`. The generated request string will be as follows:

```
Action=DescribeInstances&InstanceIds.0=ins-09dx96dg&Limit=20&Nonce=11886&Offset=0&R
```

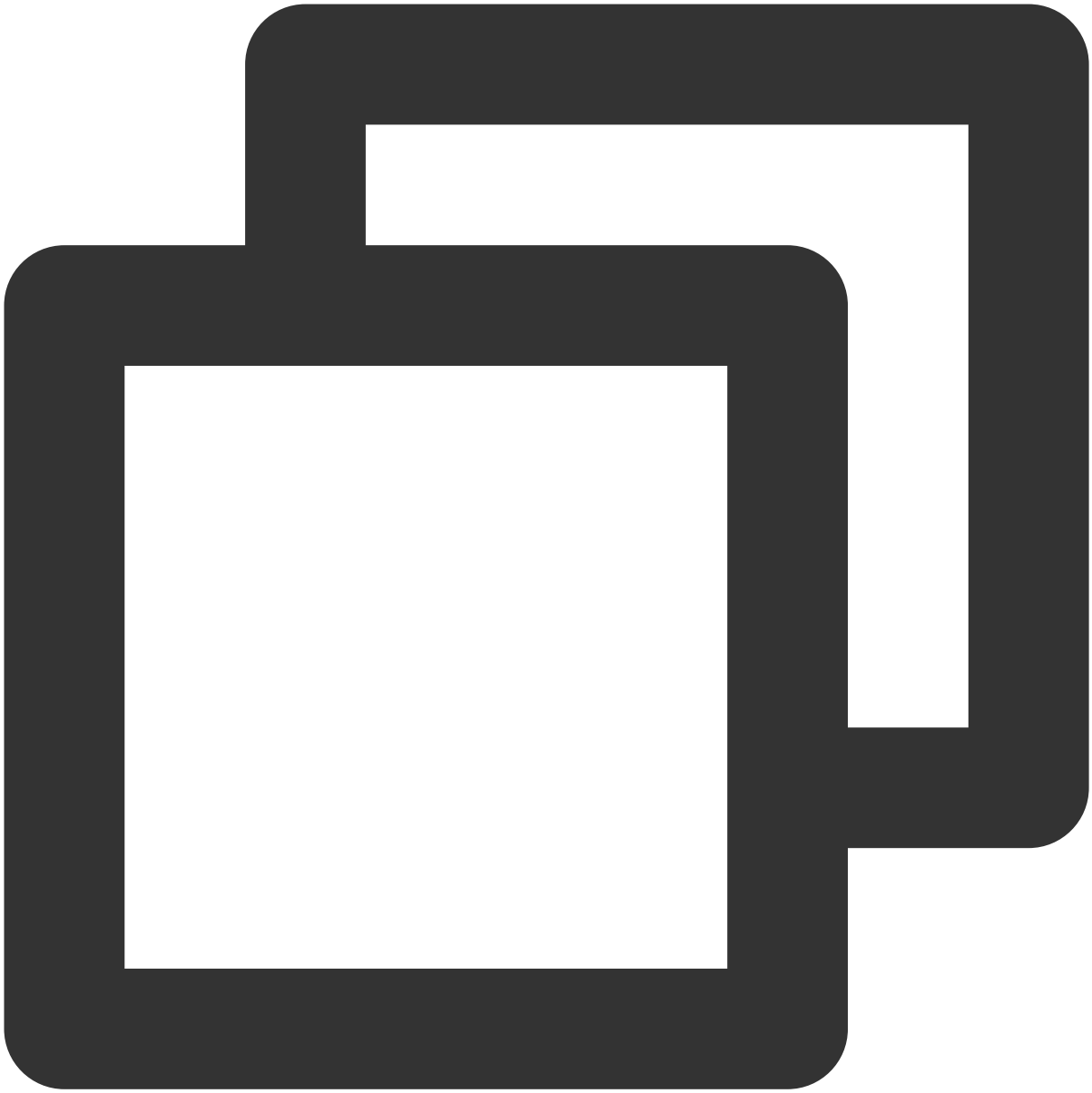
3. Concatenate the string to sign

This step generates the original signature string. The original signature string consists of the following parameters:

1. Request method: POST and GET methods are supported. GET is used here for the request. Please note that the method name should be in all capital letters.
2. Request server: the domain name of the request for querying instances (DescribeInstances) is `cvm.tencentcloudapi.com` . The actual request domain name varies by the module to which the API belongs. For more information, please see the specific API document.
3. Request path: the request path in the current version of TencentCloud API is fixed to `/` .
4. Request string: the request string generated in the previous step.

The rule for concatenating the original string of the signature is `request method + request server + request path + ? + request string` .

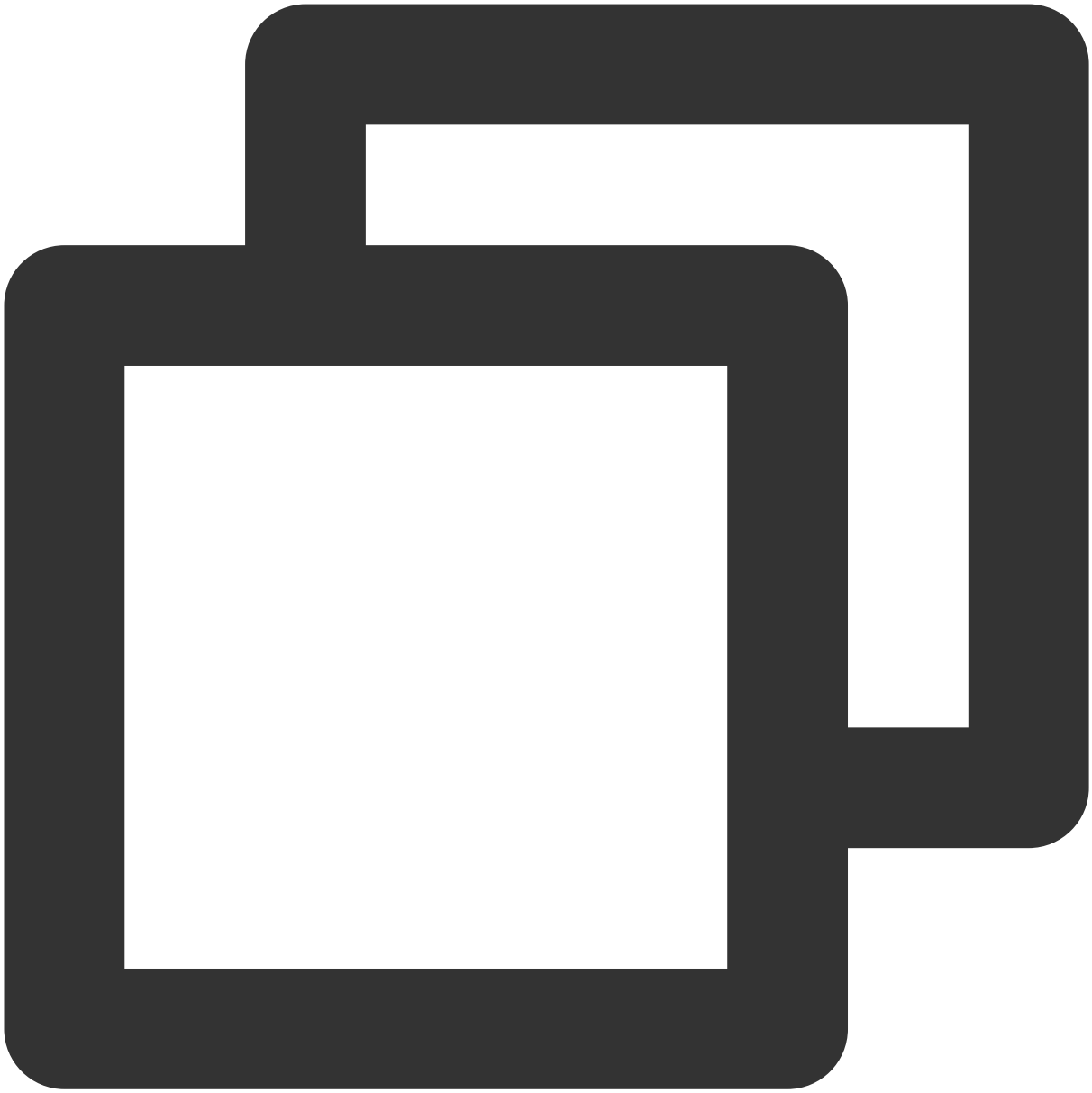
The concatenation result in the example is as follows:



```
GETcvm.tencentcloudapi.com/?Action=DescribeInstances&InstanceIds.0=ins-09dx96dg&Lim
```

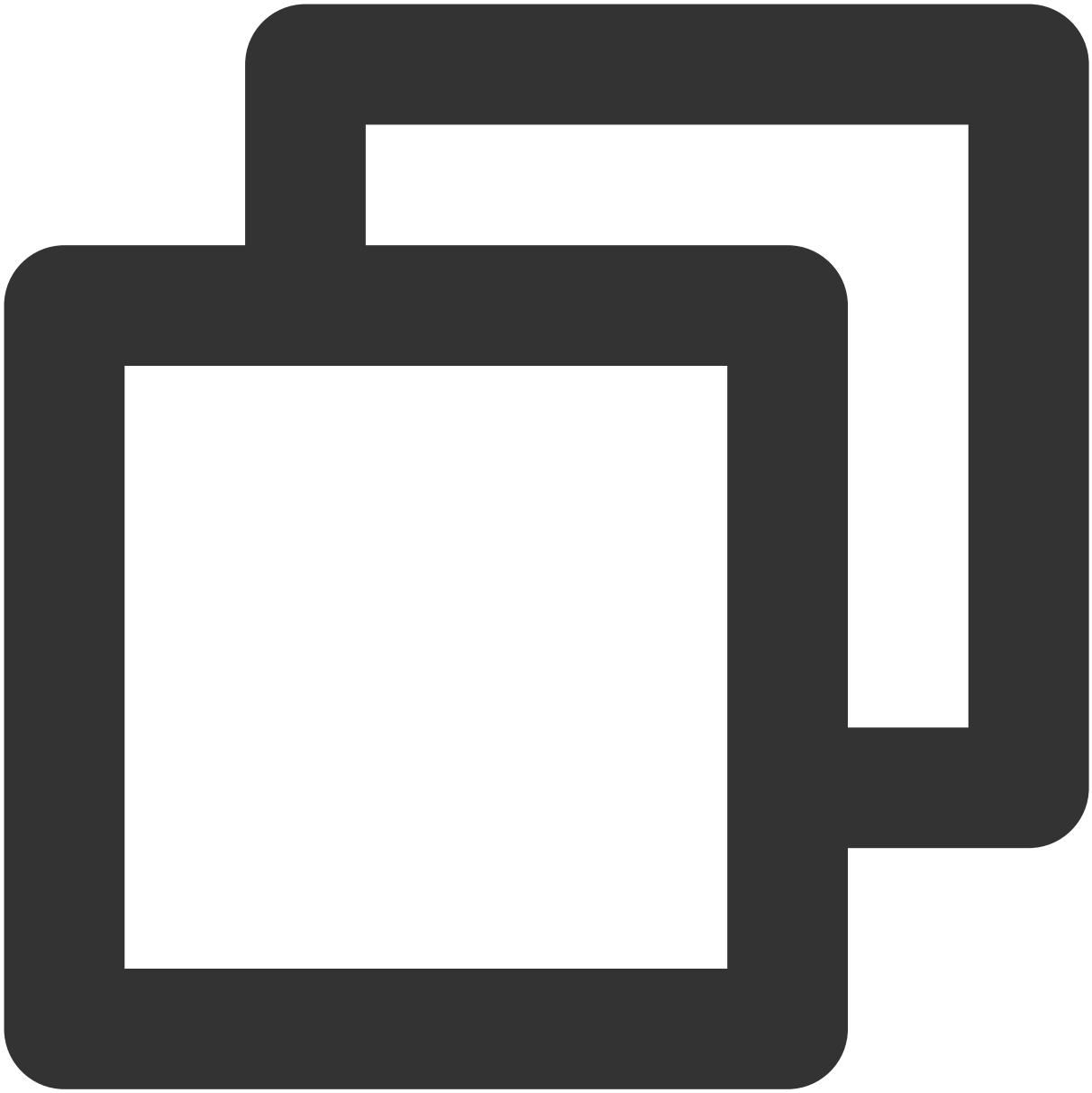
4. Calculate the signature (pseudocode)

This step generates a signature string. Use the HMAC-SHA1 algorithm to sign the **original signature string** obtained in the previous step, and then Base64-encode the generated signature to get the signature.

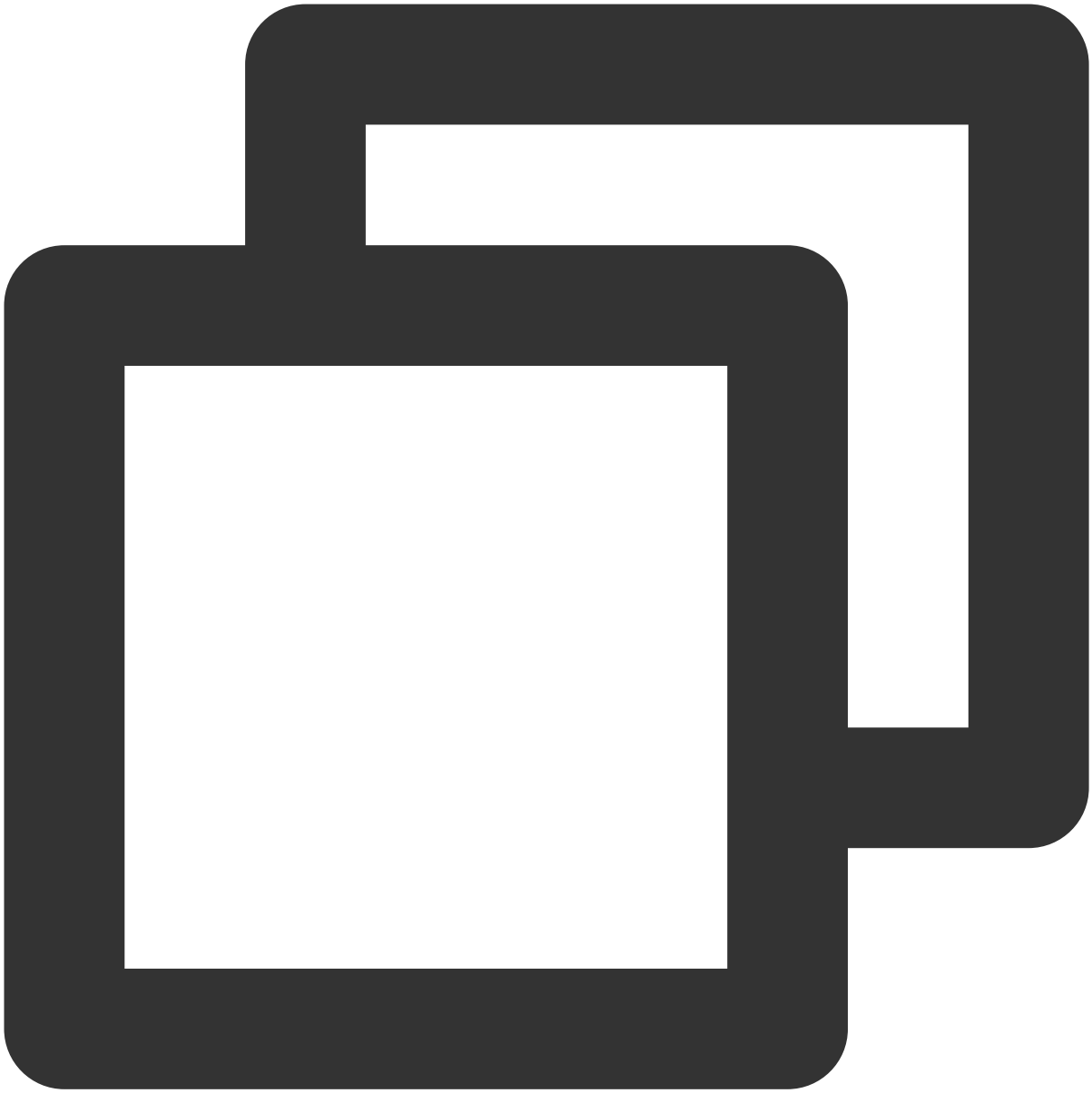


```
hashed := hmac.New(sha1.New, []byte(secretKey))
hashed.Write(buf.Bytes())

fmt.Println(base64.StdEncoding.EncodeToString(hashed.Sum(nil)))
```

5. Get the call information and send a request



```
# The API will be called actually, and fees will be incurred if it is a consumption
resp = requests.get("https://" + endpoint, params=data)
print(resp.url)
```



```
The obtained request string is as follows:
https://cvm.tencentcloudapi.com/?Nonce=11886&SecretId=AKIDz8krbsJ5**********mLPx3EX
```

Field	Description
endpoint	Service address, such as `cvm.tencentcloudapi.com` .
data	API parameter of the sample API 3.0 signature v1. **Note:** you should add the calculated signature in the format of key-value pair to `data` .

Note:

The key in the example is not real, and the timestamp is not the current system time. If you open this URL in the browser or call it by using commands such as `curl` , an authentication error `The signature expired` will be returned. To obtain a URL that works, you need to replace the `SecretId` and `SecretKey` in this example with your own credentials and use the current system time as the `Timestamp` .

To further explain the signing process, Go is used as examples below to implement the process as described above. The request domain name, API, and parameter values in the above example are used here. The code below is for demonstration only. Please use the SDK for actual development.

6. Encode a signature string

The generated signature string cannot be directly used as a request parameter and needs to be URL-encoded. For example, if the signature string generated in the previous step is `Eli**********cGeI=` , the final value of the `Signature` request parameter will be `EliP**********eI%3D` , which will be used to generate the final request URL.

Note:

If you use the GET request method or use the POST request method with `Content-Type` of `application/x-www-form-urlencoded` , all the request parameter values must be URL-encoded (except the parameter key and the equal symbol (=)) before the request is sent. Non-ASCII characters must be encoded with UTF-8 before URL-encoding.

The network libraries of some programming languages automatically URL-encode all parameters. In this case, the signature string does not need to be URL-encoded again; otherwise, two rounds of URL-encoding will cause the signature to fail.

Other parameter values also need to be encoded with RFC 3986. Use `%XY` in percent-encoding for special characters such as Chinese characters, where `X` and `Y` are hexadecimal characters (0–9 and uppercase A–F). Using lowercase characters will cause an error.

7. Sample API 3.0 signature v1

Note:

Only the signature calculation process is described here. If you want to get the final request string, you need to add the signature key-value obtained below to `params` .

```
package main

import (
    "bytes"
    "crypto/hmac"
    "crypto/sha1"
    "encoding/base64"
    "net/url"
    "fmt"
    "sort"
)
```

```
func main() {
    secretId := "AKIDz8krbsJ5**********mLPx3EXAMPLE"
    secretKey := "Gu5t9xGAR***********EXAMPLE"
    endpoint := "cvm.tencentcloudapi.com"
    params := map[string]string{
        "Nonce":         "11886",
        "Timestamp":     "1465185768",
        "Region":        "ap-guangzhou",
        "SecretId":      secretId,
        "Version":       "2017-03-12",
        "Action":        "DescribeInstances",
        "InstanceIds.0": "ins-09dx96dg",
        "Limit":         "20",
        "Offset":        "0",
    }

    var buf bytes.Buffer
    buf.WriteString("GET")
    buf.WriteString(endpoint)
    buf.WriteString("/")
    buf.WriteString("?")

    // sort keys by ascii asc order
    keys := make([]string, 0, len(params))
    for k, _ := range params {
        keys = append(keys, k)
    }
    sort.Strings(keys)

    for i := range keys {
        k := keys[i]
        buf.WriteString(k)
        buf.WriteString("=")
        buf.WriteString(params[k])
        buf.WriteString("&")
    }
    buf.Truncate(buf.Len() - 1)

    hashed := hmac.New(sha1.New, []byte(secretKey))
    hashed.Write(buf.Bytes())

    signature := base64.StdEncoding.EncodeToString(hashed.Sum(nil))
    fmt.Println(base64.StdEncoding.EncodeToString(hashed.Sum(nil)))
    final_signature := url.QueryEscape(signature)
    fmt.Println(final_signature)
}
```

Signature Failure

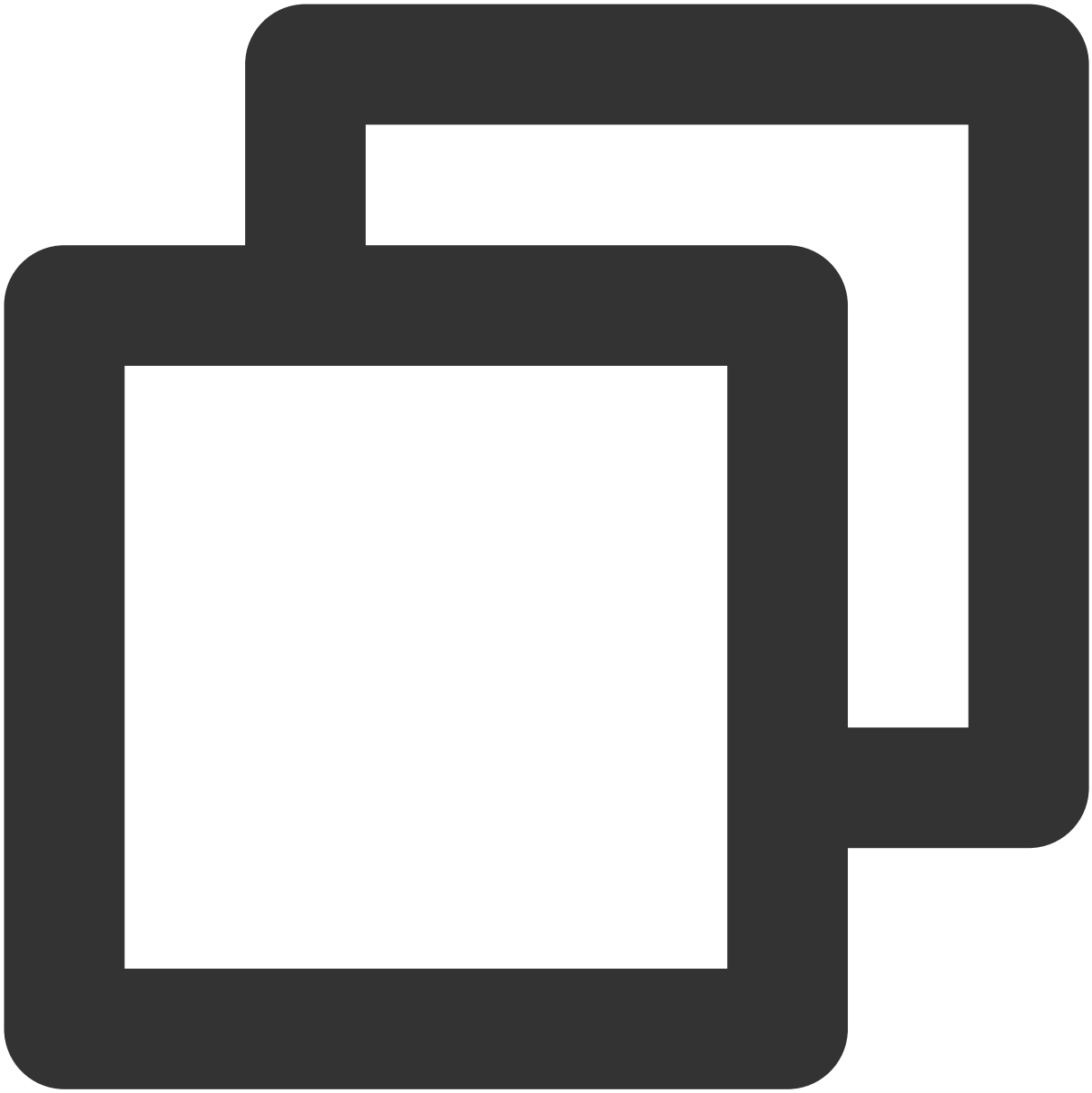
The following error codes may be returned for signature failure. Please resolve the errors accordingly.

Error Code	Error Description
AuthFailure.SignatureExpire	The signature expired. The difference between the `Timestamp` and the server time cannot be greater than five minutes.
AuthFailure.SecretIdNotFound	The key does not exist. Log in to the console and check whether it is disabled or you copied fewer or more characters.
AuthFailure.SignatureFailure	Signature error. It is possible that the signature is calculated incorrectly, the signature does not match the content that is actually sent, or the `SecretKey` is incorrect.
AuthFailure.TokenFailure	Temporary credential token error.
AuthFailure.InvalidSecretId	Invalid key (not TencentCloud API key type).

Returned Result

Successful response

For example, when calling the CVM API `DescribeInstancesStatus` (version: 2017-03-12) to view the status of instances, if the request succeeds, you may see the following response:



```
{
    "Response": {
        "TotalCount": 0,
        "InstanceStatusSet": [],
        "RequestId": "b5b41468-520d-4192-b42f-595cc34b6c1c"
    }
}
```

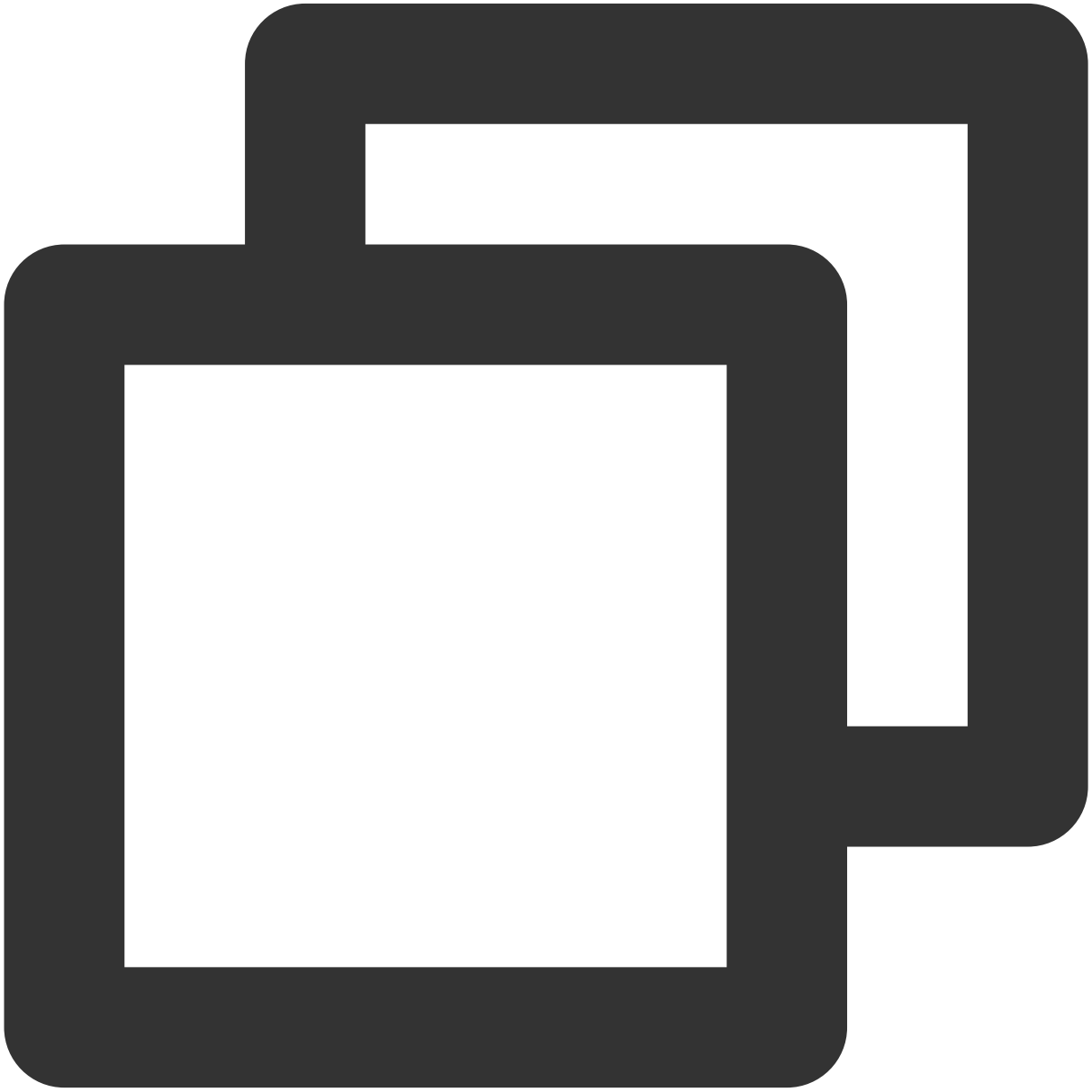
The API will return `Response` , which contains `RequestId` , as long as it processes the request, no matter whether the request is successful or not.

`RequestId` is the unique ID of an API request. It is required to troubleshoot issues.

Any fields other than the common fields are API-specific. For more information on such fields, please see the relevant API documentation. In this example, both `TotalCount` and `InstanceStatusSet` are specific to the `DescribeInstancesStatus` API. Since the user who initiated the request does not have a CVM instance yet, 0 is returned for `TotalCount` and `InstanceStatusSet` is empty.

Error response

If the call fails, you may see the following response:



```
{
```

```
    "Response": {
        "Error": {
            "Code": "AuthFailure.SignatureFailure",
            "Message": "The provided credentials could not be validated. Please che
        },
        "RequestId": "ed93f3cb-f35e-473f-b9f3-0d451b8b79c6"
    }
}
```

`Error` indicates that the request failed. A response for a failed request will always include the `Error`, `Code`, and `Message` fields.

`Code` indicates the specific error code, which is returned when an API request failed. You can use this code to locate the cause and solution of the error in the common or API-specific error code list.

`Message` explains the cause of the error. Note that the returned messages are subject to service updates. The information the messages provide may not be up-to-date and should not be the only source of reference.

`RequestId` is the unique ID of an API request. It is required to troubleshoot issues.

Common error codes

The `Error` field in a response indicates that the API call failed. The `Code` field in `Error` indicates the error code. The following table lists the common error codes that any services may return.

Error Code	Error Description
AuthFailure.InvalidSecretId	Invalid key (not TencentCloud API key type).
AuthFailure.MFAFailure	MFA failure.
AuthFailure.SecretIdNotFound	The key does not exist.
AuthFailure.SignatureExpire	The signature expired.
AuthFailure.SignatureFailure	Signature error.
AuthFailure.TokenFailure	Token error.
AuthFailure.UnauthorizedOperation	No CAM authorization.
DryRunOperation	DryRun Operation. It means that the request would have succeeded, but the DryRun parameter was used.
FailedOperation	The operation failed.
InternalError	Internal error.
InvalidAction	The API does not exist.

InvalidParameter	Incorrect parameter.
InvalidParameterValue	Invalid parameter value.
LimitExceeded	The quota limit is exceeded.
MissingParameter	A parameter is missing.
NoSuchVersion	The API version does not exist.
RequestLimitExceeded	The request rate limit is exceeded.
ResourceInUse	The resource is in use.
ResourceInsufficient	Insufficient resource.
ResourceNotFound	The resource does not exist.
ResourceUnavailable	The resource is unavailable.
UnauthorizedOperation	Unauthorized operation.
UnknownParameter	Unknown parameter error.
UnsupportedOperation	Unsupported operation.
UnsupportedProtocol	Unsupported HTTPS request method. Only GET and POST requests are supported.
UnsupportedRegion	Unsupported region.

API for Java

Last updated：2023-03-07 18:16:40

TencentCloud API has been upgraded to v3.0. This version is optimized for performance and deployed in all regions. It supports nearby access and access by region for significantly reduced access latency. In addition, it features more detailed API descriptions and error codes and API-level comments for SDKs, enabling you to use Tencent Cloud services more conveniently and quickly. This document describes how to call APIs for Java.
This version currently supports various Tencent Cloud services such as CVM, CBS, VPC, and TencentDB and will support more services in the future.

Request Structure

1. Service address (endpoint)

TencentCloud API supports access from either a nearby region (such as `cvm.tencentcloudapi.com` for CVM) or a specified region (such as `cvm.ap-guangzhou.tencentcloudapi.com` for CVM in the Guangzhou region). For values of the region parameter, please see the region list in the "Common Parameters" section below. To check whether a region is supported by a specific Tencent Cloud service, please see its "Request Structure" document.

Note:

For latency-sensitive businesses, we recommend you specify a domain name with a region.

2. Communications protocol

All TencentCloud APIs communicate over HTTPS, providing highly secure communications tunnels.

3. Request method

Supported HTTP request methods:
POST (recommended)
GET
`Content-Type` types supported by POST request:
application/json (recommended). The signature algorithm v3 (TC3-HMAC-SHA256) must be used.
application/x-www-form-urlencoded. The signature algorithm v1 (HmacSHA1 or HmacSHA256) must be used.
multipart/form-data (only supported by certain APIs). The signature algorithm v3 (TC3-HMAC-SHA256) must be used.
The size of a GET request packet cannot exceed 32 KB. The size of a POST request cannot exceed 1 MB for the signature algorithm v1 (HmacSHA1 or HmacSHA256) or 10 MB for the signature algorithm v3 (TC3-HMAC-SHA256).

4. Character encoding

UTF-8 encoding is always used.

Common Parameters

The common parameters are used to identity the user and API signature. They should be carried by each request to initiate properly.

Signature algorithm v3

The signature algorithm v3 (sometimes referred to as "TC3-HMAC-SHA256") is more secure than the signature algorithm v1 (referred to as signature algorithm in certain documents), supports larger request packets and POST JSON format, and has a higher performance. We recommend you use it to calculate signatures. For more information on how to use it, please see below.

Parameter Name	Type	Required	Description
X-TC-Action	String	Yes	Name of the API for the desired operation. For the specific value, please see the description of common parameter `Action` in the input parameters in the related API document. For example, the API for querying CVM instance list is `DescribeInstances` .
X-TC-Region	String	-	Region parameter, which is used to identify the region where the data you want to manipulate resides. For values supported for an API, please see the description of common parameter `Region` in the input parameters in related API documentation. **Note: this parameter is not required for some APIs (which will be indicated in related API documentation) and will not take effect even if it is passed.**
X-TC-Timestamp	Integer	Yes	The current UNIX timestamp that records the time when the API request was initiated, such as 1529223702. **Note: if the difference between the UNIX timestamp and the server time is greater than 5 minutes, a signature expiration error may occur.**
X-TC-Version	String	Yes	Version of the API for the desired operation, such as 2017-03-12 for CVM. For the specific value, please see the description of common parameter `Version` in the input parameters in related API documentation.
Authorization	String	Yes	HTTP authentication request header, such as TC3-HMAC-SHA256 Credential=AKIDEXAMPLE/Date/service/tc3_request, SignedHeaders=content-type;host, Signature=72e494ea8**********a96525168 Here,

			TC3-HMAC-SHA256: signature algorithm, currently fixed as this value. Credential: signature credential. `AKIDEXAMPLE` indicates the `SecretId` . `Date` indicates a UTC date which must match the value of `X-TC-Timestamp` (a common parameter) in UTC format. `service` indicates the name of the service and is generally a domain name prefix; for example, the domain name `cvm.tencentcloudapi.com` means the CVM service, and the value for this service is `cvm` . SignedHeaders: the headers that contain the authentication information. `content-type` and `host` are required. Signature: signature digest. For the calculation process, please see below.
X-TC-Token	String	No	Token used for temporary credentials. It must be used with a temporary key. You can get the temporary key and token by calling a CAM API. No token is required for a long-term key.

Signature algorithm v1

When the signature algorithm v1 (sometimes referred to as "HmacSHA256" or "HmacSHA1") is used, the common parameters should be uniformly placed in the request string.

Parameter Name	Type	Required	Description
Action	String	Yes	Name of the API for the desired operation. For the specific value, please see the description of common parameter `Action` in the input parameters in the related API document. For example, the API for querying CVM instance list is `DescribeInstances` .
Region	String	-	Region parameter, which is used to identify the region where the data you want to manipulate resides. For values supported for an API, please see the description of common parameter `Region` in the input parameters in related API documentation. **Note: this parameter is not required for some APIs (which will be indicated in related API documentation) and will not take effect even if it is passed.**
Timestamp	Integer	Yes	The current UNIX timestamp that records the time when the API request was initiated, such as 1529223702. If the difference between the UNIX timestamp and the current time is too large, a signature expiration error may occur.
Nonce	Integer	Yes	A random positive integer used in conjunction with

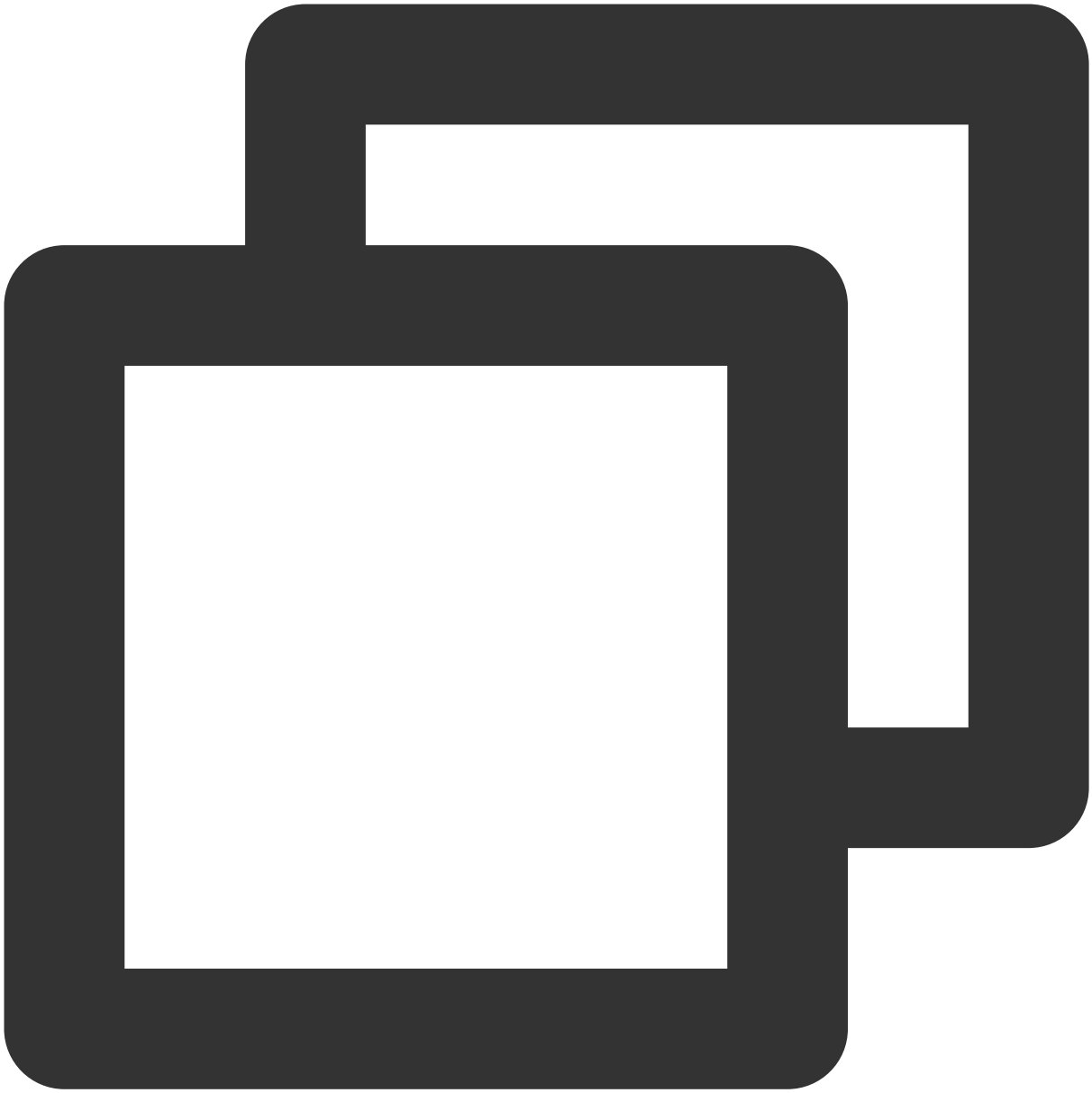
			`Timestamp` to prevent replay attacks.
SecretId	String	Yes	The identifying `SecretId` obtained on the TencentCloud API Key page. A `SecretId` corresponds to a unique `SecretKey` which is used to generate the request signature (`Signature`).
Signature	String	Yes	Request signature, which is used to verify the validity of the request. It is generated based on input parameters. For more information on how to calculate the signature, please see below.
Version	String	Yes	Version of the API for the desired operation, such as 2017-03-12 for CVM. For the specific value, please see the description of common parameter `Version` in the input parameters in related API documentation.
SignatureMethod	String	No	Signature algorithm. Currently, only HmacSHA256 and HmacSHA1 are supported. The HmacSHA256 algorithm is used to verify the signature only when this parameter is specified as HmacSHA256. In other cases, the signature is verified with HmacSHA1.
Token	String	No	Token used for temporary credentials. It must be used with a temporary key. You can get the temporary key and token by calling a CAM API. No token is required for a long-term key.

Region list

As the supported regions vary by service, please refer to the region list in each service's product documentation for specific details.
For example, you can see the region list of CVM.

API Call Method for Java

TencentCloud API authenticates every request, that is, the request must be signed with the security credentials in the designated steps. Each request must contain the signature information in the common request parameters and be sent in the specified way and format.
Suppose your `SecretId` and `SecretKey` are `AKIDz8krbsJ5**********mLPx3EXAMPL` and `Gu5t9xGAR***********EXAMPLE` , respectively. If you want to view the status of an unnamed instance in the Guangzhou region and have only one data entry returned, the request may be:



```
curl -X POST https://cvm.tencentcloudapi.com \\
-H "Authorization: TC3-HMAC-SHA256 Credential=AKIDz8krbsJ5**********mLPx3EXAMPL/201
-H "Content-Type: application/json; charset=utf-8" \\
-H "Host: cvm.tencentcloudapi.com" \\
-H "X-TC-Action: DescribeInstances" \\
-H "X-TC-Timestamp: 1551113065" \\
-H "X-TC-Version: 2017-03-12" \\
-H "X-TC-Region: ap-guangzhou" \\
-d '{"Limit": 1, "Filters": [{"Values": ["\\u672a\\u547d\\u540d"], "Name": "instanc
```

Step 1. Apply for security credentials

In this document, the security credential used is a key pair, which consists of a `SecretId` and a `SecretKey`.

Each user can have up to two key pairs.

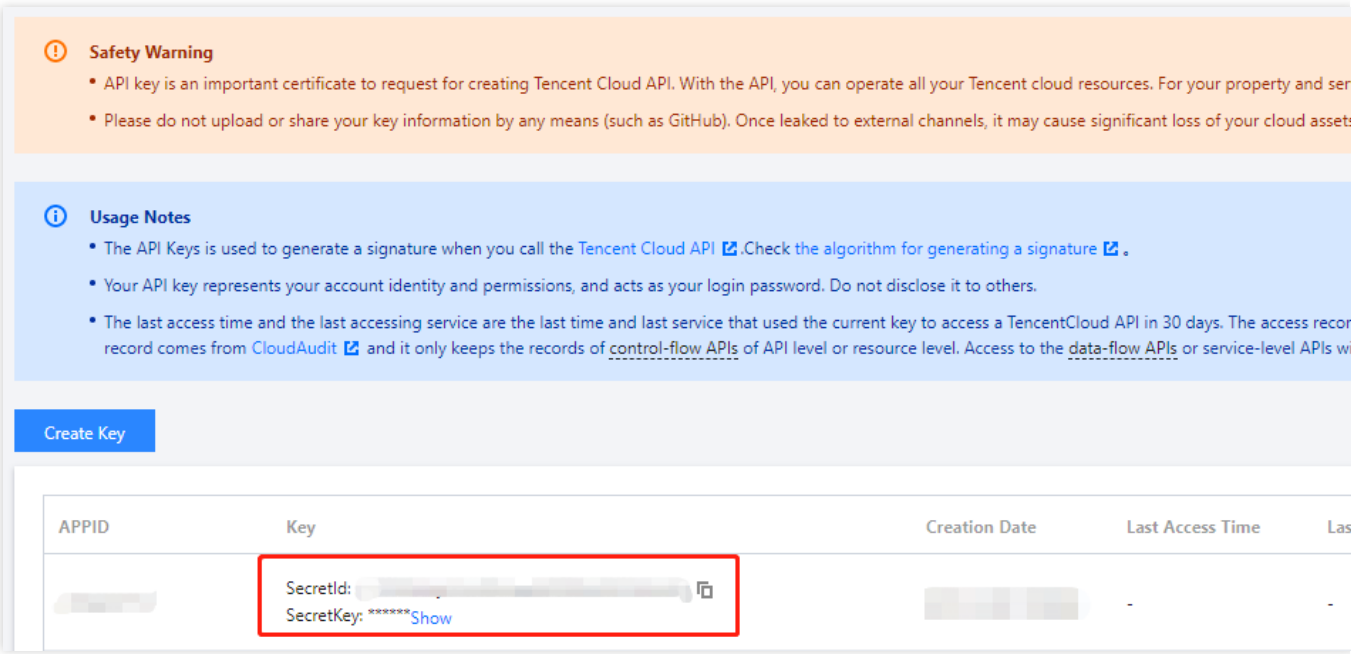
SecretId: identifies the user that calls an API, which is similar to a username.

SecretKey: authenticates the user that calls the API, which is similar to a password.

Note:

You must keep your security credentials private and avoid disclosure; otherwise, your assets may be compromised. If they are disclosed, please disable them as soon as possible.

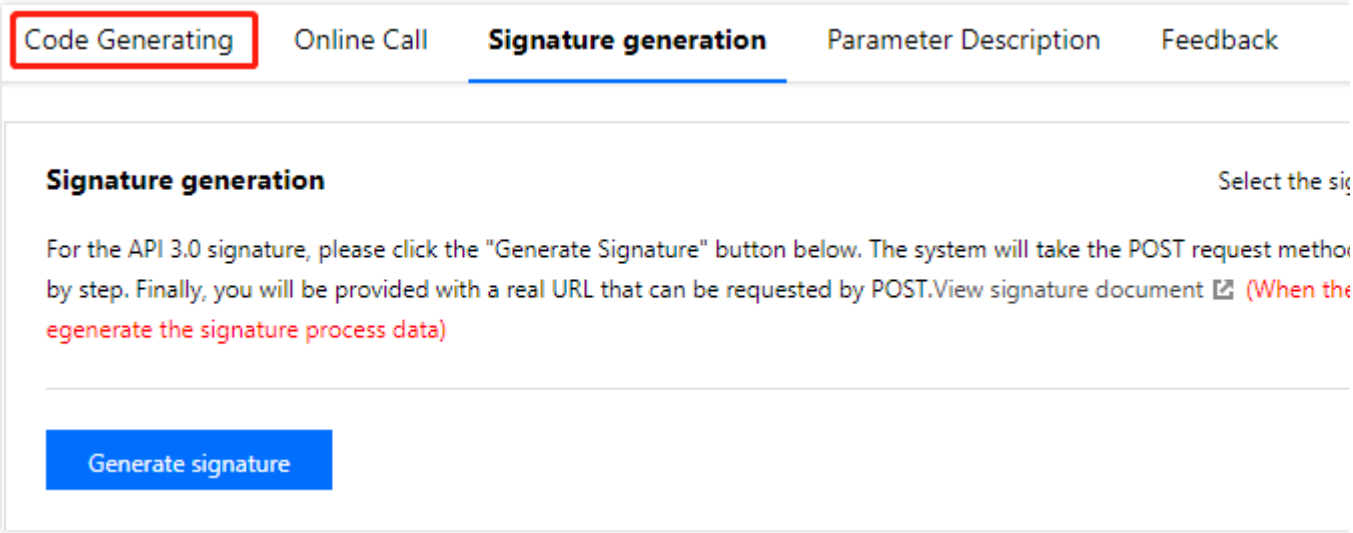
Go to the API key management page to get API keys as shown below:



Step 2

1. Get an API 3.0 signature v3

The signature algorithm v3 (TC3-HMAC-SHA256) is compatible with the previous signature algorithm v1 and more secure, supports larger request packets and POST JSON format, and has a higher performance. We recommend you use it to calculate signatures.



Note:

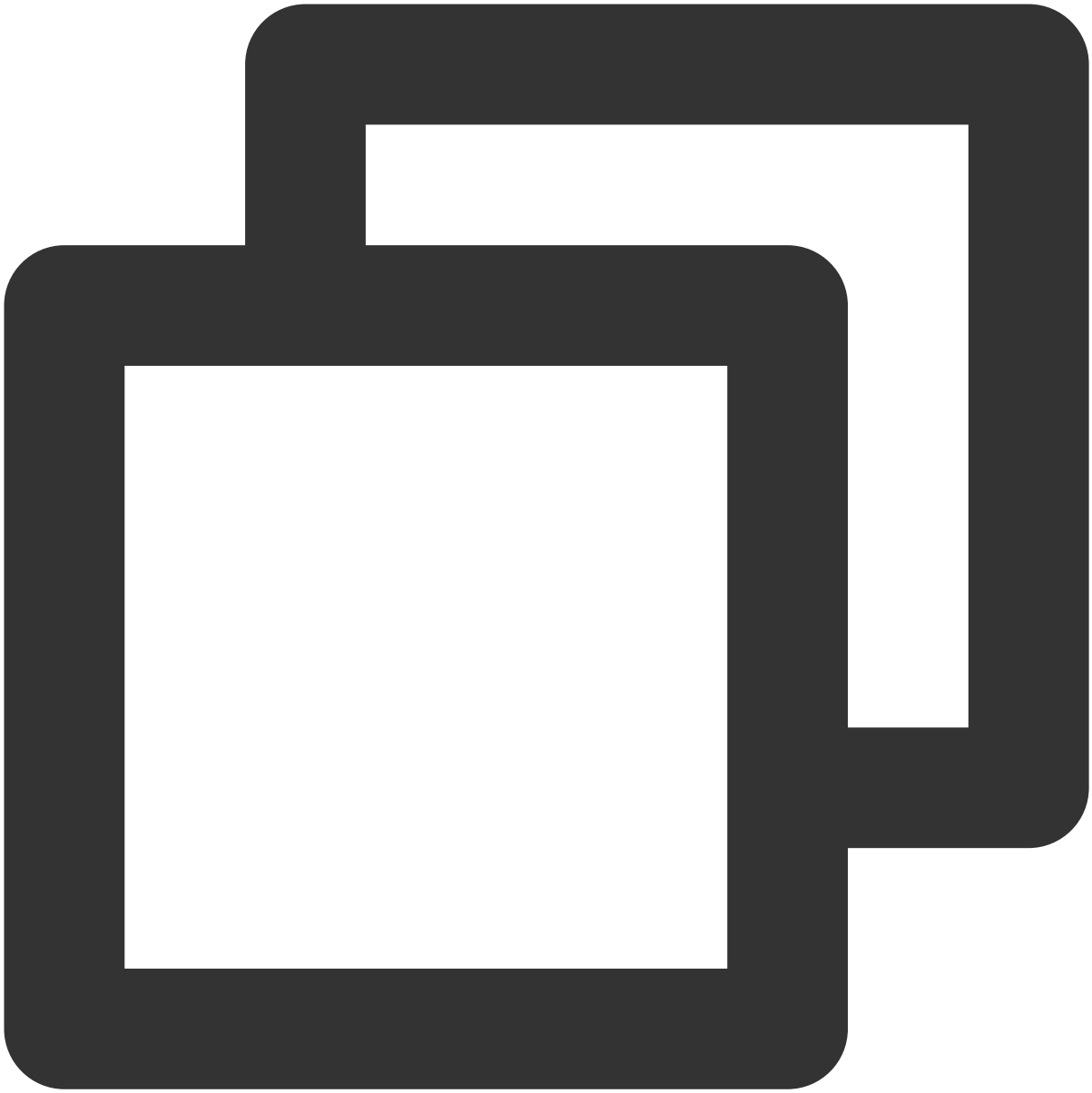
If you are using the signature algorithm for the first time, we recommend you use the "signature string generation" feature in API Explorer and select "API 3.0 signature v3" as the signature version, which can generate a signature for demonstration and verification. Plus, it can also generate SDK code directly. Seven common open-source programming language SDKs are available for TencentCloud API, including Python, Java, PHP, Go, Node.js, .NET, and C++.

TencentCloud API supports both GET and POST requests. For the GET method, only the `Content-Type: application/x-www-form-urlencoded` protocol format is supported. For the POST method, `Content-Type: application/json` and `Content-Type: multipart/form-data` are supported. The JSON format is supported by all business APIs, while the multipart format is supported only by specific APIs (in this case, an API cannot be called in JSON format). For more information, please see the specific business API document. We recommend you use the POST method because the two methods generate the same results, but the GET method only supports request packets below 32 KB in size.

The following describes how to calculate a signature by calling the DescribeInstances API. This API is chosen because:

1. The CVM API is enabled by default, and this API is often used.
2. It is read-only and does not change the status of existing resources.
3. It covers many types of parameters so that it is easy to show how to use an array that contains data structures.

1. Concatenate the canonical request string

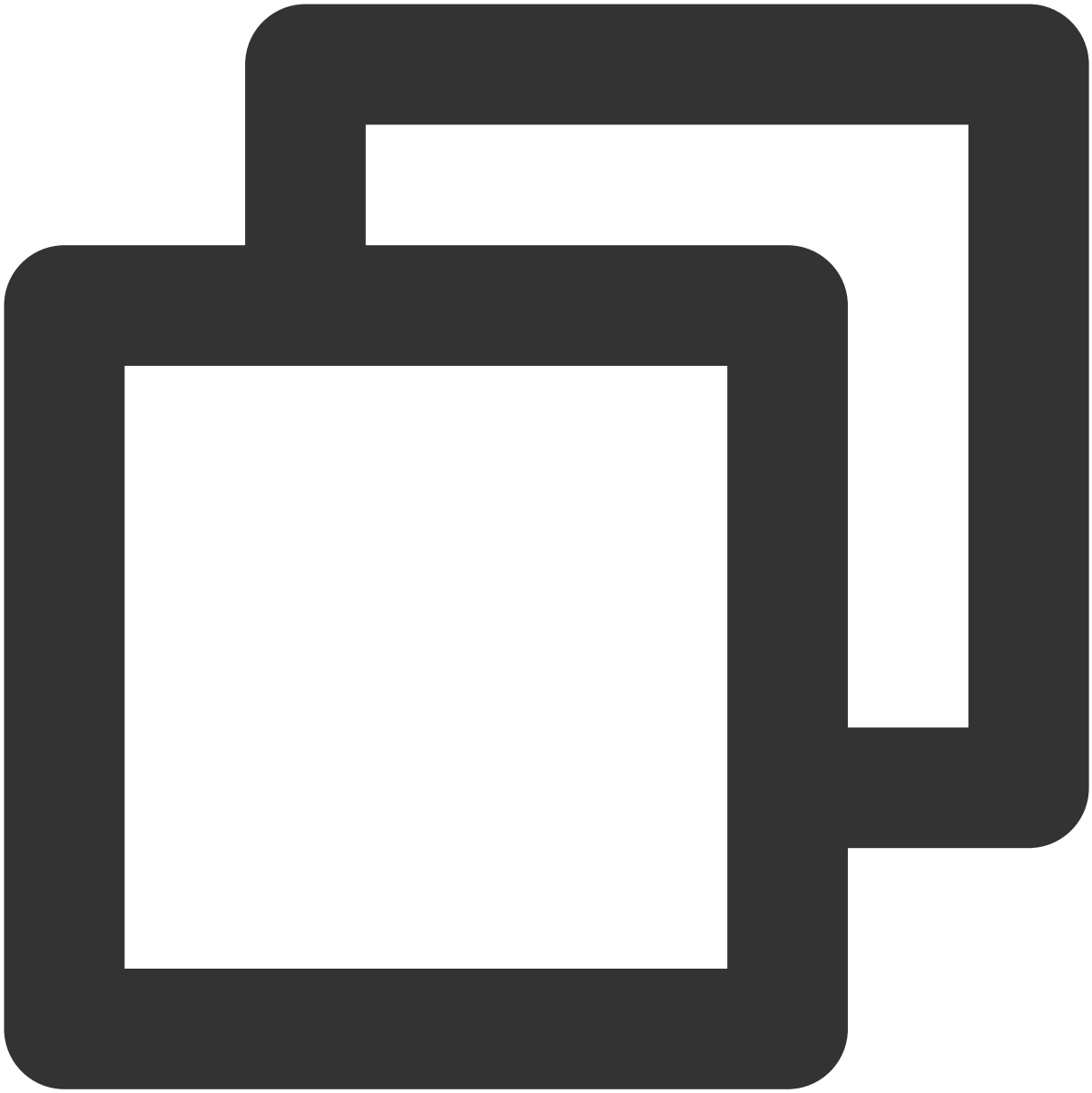


```
CanonicalRequest =
    HTTPRequestMethod + '\\n' +
    CanonicalURI + '\\n' +
    CanonicalQueryString + '\\n' +
    CanonicalHeaders + '\\n' +
    SignedHeaders + '\\n' +
    HashedRequestPayload
```

Field	Description

HTTPRequestMethod	HTTP request method (GET or POST). This example uses `POST` .
CanonicalURI	URI parameter. Slash ("/") is used for API 3.0.
CanonicalQueryString	Query string in the URL of the originating HTTP request. This is always an empty string POST requests and is the string after the question mark (?) for GET requests such as `Limit=10&Offset=0` . Note: `CanonicalQueryString` must be URL-encodec instructed in RFC 3986 with the UTF-8 character set. The applicable standard program language library is recommended. All special characters must be encoded and capitaliz
CanonicalHeaders	Header information for signature calculation, including at least `host` and `content type` . Custom headers can also be added to the signature process to improve the uniqueness and security of the request. Concatenation rules: both the key and value of header should be converted to lowercase with the leading and trailing spaces removed that they are concatenated in the `key:value\\n` format. If there are multiple heade they should be sorted in ASCII ascending order by header key (lowercase). The calcula result in this example is `content-type:application/json; charset=utf-8\\nhost:cvm.tencentcloudapi.com\\n` . **Note: `content-type` must ma the content that is actually sent. In some programming languages, a `charset` value is automatically added even if it is not specified. In this case, the reques sent will be different from the one signed, and the sever will return a signature verification failure.**
SignedHeaders	Header information for signature calculation, indicating the request headers that are inv in the signature process. The request headers must correspond to the headers in `CanonicalHeaders` . `Content-type` and `host` are required headers. Concatenation rules: both the key and value of a header should be converted to lowerca there are multiple headers, they should be sorted in ASCII ascending order by header k (lowercase) and separated by semicolons (;). The value in this example is `content-type;host` .
HashedRequestPayload	Hash value of `Requestpayload` (i.e., the request body, such as `{"Limit": 1, "Filters": [{"Values": ["\\u672a\\u547d\\u540d"], "Name": "instance-name"}]}` in this example). The pseudo-code for calculation is `Lowercase(HexEncode(Hash.SHA256(RequestPayload)))` , which means t SHA256 hashing is performed on the payload of the HTTP request, then hexadecimal encoding is performed, and finally the encoded string is converted to lowercase letters. GET requests, `RequestPayload` is always an empty string. The calculation result i example is `35e9c5b0e3ae67532d3c9f17ead6c90222632e5b1ff7f6e89887f1398934f(`

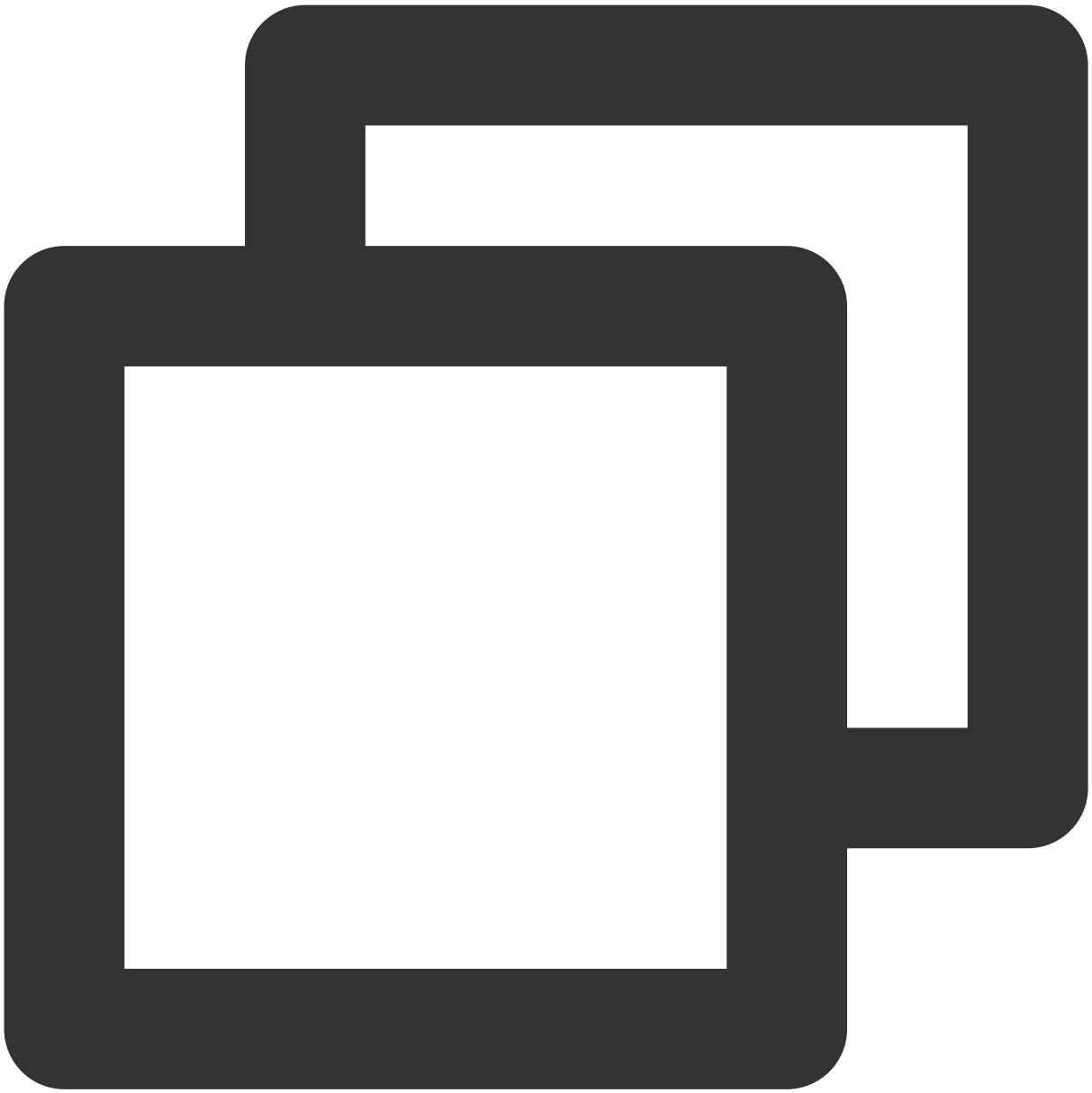
According to the rules above, the canonical request string obtained in the example is as follows:



```
POST
/

content-type:application/json; charset=utf-8
host:cvm.tencentcloudapi.com

content-type;host
35e9c5b0e3ae67532d3c9f17ead6c90222632e5b1ff7f6e89887f1398934f064
```

2. Concatenate the string to sign

Concatenate the string to sign in the following format:



```
StringToSign =
    Algorithm + \\n +
    RequestTimestamp + \\n +
    CredentialScope + \\n +
    HashedCanonicalRequest
```

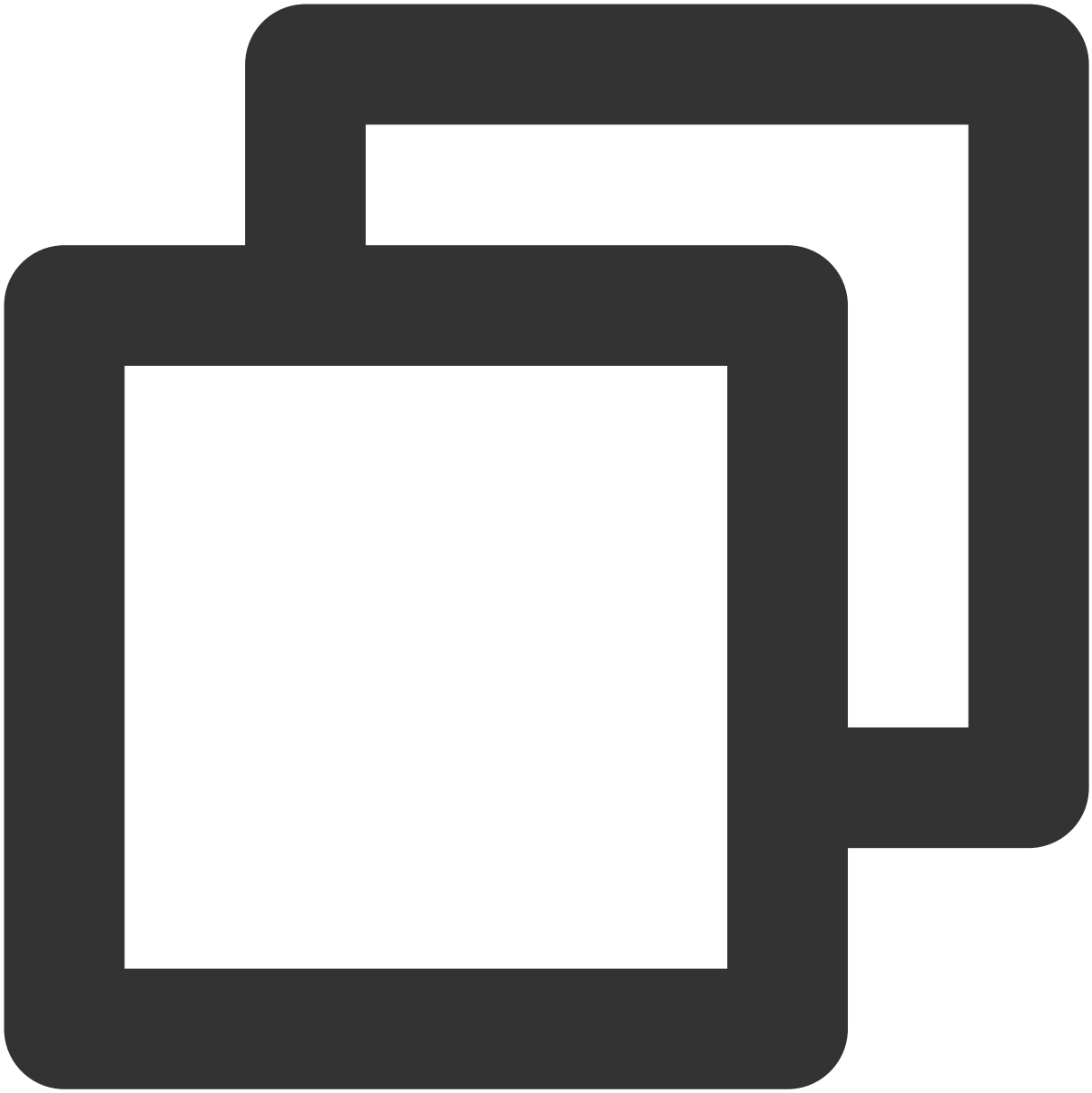
Field	Description
Algorithm	Signature algorithm, which is always `TC3-HMAC-SHA256` currently.

RequestTimestamp	Request timestamp, i.e., the value of the common parameter `X-TC-Timestamp` in request header. It is the UNIX timestamp of the current time in seconds, such as `1551113065` in this example.
CredentialScope	Scope of the credential in the format of `Date/service/tc3_request`, including date, requested service, and termination string (tc3_request). **`Date` indicates a U date, which should match the UTC date converted by the common parameter `TC-Timestamp`.** `service` is the service name, which should match the domain n of the service called. The calculation result in this example is `2019-02-25/cvm/tc3_request`.
HashedCanonicalRequest	Hash value of the canonical request string concatenated in the steps above. The pseu code for calculation is `Lowercase(HexEncode(Hash.SHA256(CanonicalRequest)))`. The calcu result in this example is `5ffe6a04c0664d6b969fab9a13bdab201d63ee709638e2749d62a09ca18d`

Note:

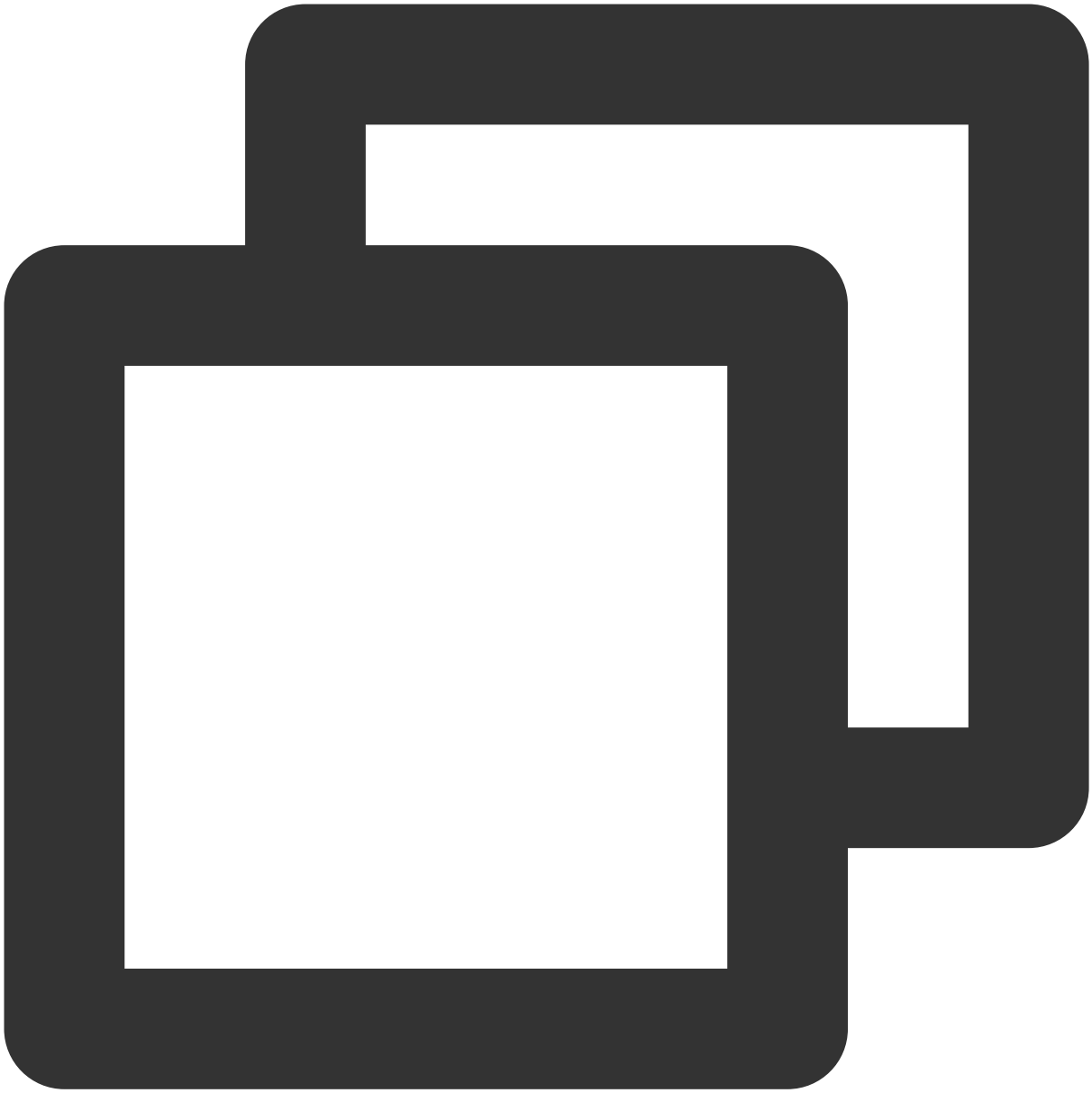
1. `Date` must be calculated from the timestamp `X-TC-Timestamp` and the time zone is UTC+0. If you add the local time zone information (such as UTC+8) in the system, calls can succeed both day and night but will definitely fail at 00:00. For example, if the timestamp is 1551113065 and the time in UTC+8 is 2019-02-26 00:44:25, the UTC+0 date in the calculated `Date` value should be 2019-02-25 instead of 2019-02-26.
2. Timestamp must be the same as your current system time, and your system time must be in sync with the UTC time. If the difference between the timestamp and your current system time is greater than five minutes, the request will fail. If your system time is out of sync with the UTC time for a prolonged period, the request will fail, and a signature expiration error will be returned.

According to the rules above, the string to sign obtained in the example is as follows:



```
TC3-HMAC-SHA256
1551113065
2019-02-25/cvm/tc3_request
5ffe6a04c0664d6b969fab9a13bdab201d63ee709638e2749d62a09ca18d7031
```

3. Calculate the signature (pseudocode)



```
SecretKey = "Gu5t9xGAR***********EXAMPLE"
byte[] secretDate = hmac256(("TC3" + SecretKey).getBytes(UTF8), date);
byte[] secretService = hmac256(secretDate, service);
byte[] secretSigning = hmac256(secretService, "tc3_request");
String signature = DatatypeConverter.printHexBinary(hmac256(secretSigning, stringTo
System.out.println(signature);
```

The derived key `SecretDate`, `SecretService`, and `SecretSigning` are binary data and may contain non-printable characters. Intermediate results are not displayed here.

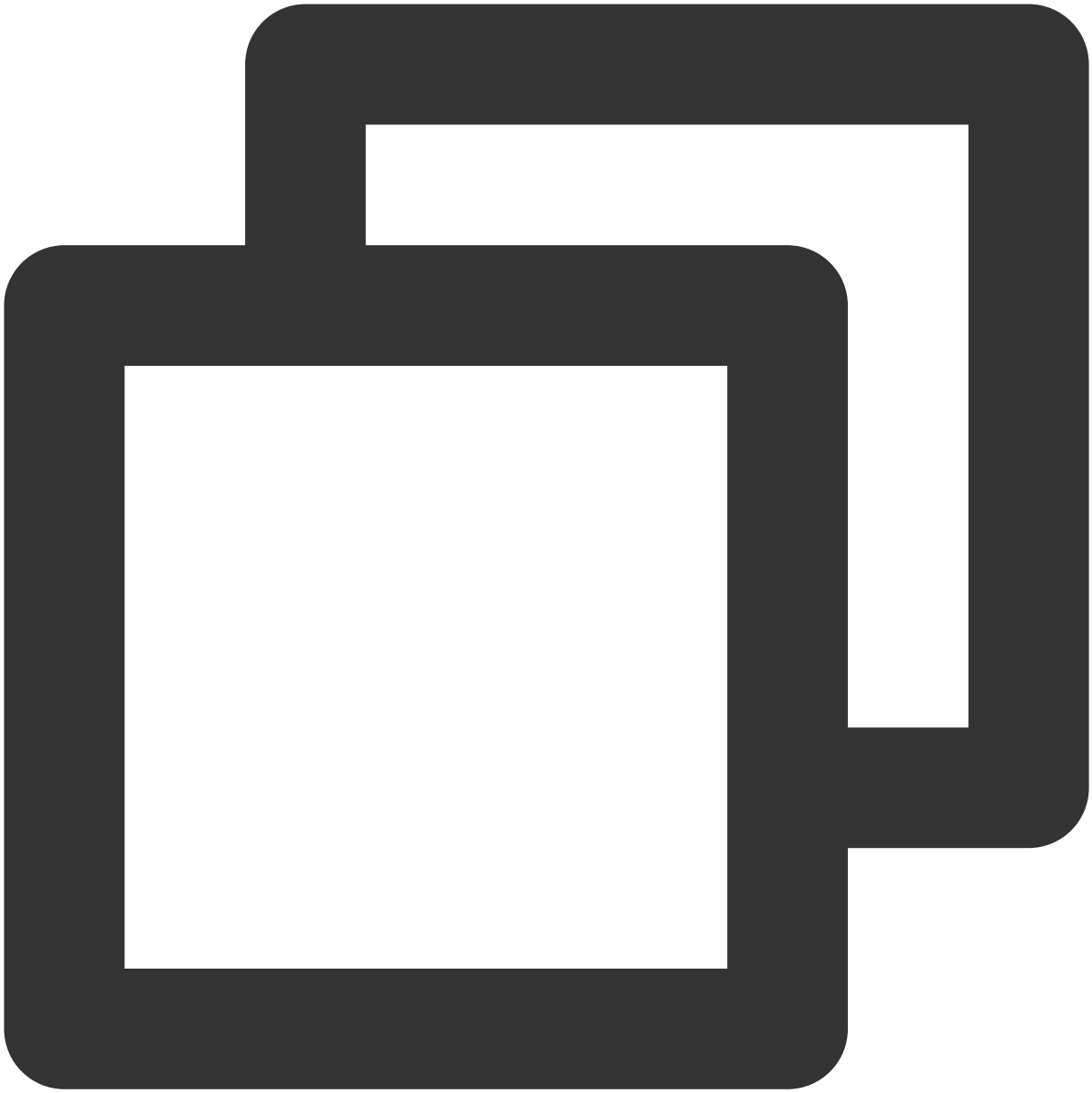
Field	Description

SecretKey	Original `SecretKey` , i.e., `Gu5t9xGAR**********EXAMPLE` .
date	Value of the `Date` field in `Credential` , such as `2019-02-25` in this example.
service	Value of the `Service` field in `Credential` , such as `cvm` in this example.

The calculation result in this example is `72e494ea8****************************************a96525168` .

4. Concatenate the Authorization string

Concatenate the `Authorization` string in the following format:

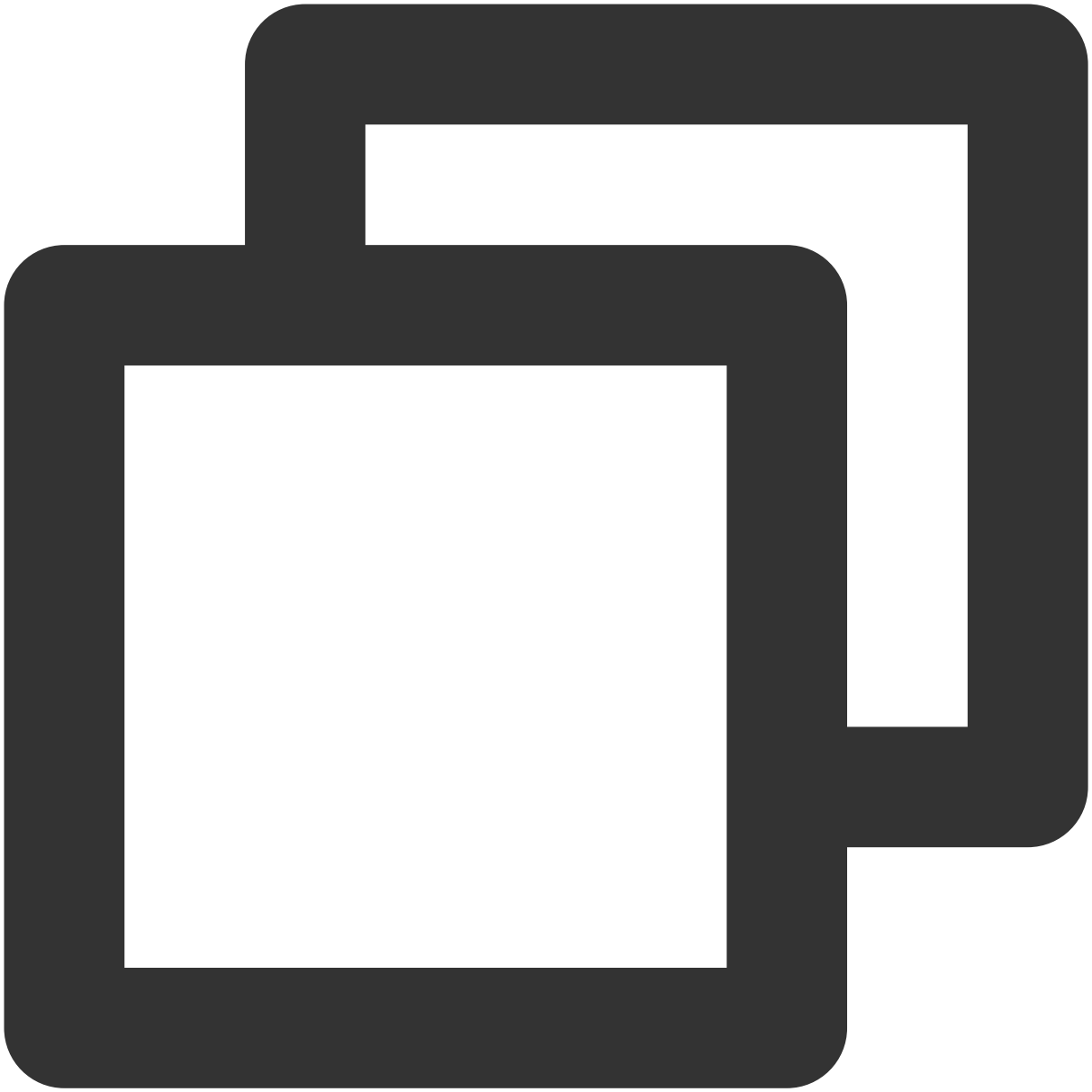


```
String Authorization =
    Algorithm + ' ' +
    'Credential=' + SecretId + '/' + CredentialScope + ', ' +
    'SignedHeaders=' + SignedHeaders + ', ' +
    'Signature=' + Signature
```

Field	Description
Algorithm	Signature algorithm, which is always `TC3-HMAC-SHA256` .
SecretId	

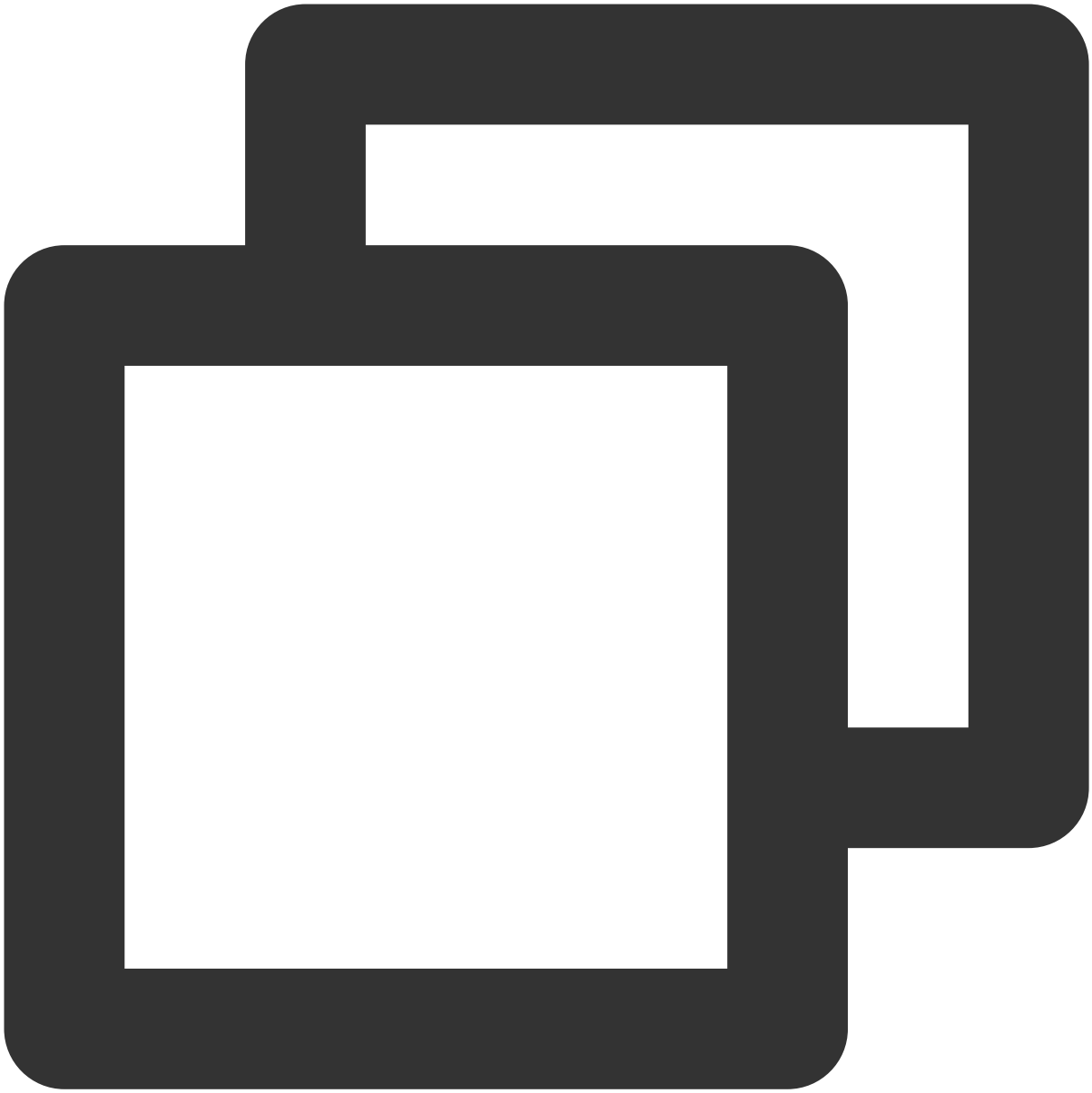
	`SecretId` in the key pair, i.e., `AKIDz8krbsJ5*********mLPx3EXAMPL` .
CredentialScope	Credential scope (see above). The calculation result in this example is `2019-02-25/cvm/tc3_request` .
SignedHeaders	Header information for signature calculation (see above), such as `content-type;host` in this example.
Signature	Signature value. The calculation result in this example is `72e494ea8****************************************a96525168` .

According to the rules above, the values obtained in this example are:



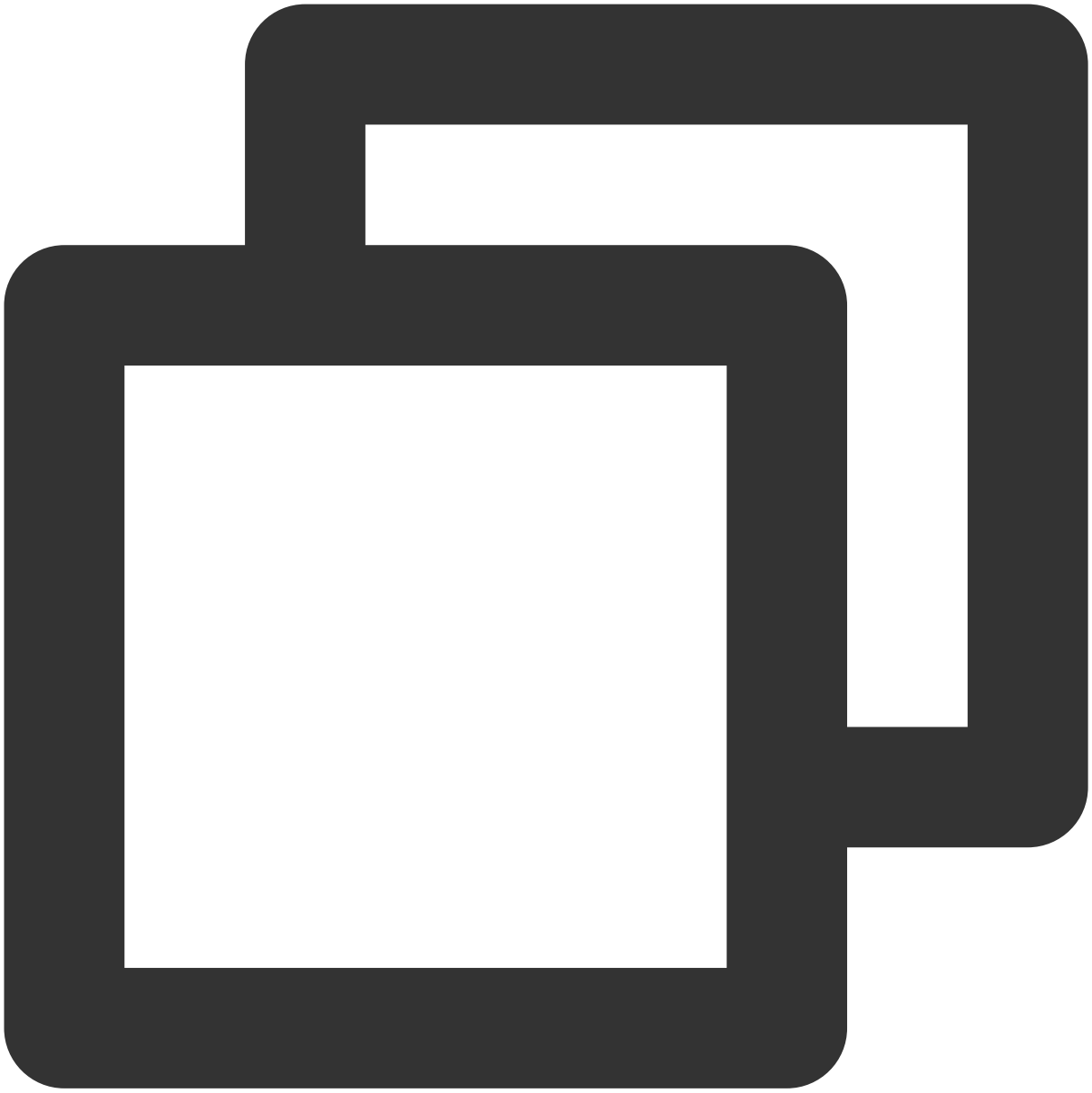
```
TC3-HMAC-SHA256 Credential=AKIDz8krbsJ5**********mLPx3EXAMPL/2019-02-25/cvm/tc3_req
```

The complete call information is as follows:



```
POST https://cvm.tencentcloudapi.com/
Authorization: TC3-HMAC-SHA256 Credential=AKIDz8krbsJ5**********mLPx3EXAMPL/2019-02
Content-Type: application/json; charset=utf-8
Host: cvm.tencentcloudapi.com
X-TC-Action: DescribeInstances
X-TC-Version: 2017-03-12
X-TC-Timestamp: 1551113065
X-TC-Region: ap-guangzhou

{"Limit": 1, "Filters": [{"Values": ["\\u672a\\u547d\\u540d"], "Name": "instance-na
```

5. Sample API 3.0 signature v3



```
import java.nio.charset.Charset;
import java.nio.charset.StandardCharsets;
import java.security.MessageDigest;
import java.text.SimpleDateFormat;
import java.util.Date;
import java.util.TimeZone;
import java.util.TreeMap;
import javax.crypto.Mac;
import javax.crypto.spec.SecretKeySpec;
```

```
import javax.xml.bind.DatatypeConverter;

public class TencentCloudAPITC3Demo {
    private final static Charset UTF8 = StandardCharsets.UTF_8;
    private final static String SECRET_ID = "AKIDz8krbsJ5**********mLPx3EXAMPL";
    private final static String SECRET_KEY = "Gu5t9xGAR***********EXAMPLE";
    private final static String CT_JSON = "application/json; charset=utf-8";

    public static byte[] hmac256(byte[] key, String msg) throws Exception {
        Mac mac = Mac.getInstance("HmacSHA256");
        SecretKeySpec secretKeySpec = new SecretKeySpec(key, mac.getAlgorithm());
        mac.init(secretKeySpec);
        return mac.doFinal(msg.getBytes(UTF8));
    }

    public static String sha256Hex(String s) throws Exception {
        MessageDigest md = MessageDigest.getInstance("SHA-256");
        byte[] d = md.digest(s.getBytes(UTF8));
        return DatatypeConverter.printHexBinary(d).toLowerCase();
    }

    public static void main(String[] args) throws Exception {
        String service = "cvm";
        String host = "cvm.tencentcloudapi.com";
        String region = "ap-guangzhou";
        String action = "DescribeInstances";
        String version = "2017-03-12";
        String algorithm = "TC3-HMAC-SHA256";
        String timestamp = "1551113065";
        //String timestamp = String.valueOf(System.currentTimeMillis() / 1000);
        SimpleDateFormat sdf = new SimpleDateFormat("yyyy-MM-dd");
        // Make sure that the time zone is correct
        sdf.setTimeZone(TimeZone.getTimeZone("UTC"));
        String date = sdf.format(new Date(Long.valueOf(timestamp + "000")));

        // ************* Step 1. Concatenate the canonical request string *********
        String httpRequestMethod = "POST";
        String canonicalUri = "/";
        String canonicalQueryString = "";
        String canonicalHeaders = "content-type:application/json; charset=utf-8\\n"
        String signedHeaders = "content-type;host";

        String payload = "{\\"Limit\\": 1, \\"Filters\\": [{\\"Values\\": [\\"\\\\u
        String hashedRequestPayload = sha256Hex(payload);
        String canonicalRequest = httpRequestMethod + "\\n" + canonicalUri + "\\n"
                + canonicalHeaders + "\\n" + signedHeaders + "\\n" + hashedRequestP
        System.out.println(canonicalRequest);
```

```
        // ************* Step 2. Concatenate the string to sign *************
        String credentialScope = date + "/" + service + "/" + "tc3_request";
        String hashedCanonicalRequest = sha256Hex(canonicalRequest);
        String stringToSign = algorithm + "\\n" + timestamp + "\\n" + credentialSco
        System.out.println(stringToSign);

        // ************* Step 3. Calculate the signature *************
        byte[] secretDate = hmac256(("TC3" + SECRET_KEY).getBytes(UTF8), date);
        byte[] secretService = hmac256(secretDate, service);
        byte[] secretSigning = hmac256(secretService, "tc3_request");
        String signature = DatatypeConverter.printHexBinary(hmac256(secretSigning,
        System.out.println(signature);

        // ************* Step 4. Concatenate the `Authorization` string ***********
        String authorization = algorithm + " " + "Credential=" + SECRET_ID + "/" +
                + "SignedHeaders=" + signedHeaders + ", " + "Signature=" + signatur
        System.out.println(authorization);

        TreeMap<String, String> headers = new TreeMap<String, String>();
        headers.put("Authorization", authorization);
        headers.put("Content-Type", CT_JSON);
        headers.put("Host", host);
        headers.put("X-TC-Action", action);
        headers.put("X-TC-Timestamp", timestamp);
        headers.put("X-TC-Version", version);
        headers.put("X-TC-Region", region);

        StringBuilder sb = new StringBuilder();
        sb.append("curl -X POST https://").append(host)
        .append(" -H \\"Authorization: ").append(authorization).append("\\"")
        .append(" -H \\"Content-Type: application/json; charset=utf-8\\"")
        .append(" -H \\"Host: ").append(host).append("\\"")
        .append(" -H \\"X-TC-Action: ").append(action).append("\\"")
        .append(" -H \\"X-TC-Timestamp: ").append(timestamp).append("\\"")
        .append(" -H \\"X-TC-Version: ").append(version).append("\\"")
        .append(" -H \\"X-TC-Region: ").append(region).append("\\"")
        .append(" -d '").append(payload).append("'");
        System.out.println(sb.toString());
    }
}
```

2. Get an API 3.0 signature v1

The signature algorithm v1 is simple and easy to use, but its functionality and security are not as good as the signature algorithm v3 which is therefore recommended.

Note:

If you are using the signature algorithm for the first time, we recommend you use the "signature string generation" feature in API Explorer and select "API 3.0 signature v1" as the signature version, which can generate a signature for demonstration and verification and provides signing examples for certain programming languages. Plus, it can also generate SDK code directly. Seven common open-source programming language SDKs are available for TencentCloud API, including Python, Java, PHP, Go, Node.js, .NET, and C++.

For example, if you call the `DescribeInstances` API to query CVM instances, the request parameters may be as follows:

Parameter Name	Description	Value
Action	Method	DescribeInstances
SecretId	Key ID	`AKIDz8krbsJ5*********mLPx3EXAMPL`
Timestamp	Current timestamp	1465185768
Nonce	Random positive integer	11886
Region	Instance region	ap-guangzhou
InstanceIds.0	ID of the instance to be queried	ins-09dx96dg
Offset	Offset	0
Limit	Allowed maximum number of output entries	20
Version	API version number	2017-03-12

1. Sort parameters

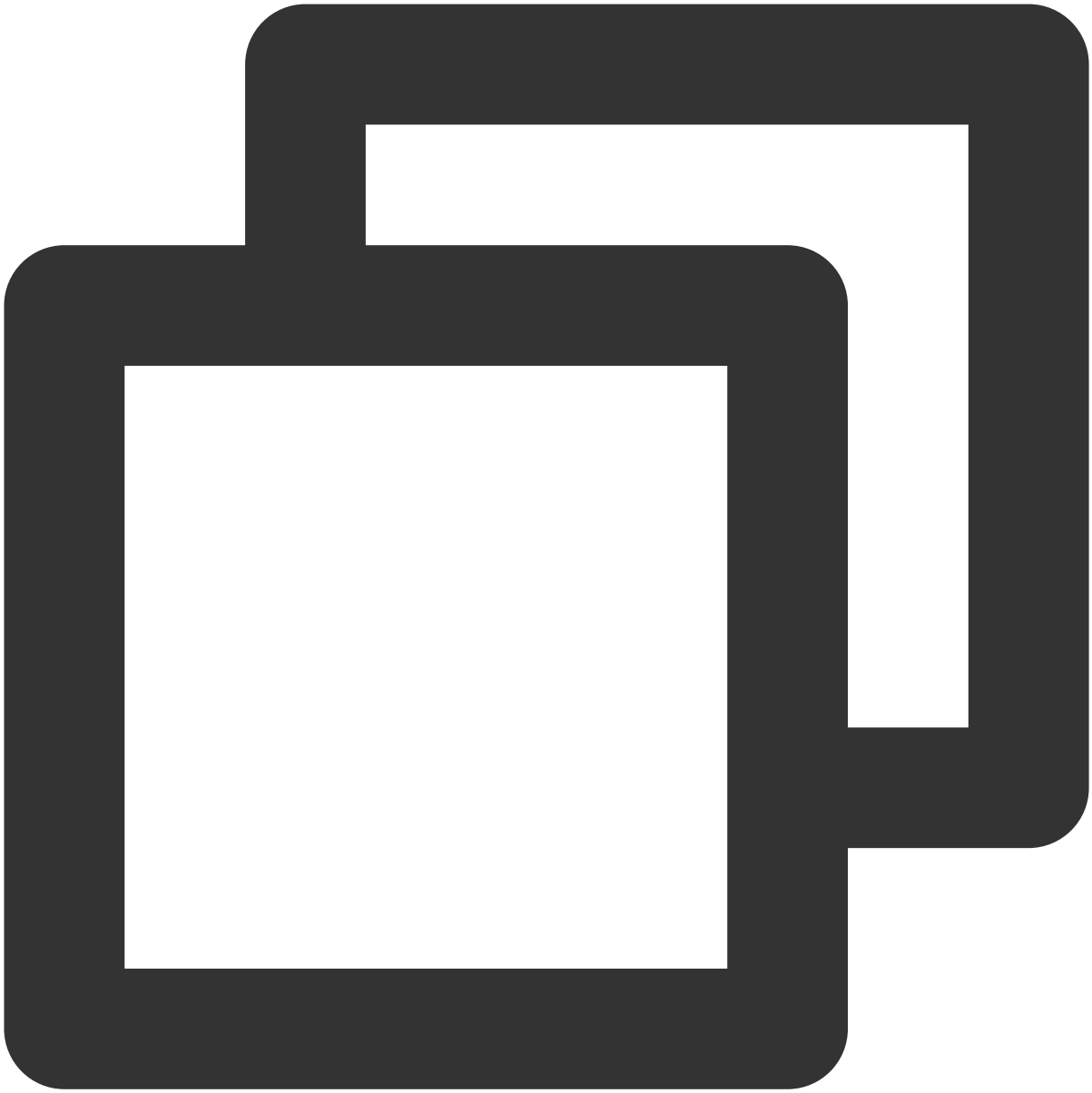
Sort all the request parameters in an ascending lexicographical order (ASCII code) by their names.

Note:

The parameters are sorted only by name but not by value.

The parameters are sorted based on ASCII code but not in an alphabetical order or by value. For example, `InstanceIds.2` should be arranged behind `InstanceIds.12`. You can complete sorting by using a sorting function in a programming language, such as the `ksort` function in PHP.

The parameters in the example are sorted as follows:



```
{
    'Action' : 'DescribeInstances',
    'InstanceIds.0' : 'ins-09dx96dg',
    'Limit' : 20,
    'Nonce' : 11886,
    'Offset' : 0,
    'Region' : 'ap-guangzhou',
    'SecretId' : 'AKIDz8krbsJ5**********mLPx3EXAMPL',
    'Timestamp' : 1465185768,
    'Version': '2017-03-12',
}
```

Any other programming languages can be used to sort these parameters as long as the same result is produced.

2. Concatenate the canonical request string

This step generates a request string. Format the request parameters sorted in the previous step into the form of `parameter=value`. For example, for the `Action` parameter, its parameter is `Action` and its value is `DescribeInstances`; therefore, the parameter will be formatted into `Action=DescribeInstances`.

Note:

The `value` is the original value instead of the URL-encoded value.

Then, concatenate the formatted parameters with `&`. The generated request string will be as follows:

```
Action=DescribeInstances&InstanceIds.0=ins-09dx96dg&Limit=20&Nonce=11886&Offset=0&R
```

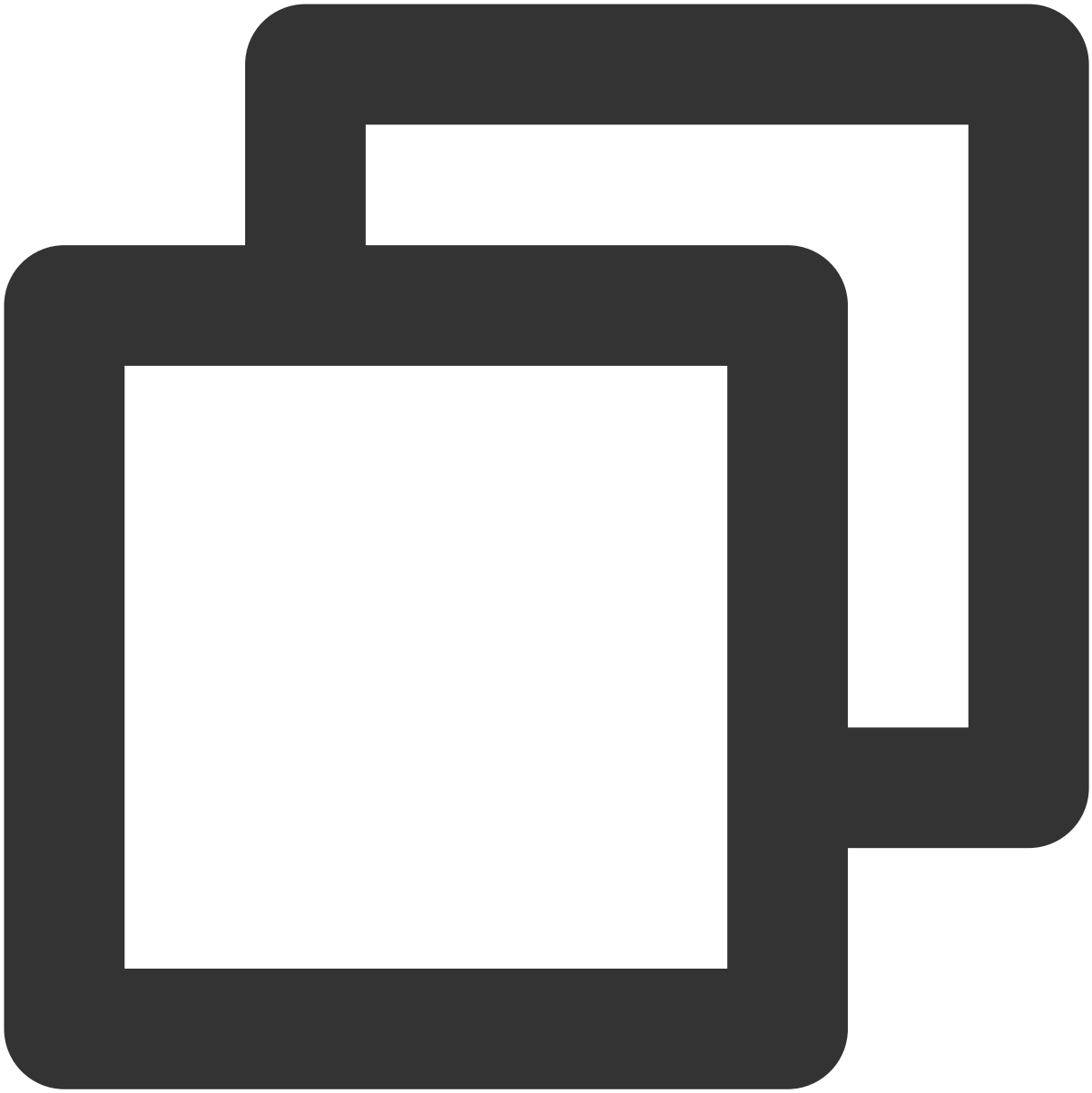
3. Concatenate the string to sign

This step generates the original signature string. The original signature string consists of the following parameters:

1. Request method: POST and GET methods are supported. GET is used here for the request. Please note that the method name should be in all capital letters.
2. Request server: the domain name of the request for querying instances (DescribeInstances) is `cvm.tencentcloudapi.com` . The actual request domain name varies by the module to which the API belongs. For more information, please see the specific API document.
3. Request path: the request path in the current version of TencentCloud API is fixed to `/` .
4. Request string: the request string generated in the previous step.

The rule for concatenating the original string of the signature is `request method + request server + request path + ? + request string` .

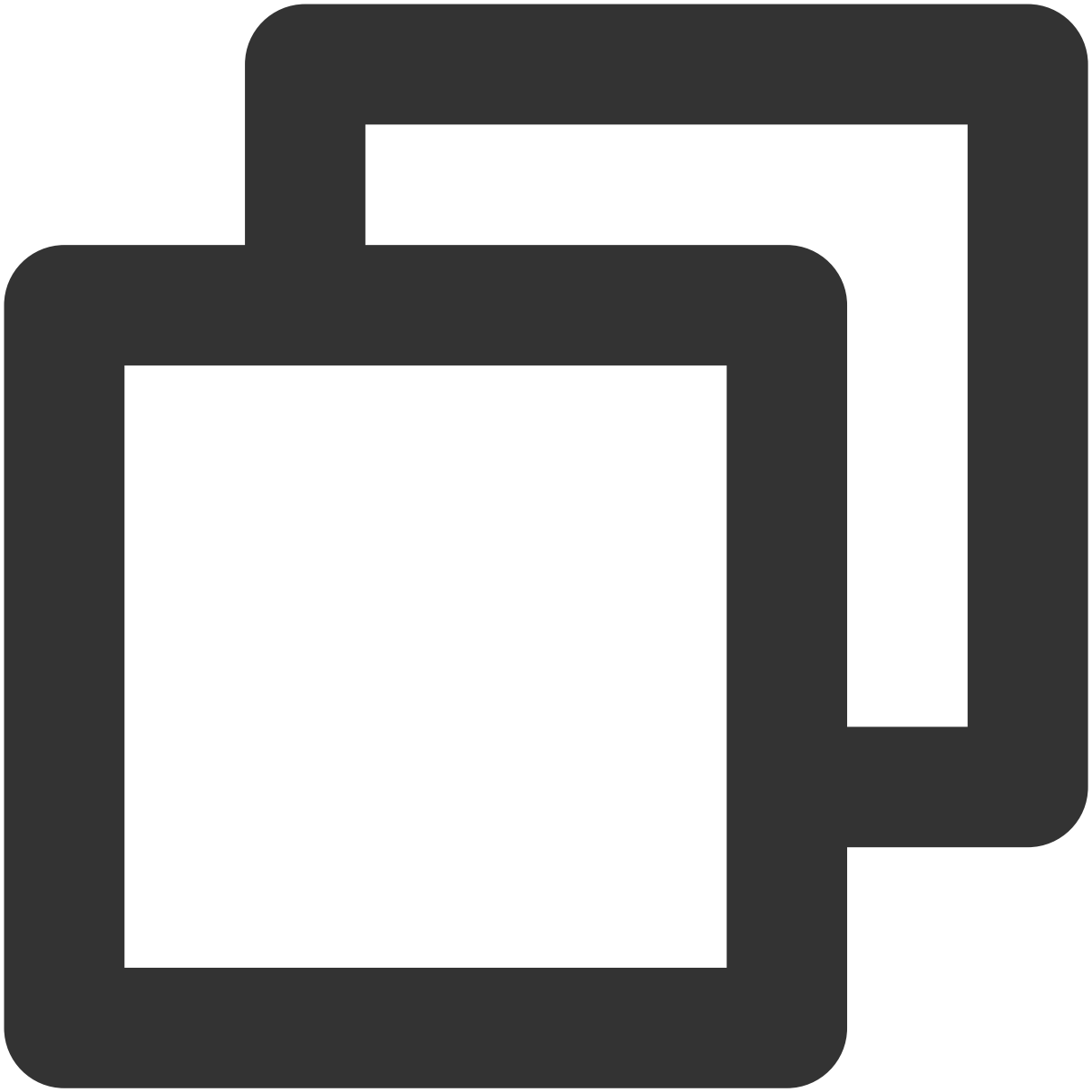
The concatenation result in the example is as follows:



```
GETcvm.tencentcloudapi.com/?Action=DescribeInstances&InstanceIds.0=ins-09dx96dg&Lim
```

4. Calculate the signature (pseudocode)

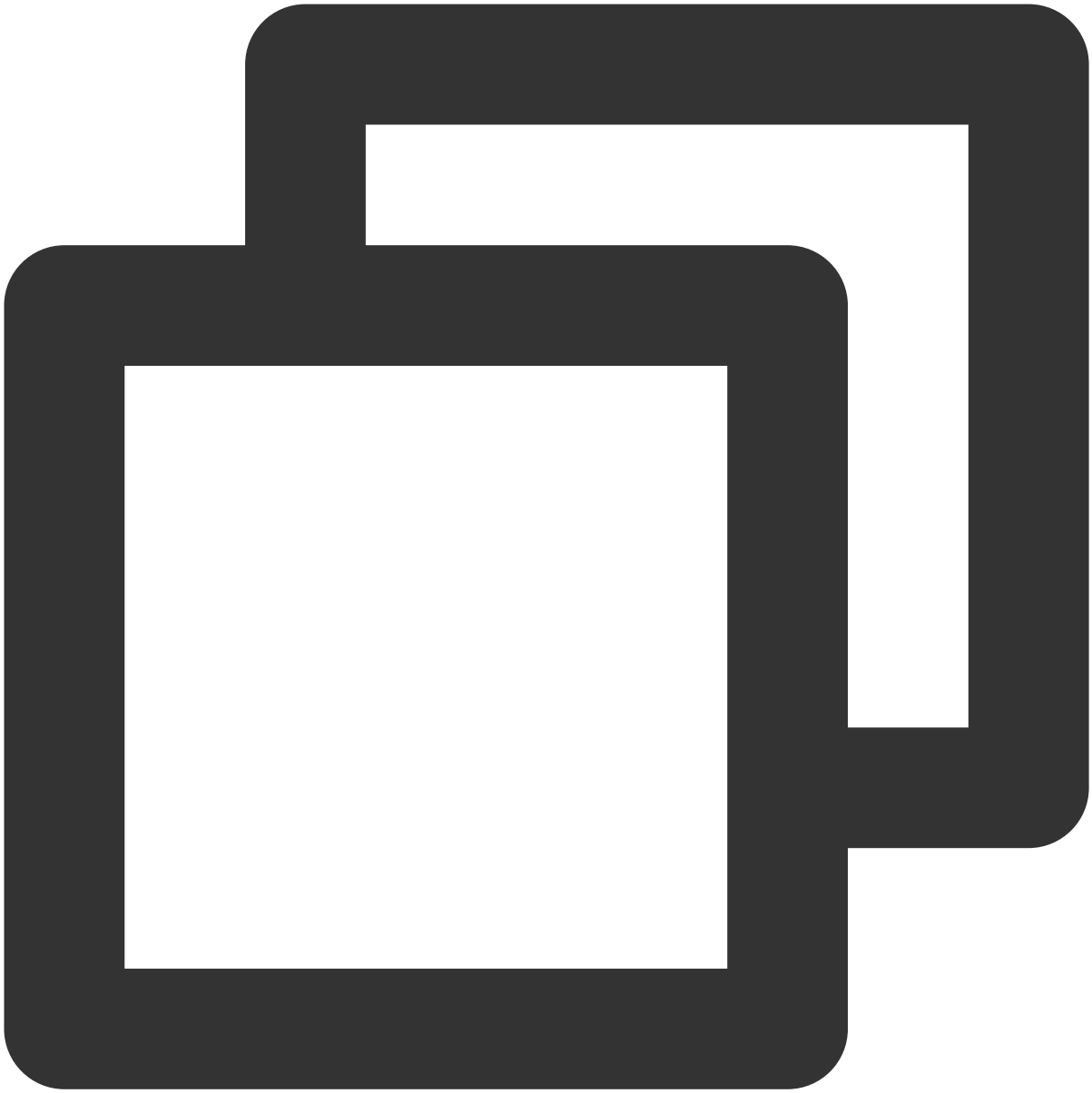
This step generates a signature string. Use the HMAC-SHA1 algorithm to sign the **original signature string** obtained in the previous step, and then Base64-encode the generated signature to get the final signature.



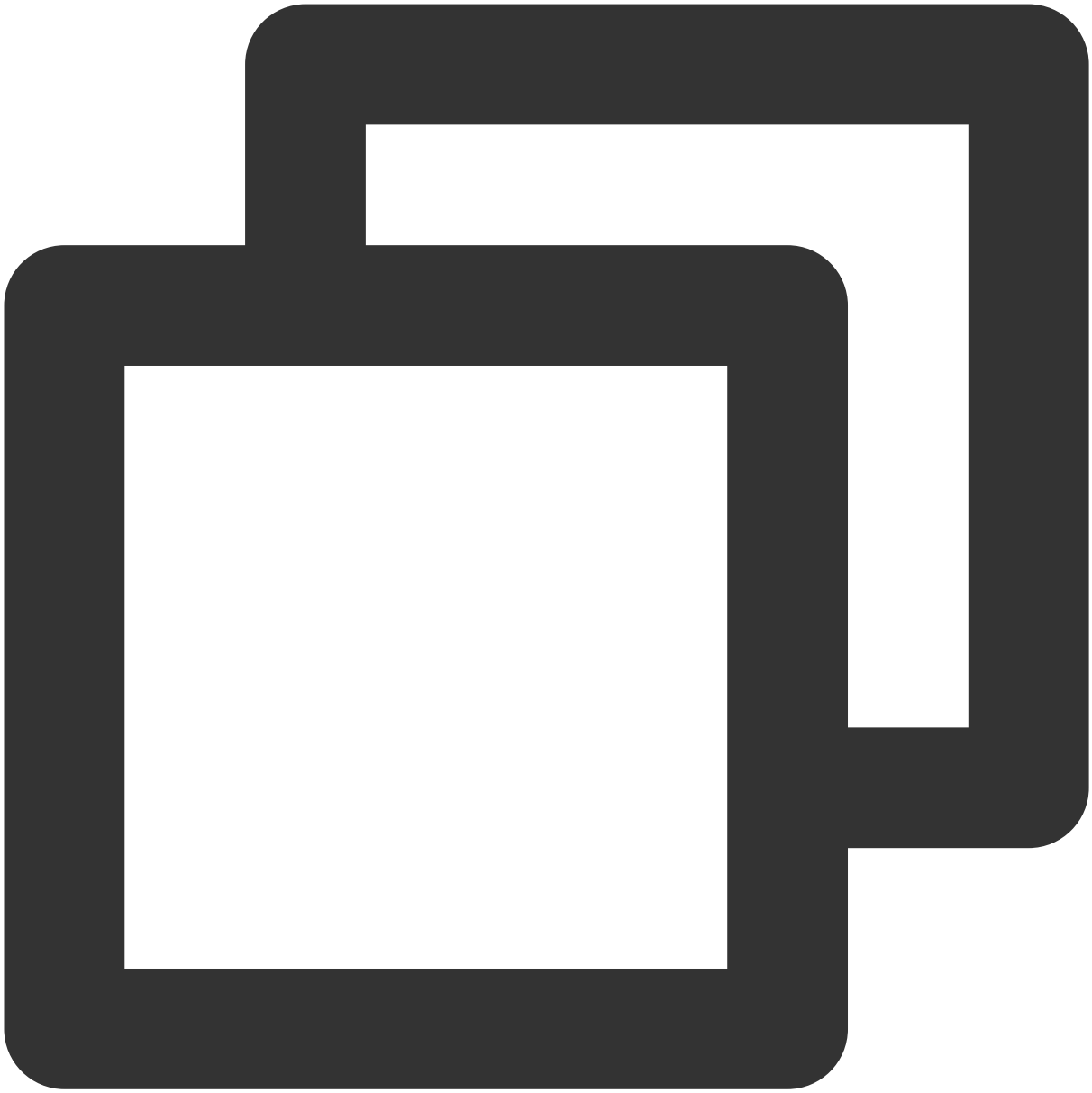
```
import javax.crypto.Mac;
import javax.crypto.SecretKey;
import javax.crypto.spec.SecretKeySpec;
import org.apache.commons.codec.binary.Base64;

/**
 *
 * [Hmac-SHA1 signature algorithm]
 * @String  encryptText [encrypted original string]
 * @String  encryptKey  [encryption key]
 * @return [signature value]
```

```
 */
public class HmacSHA1 {
    private static final String MAC_NAME = "HmacSHA1";
    public static final String ENCODING = "UTF-8";
    // Signature algorithm
    public static String HmacSHA1Encrypt(String s,String secret_key ) throws Except
        byte[] data = encryptKey.getBytes( ENCODING );
        SecretKey secretKey = new SecretKeySpec( data, MAC_NAME );
        Mac mac = Mac.getInstance( MAC_NAME );
        mac.init( secretKey );
        byte[] text = encryptText.getBytes( ENCODING );
        byte[] digest = mac.doFinal( text );
        return new String(Base64.encodeBase64(digest));
    }
}
String secret_key = "Gu5t9xGAR**********EXAMPLE";
String s = "GETcvm.tencentcloudapi.com/?Action=DescribeInstances&InstanceIds.0=ins-
// Final signature string
String Signature = new HmacSHA1();
System.out.println(Signature)
```

5. Get the call information and send a request



```
# The API will be called actually, and fees will be incurred if it is a consumption
resp = requests.get("https://" + endpoint, params=data)
print(resp.url)
```



```
The obtained request string is as follows:
https://cvm.tencentcloudapi.com/?Action=DescribeInstances&InstanceIds.0=ins-09dx96d
```

Field	Description
endpoint	Service address, such as `cvm.tencentcloudapi.com` .
data	API parameter of the sample API 3.0 signature v1. **Note:** you should add the calculated signature in the format of key-value pair to `data` .

Note:

The key in the example is not real, and the timestamp is not the current system time. If you open this URL in the browser or call it by using commands such as `curl` , an authentication error `The signature expired` will be returned. To obtain a URL that works, you need to replace the `SecretId` and `SecretKey` in this example with your own credentials and use the current system time as the `Timestamp` .

To further explain the signing process, Java is used as examples below to implement the process as described above. The request domain name, API, and parameter values in the above example are used here. The code below is for demonstration only. Please use the SDK for actual development.

6. Encode a signature string

The generated signature string cannot be directly used as a request parameter and needs to be URL-encoded. For example, if the signature string generated in the previous step is `Eli**********cGeI=` , the final value of the `Signature` request parameter will be `EliP**********eI%3d` , which will be used to generate the final request URL.

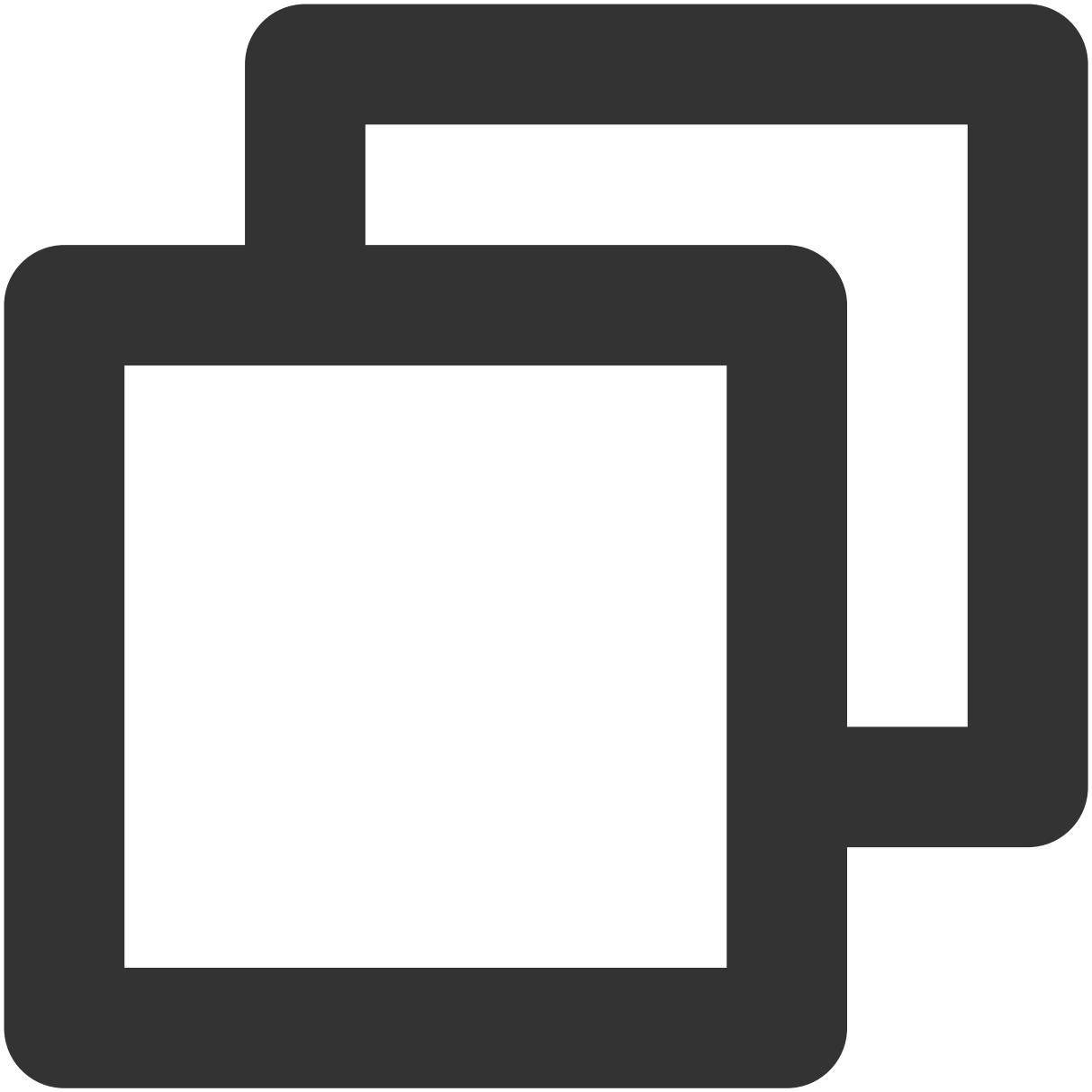
Note:

If you use the GET request method or use the POST request method with `Content-Type` of `application/x-www-form-urlencoded` , all the request parameter values must be URL-encoded (except the parameter key and the equal symbol (=)) before the request is sent. Non-ASCII characters must be encoded with **UTF-8** before URL-encoding.

The network libraries of some programming languages automatically URL-encode all parameters. In this case, the signature string does not need to be URL-encoded again; otherwise, two rounds of URL-encoding will cause the signature to fail.

Other parameter values also need to be encoded with RFC 3986. Use `%XY` in percent-encoding for special characters such as Chinese characters, where "X" and "Y" are hexadecimal characters (0–9 and uppercase A–F). Using lowercase characters will cause an error.

7. Sample API 3.0 signature v1



```
import java.io.UnsupportedEncodingException;
import java.net.URLEncoder;
import java.util.Random;
import java.util.TreeMap;
import javax.crypto.Mac;
import javax.crypto.spec.SecretKeySpec;
import javax.xml.bind.DatatypeConverter;

public class TencentCloudAPIDemo {
    private final static String CHARSET = "UTF-8";
```

```
    public static String sign(String s, String key, String method) throws Exception
        Mac mac = Mac.getInstance(method);
        SecretKeySpec secretKeySpec = new SecretKeySpec(key.getBytes(CHARSET), mac.
        mac.init(secretKeySpec);
        byte[] hash = mac.doFinal(s.getBytes(CHARSET));
        return DatatypeConverter.printBase64Binary(hash);
    }

    public static String getStringToSign(TreeMap<String, Object> params) {
        StringBuilder s2s = new StringBuilder("GETcvm.tencentcloudapi.com/?");
        // In the signing process, the parameters need to be sorted in lexicographi
        for (String k : params.keySet()) {
            s2s.append(k).append("=").append(params.get(k).toString()).append("&");
        }
        return s2s.toString().substring(0, s2s.length() - 1);
    }

    public static String getUrl(TreeMap<String, Object> params) throws UnsupportedE
        StringBuilder url = new StringBuilder("https://cvm.tencentcloudapi.com/?");
        // An actual request URL has no requirement for the order of parameters
        for (String k : params.keySet()) {
            // The request string needs to be URL-encoded. As the key consists of o
            url.append(k).append("=").append(URLEncoder.encode(params.get(k).toStri
        }
        return url.toString().substring(0, url.length() - 1);
    }

    public static void main(String[] args) throws Exception {
        TreeMap<String, Object> params = new TreeMap<String, Object>(); // TreeMap
        // A random number should be used during an actual call, such as `params.pu
        params.put("Nonce", 11886); // Common parameter
        // The current system time should be used during an actual call, such as `p
        params.put("Timestamp", 1465185768); // Common parameter
        params.put("SecretId", "AKIDz8krbsJ5**********mLPx3EXAMPL"); // Common para
        params.put("Action", "DescribeInstances"); // Common parameter
        params.put("Version", "2017-03-12"); // Common parameter
        params.put("Region", "ap-guangzhou"); // Common parameter
        params.put("Limit", 20); // Service parameter
        params.put("Offset", 0); // Service parameter
        params.put("InstanceIds.0", "ins-09dx96dg"); // Service parameter
        params.put("Signature", sign(getStringToSign(params), "Gu5t9xGAR***********
        System.out.println(getUrl(params));
    }
}
```

API 2.0 Signature

This signature version has been disused. We recommend you use **API 3.0 signature** with better performance. If you still need to use it, please go to API Explorer > **Signature Generation** and select **API 2.0 Signature** as the signature version.

Signature Failure

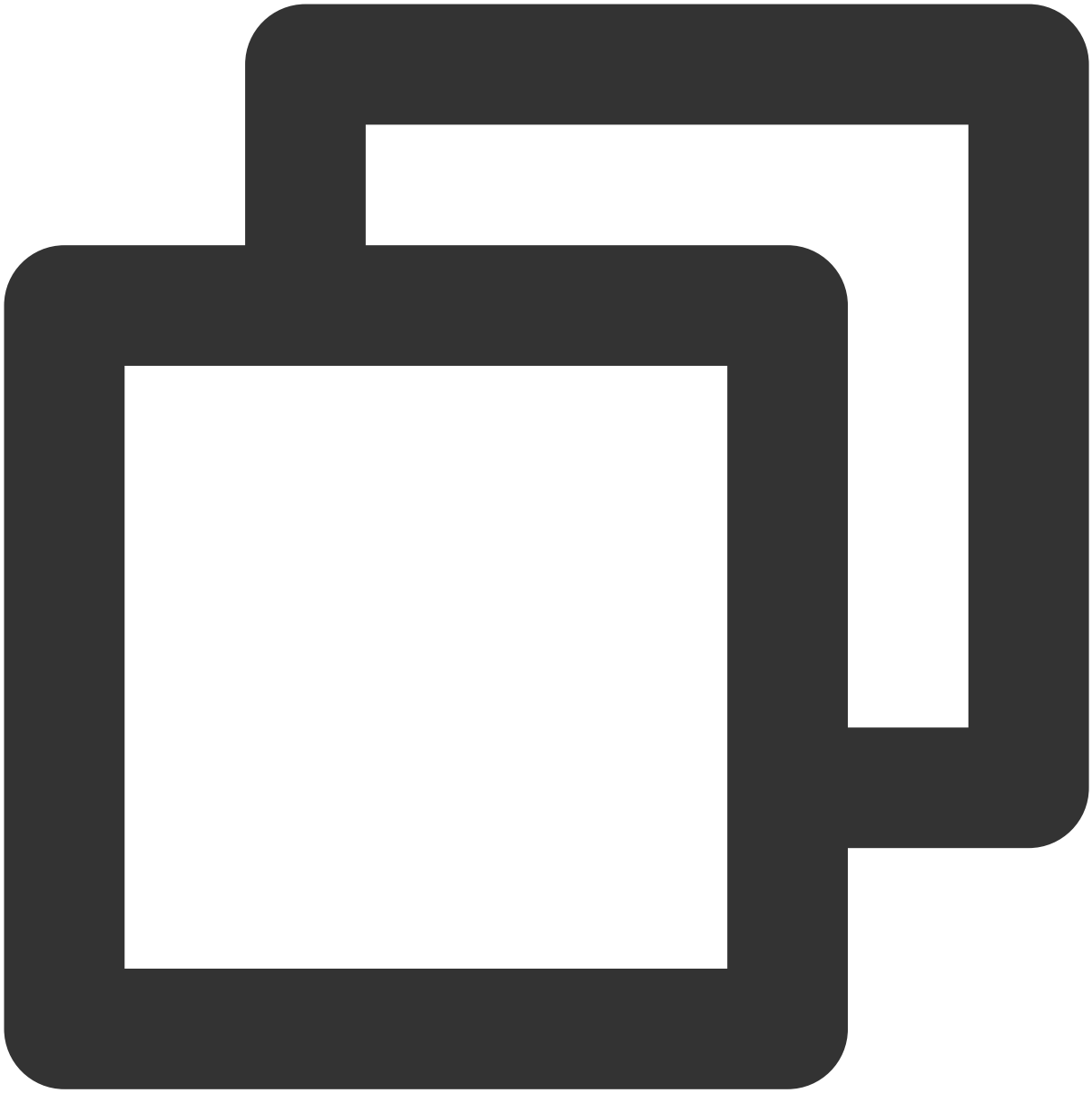
The following error codes may be returned for signature failure. Please resolve the errors accordingly.

Error Code	Error Description
AuthFailure.SignatureExpire	The signature expired. The difference between the `Timestamp` and the server time cannot be greater than five minutes.
AuthFailure.SecretIdNotFound	The key does not exist. Log in to the console and check whether it is disabled or you copied fewer or more characters.
AuthFailure.SignatureFailure	Signature error. It is possible that the signature is calculated incorrectly, the signature does not match the content that is actually sent, or the `SecretKey` is incorrect.
AuthFailure.TokenFailure	Temporary credential token error.
AuthFailure.InvalidSecretId	Invalid key (not TencentCloud API key type).

Returned Result

Successful response

For example, when calling the CVM API `DescribeInstancesStatus` (version: 2017-03-12) to view the status of instances, if the request succeeds, you may see the following response:



```
{
    "Response": {
        "TotalCount": 0,
        "InstanceStatusSet": [],
        "RequestId": "b5b41468-520d-4192-b42f-595cc34b6c1c"
    }
}
```

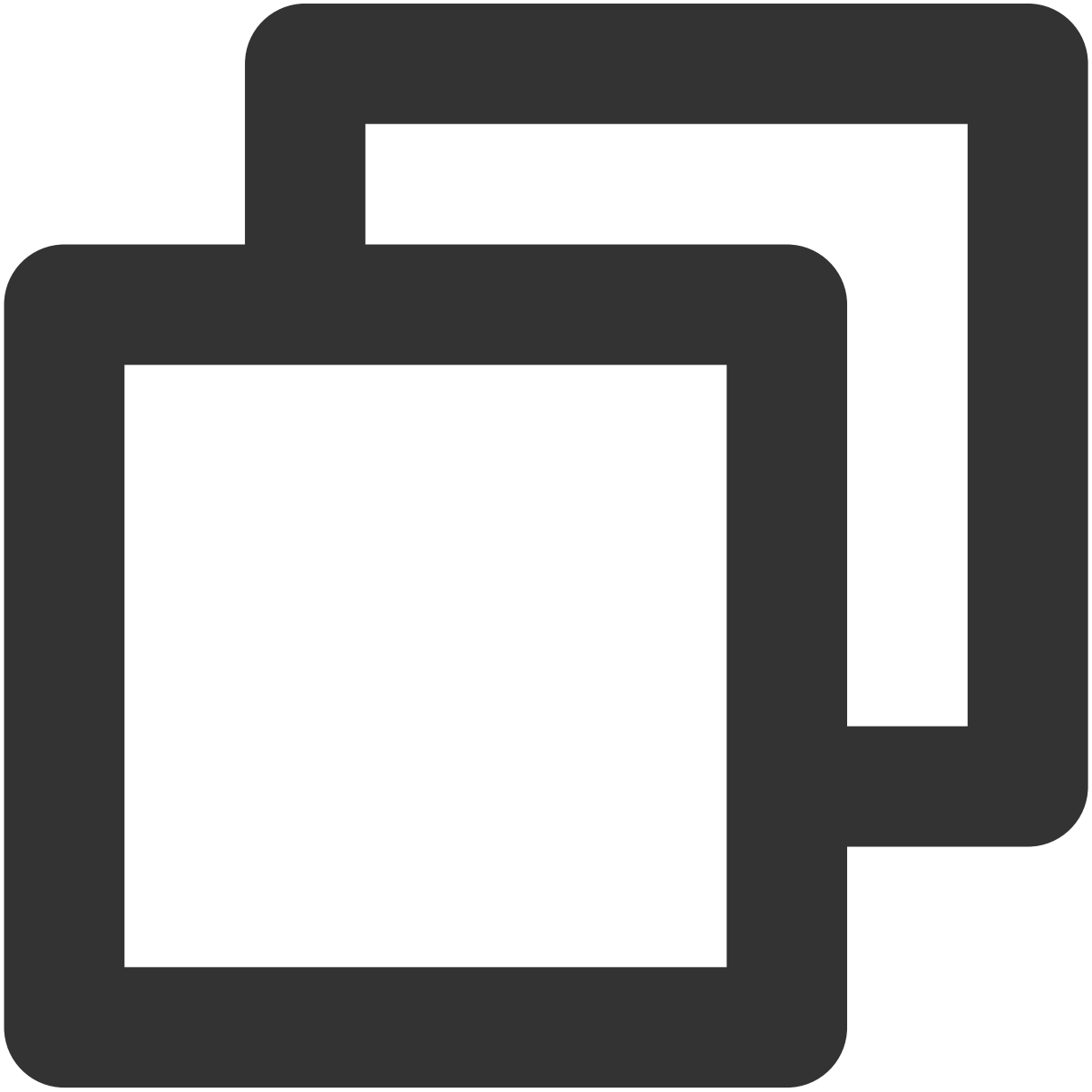
The API will return `Response` , which contains `RequestId` , as long as it processes the request, no matter whether the request is successful or not.

`RequestId` is the unique ID of an API request. It is required to troubleshoot issues.

Any fields other than the common fields are API-specific. For more information on such fields, please see the relevant API documentation. In this example, both `TotalCount` and `InstanceStatusSet` are specific to the `DescribeInstancesStatus` API. Since the user who initiated the request does not have a CVM instance yet, 0 is returned for `TotalCount` and `InstanceStatusSet` is empty.

Error response

If the call fails, you may see the following response:



```
{
```

```
    "Response": {
        "Error": {
            "Code": "AuthFailure.SignatureFailure",
            "Message": "The provided credentials could not be validated. Please che
        },
        "RequestId": "ed93f3cb-f35e-473f-b9f3-0d451b8b79c6"
    }
}
```

`Error` indicates that the request failed. A response for a failed request will always include the `Error`, `Code`, and `Message` fields.

`Code` indicates the specific error code, which is returned when an API request failed. You can use this code to locate the cause and solution of the error in the common or API-specific error code list.

`Message` explains the cause of the error. Note that the returned messages are subject to service updates. The information the messages provide may not be up-to-date and should not be the only source of reference.

`RequestId` is the unique ID of an API request. It is required to troubleshoot issues.

Common error codes

The `Error` field in a response indicates that the API call failed. The `Code` field in `Error` indicates the error code. The following table lists the common error codes that any services may return.

Error Code	Error Description
AuthFailure.InvalidSecretId	Invalid key (not TencentCloud API key type).
AuthFailure.MFAFailure	MFA failure.
AuthFailure.SecretIdNotFound	The key does not exist.
AuthFailure.SignatureExpire	The signature expired.
AuthFailure.SignatureFailure	Signature error.
AuthFailure.TokenFailure	Token error.
AuthFailure.UnauthorizedOperation	No CAM authorization.
DryRunOperation	DryRun Operation. It means that the request would have succeeded, but the DryRun parameter was used.
FailedOperation	The operation failed.
InternalError	Internal error.
InvalidAction	The API does not exist.

InvalidParameter	Incorrect parameter.
InvalidParameterValue	Invalid parameter value.
LimitExceeded	The quota limit is exceeded.
MissingParameter	A parameter is missing.
NoSuchVersion	The API version does not exist.
RequestLimitExceeded	The request rate limit is exceeded.
ResourceInUse	The resource is in use.
ResourceInsufficient	Insufficient resource.
ResourceNotFound	The resource does not exist.
ResourceUnavailable	The resource is unavailable.
UnauthorizedOperation	Unauthorized operation.
UnknownParameter	Unknown parameter error.
UnsupportedOperation	Unsupported operation.
UnsupportedProtocol	Unsupported HTTPS request method. Only GET and POST requests are supported.
UnsupportedRegion	Unsupported region.

API for Node.js

Last updated：2023-03-07 18:16:40

TencentCloud API has been upgraded to v3.0. This version is optimized for performance and deployed in all regions. It supports nearby access and access by region for significantly reduced access latency. In addition, it features more detailed API descriptions and error codes and API-level comments for SDKs, enabling you to use Tencent Cloud services more conveniently and quickly. This document describes how to call APIs for Node.js.
This version currently supports various Tencent Cloud services such as CVM, CBS, VPC, and TencentDB and will support more services in the future.

Request Structure

1. Service address (endpoint)

TencentCloud API supports access from either a nearby region (such as `cvm.tencentcloudapi.com` for CVM) or a specified region (such as `cvm.ap-guangzhou.tencentcloudapi.com` for CVM in the Guangzhou region). For values of the region parameter, please see the region list in the "Common Parameters" section below. To check whether a region is supported by a specific Tencent Cloud service, please see its "Request Structure" document.

Note:

For latency-sensitive businesses, we recommend you specify a domain name with a region.

2. Communications protocol

All TencentCloud APIs communicate over HTTPS, providing highly secure communications tunnels.

3. Request method

Supported HTTP request methods:
POST (recommended)
GET
`Content-Type` types supported by POST request:
application/json (recommended). The signature algorithm v3 (TC3-HMAC-SHA256) must be used.
application/x-www-form-urlencoded. The signature algorithm v1 (HmacSHA1 or HmacSHA256) must be used.
multipart/form-data (only supported by certain APIs). The signature algorithm v3 (TC3-HMAC-SHA256) must be used.
The size of a GET request packet cannot exceed 32 KB. The size of a POST request cannot exceed 1 MB for the signature algorithm v1 (HmacSHA1 or HmacSHA256) or 10 MB for the signature algorithm v3 (TC3-HMAC-SHA256).

4. Character encoding

UTF-8 encoding is always used.

Common Parameters

-The common parameters are used to identity the user and API signature. They should be carried by each request to initiate properly.

Signature algorithm v3

The signature algorithm v3 (sometimes referred to as "TC3-HMAC-SHA256") is more secure than the signature algorithm v1 (referred to as signature algorithm in certain documents), supports larger request packets and POST JSON format, and has a higher performance. We recommend you use it to calculate signatures. For more information on how to use it, please see below.

Parameter Name	Type	Required	Description
X-TC-Action	String	Yes	Name of the API for the desired operation. For the specific value, please see the description of common parameter `Action` in the input parameters in the related API document. For example, the API for querying CVM instance list is `DescribeInstances` .
X-TC-Region	String	-	Region parameter, which is used to identify the region where the data you want to manipulate resides. For values supported for an API, please see the description of common parameter `Region` in the input parameters in related API documentation. **Note: this parameter is not required for some APIs (which will be indicated in related API documentation) and will not take effect even if it is passed.**
X-TC-Timestamp	Integer	Yes	The current UNIX timestamp that records the time when the API request was initiated, such as 1529223702. **Note: if the difference between the UNIX timestamp and the server time is greater than 5 minutes, a signature expiration error may occur.**
X-TC-Version	String	Yes	Version of the API for the desired operation, such as 2017-03-12 for CVM. For the specific value, please see the description of common parameter `Version` in the input parameters in related API documentation.
Authorization	String	Yes	HTTP authentication request header, such as TC3-HMAC-SHA256 Credential=AKIDEXAMPLE/Date/service/tc3_request, SignedHeaders=content-type;host, Signature=72e494ea8**********a96525168

			Here, TC3-HMAC-SHA256: signature algorithm, currently fixed as this value. Credential: signature credential. `AKIDEXAMPLE` indicates the `SecretId`. `Date` indicates a UTC date which must match the value of `X-TC-Timestamp` (a common parameter) in UTC format. `service` indicates the name of the service and is generally a domain name prefix; for example, the domain name `cvm.tencentcloudapi.com` means the CVM service, and the value for this service is `cvm`. SignedHeaders: the headers that contain the authentication information. `content-type` and `host` are required. Signature: signature digest. For the calculation process, please see below.
X-TC-Token	String	No	Token used for temporary credentials. It must be used with a temporary key. You can get the temporary key and token by calling a CAM API. No token is required for a long-term key.

Signature algorithm v1

When the signature algorithm v1 (sometimes referred to as "HmacSHA256" or "HmacSHA1") is used, the common parameters should be uniformly placed in the request string.

Parameter Name	Type	Required	Description
Action	String	Yes	Name of the API for the desired operation. For the specific value, please see the description of common parameter `Action` in the input parameters in the related API document. For example, the API for querying CVM instance list is `DescribeInstances`.
Region	String	-	Region parameter, which is used to identify the region where the data you want to manipulate resides. For values supported for an API, please see the description of common parameter `Region` in the input parameters in related API documentation. **Note: this parameter is not required for some APIs (which will be indicated in related API documentation) and will not take effect even if it is passed.**
Timestamp	Integer	Yes	The current UNIX timestamp that records the time when the API request was initiated, such as 1529223702. If the difference between the UNIX timestamp and the current time is too large, a signature expiration error may occur.

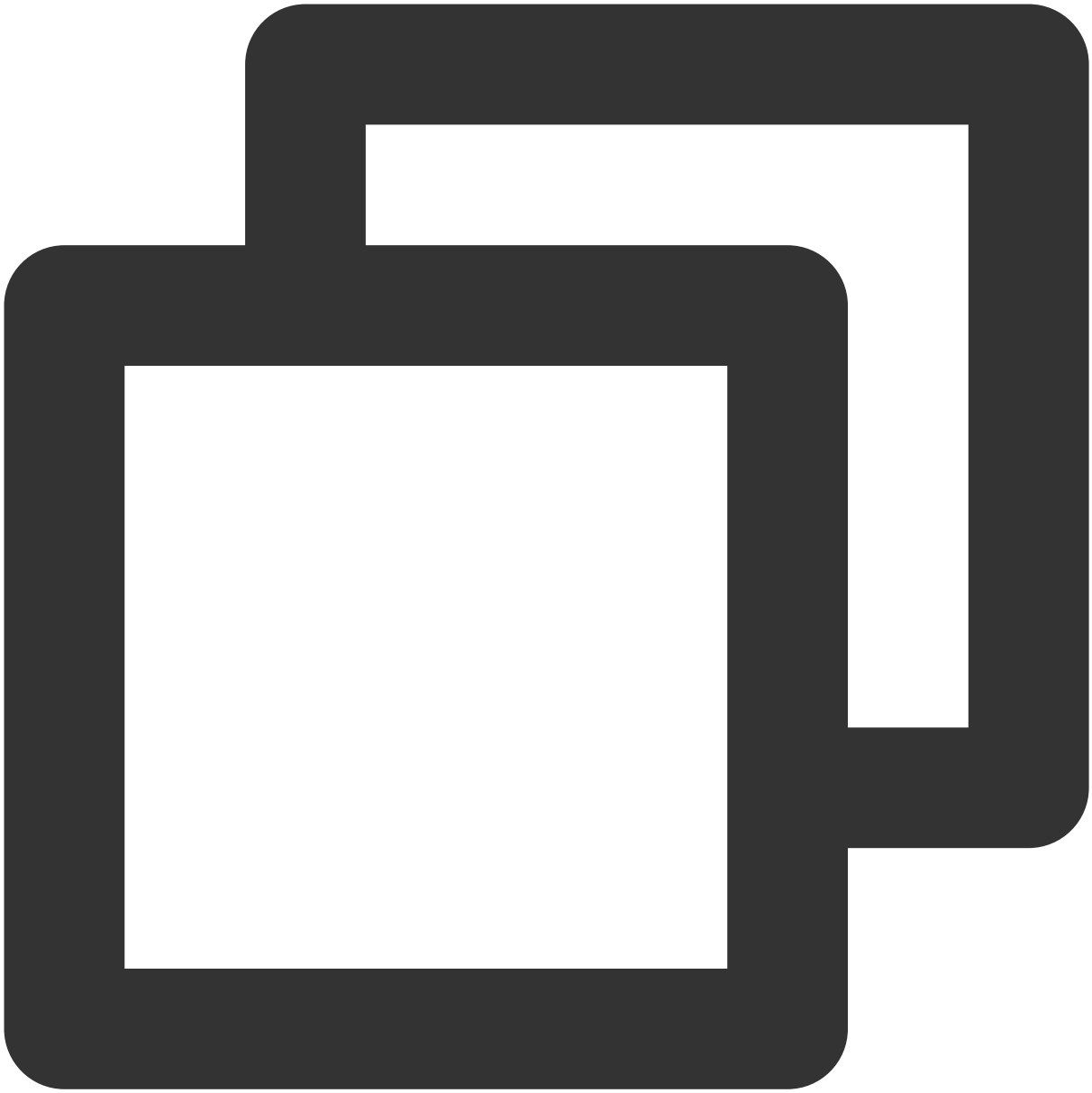
Nonce	Integer	Yes	A random positive integer used in conjunction with `Timestamp` to prevent replay attacks.
SecretId	String	Yes	The identifying `SecretId` obtained on the TencentCloud API Key page. A `SecretId` corresponds to a unique `SecretKey` which is used to generate the request signature (`Signature`).
Signature	String	Yes	Request signature, which is used to verify the validity of the request. It is generated based on input parameters. For more information on how to calculate the signature, please see below.
Version	String	Yes	Version of the API for the desired operation, such as 2017-03-12 for CVM. For the specific value, please see the description of common parameter `Version` in the input parameters in related API documentation.
SignatureMethod	String	No	Signature algorithm. Currently, only HmacSHA256 and HmacSHA1 are supported. The HmacSHA256 algorithm is used to verify the signature only when this parameter is specified as HmacSHA256. In other cases, the signature is verified with HmacSHA1.
Token	String	No	Token used for temporary credentials. It must be used with a temporary key. You can get the temporary key and token by calling a CAM API. No token is required for a long-term key.

Region list

As the supported regions vary by service, please refer to the region list in each service's product documentation for specific details.
For example, you can see the region list of CVM.

API Call Method for Node.js

TencentCloud API authenticates every request, that is, the request must be signed with the security credentials in the designated steps. Each request must contain the signature information in the common request parameters and be sent in the specified way and format.
Suppose your `SecretId` and `SecretKey` are `AKIDz8krbsJ5**********mLPx3EXAMPLE` and `Gu5t9xGAR**********EXAMPLE` , respectively. If you want to view the status of an unnamed instance in the Guangzhou region and have only one data entry returned, the request may be:



```
curl -X POST https://cvm.tencentcloudapi.com \\
-H "Authorization: TC3-HMAC-SHA256 Credential=AKIDz8krbsJ5**********mLPx3EXAMPLE/20
-H "Content-Type: application/json; charset=utf-8" \\
-H "Host: cvm.tencentcloudapi.com" \\
-H "X-TC-Action: DescribeInstances" \\
-H "X-TC-Timestamp: 1551113065" \\
-H "X-TC-Version: 2017-03-12" \\
-H "X-TC-Region: ap-guangzhou" \\
-d '{"Limit": 1, "Filters": [{"Values": ["\\u672a\\u547d\\u540d"], "Name": "instanc
```

Step 1. Apply for security credentials

In this document, the security credential used is a key pair, which consists of a `SecretId` and a `SecretKey`.

Each user can have up to two key pairs.

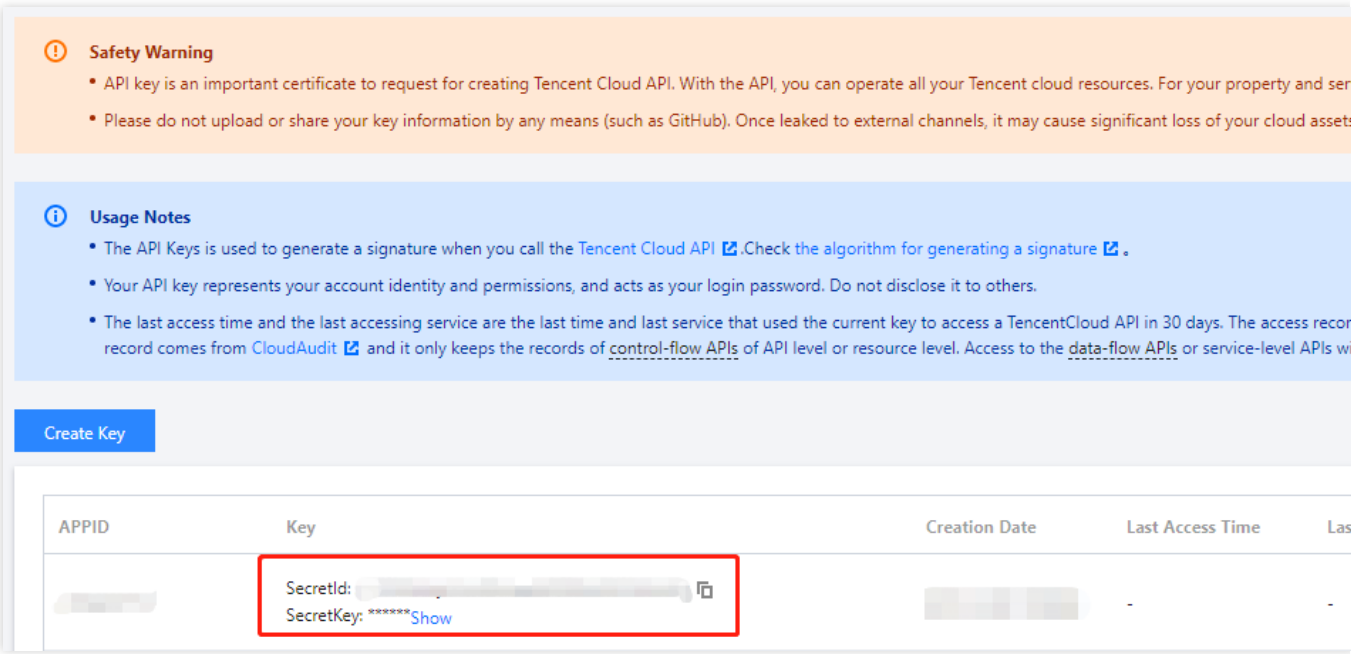
SecretId: identifies the user that calls an API, which is similar to a username.

SecretKey: authenticates the user that calls the API, which is similar to a password.

Note:

You must keep your security credentials private and avoid disclosure; otherwise, your assets may be compromised. If they are disclosed, please disable them as soon as possible.

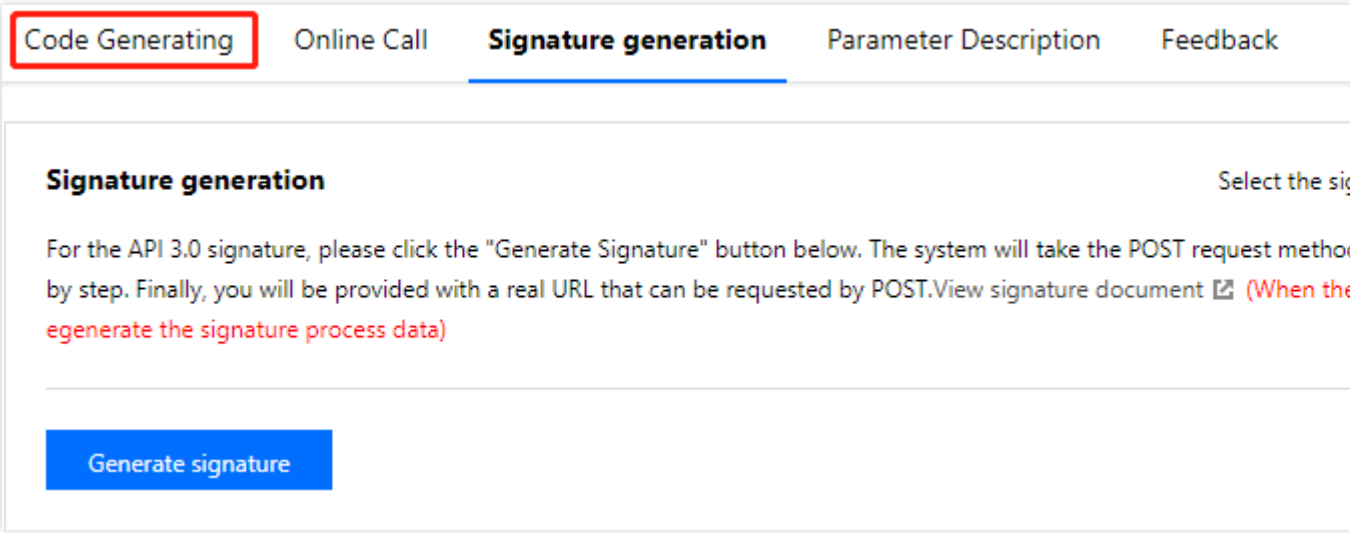
Go to the API key management page to get API keys as shown below:



Step 2

1. Get an API 3.0 signature v3

The signature algorithm v3 (TC3-HMAC-SHA256) is compatible with the previous signature algorithm v1 and more secure, supports larger request packets and POST JSON format, and has a higher performance. We recommend you use it to calculate signatures.



Note:

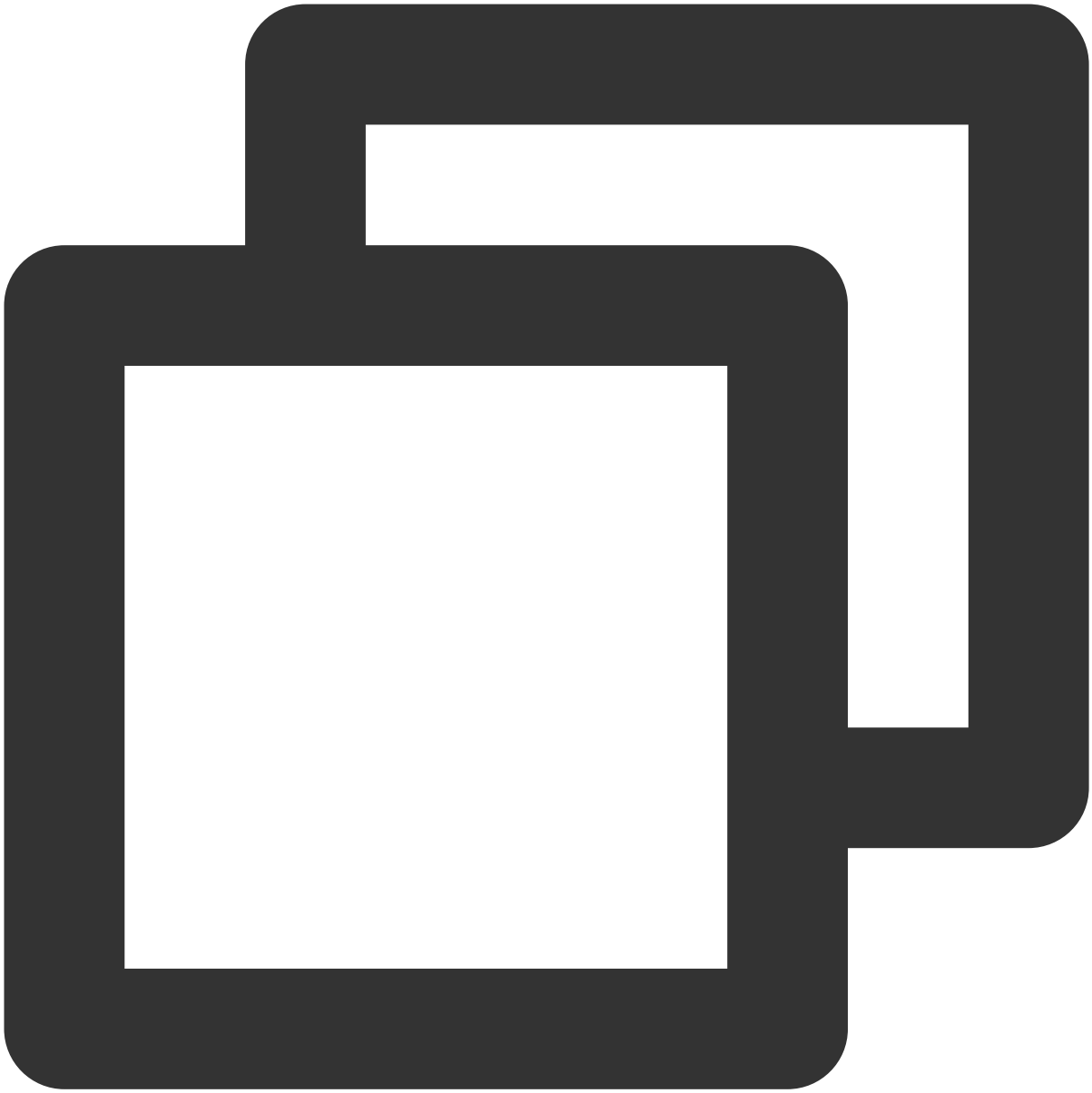
If you are using the signature algorithm for the first time, we recommend you use the "signature string generation" feature in API Explorer and select "API 3.0 signature v3" as the signature version, which can generate a signature for demonstration and verification. Plus, it can also generate SDK code directly. Seven common open-source programming language SDKs are available for TencentCloud API, including Python, Java, PHP, Go, Node.js, .NET, and C++.

TencentCloud API supports both GET and POST requests. For the GET method, only the `Content-Type: application/x-www-form-urlencoded` protocol format is supported. For the POST method, `Content-Type: application/json` and `Content-Type: multipart/form-data` are supported. The JSON format is supported by all business APIs, while the multipart format is supported only by specific APIs (in this case, an API cannot be called in JSON format). For more information, please see the specific business API document. We recommend you use the POST method because the two methods generate the same results, but the GET method only supports request packets below 32 KB in size.

The following describes how to calculate a signature by calling the DescribeInstances API. This API is chosen because:

1. The CVM API is enabled by default, and this API is often used.
2. It is read-only and does not change the status of existing resources.
3. It covers many types of parameters so that it is easy to show how to use an array that contains data structures.

1. Concatenate the canonical request string

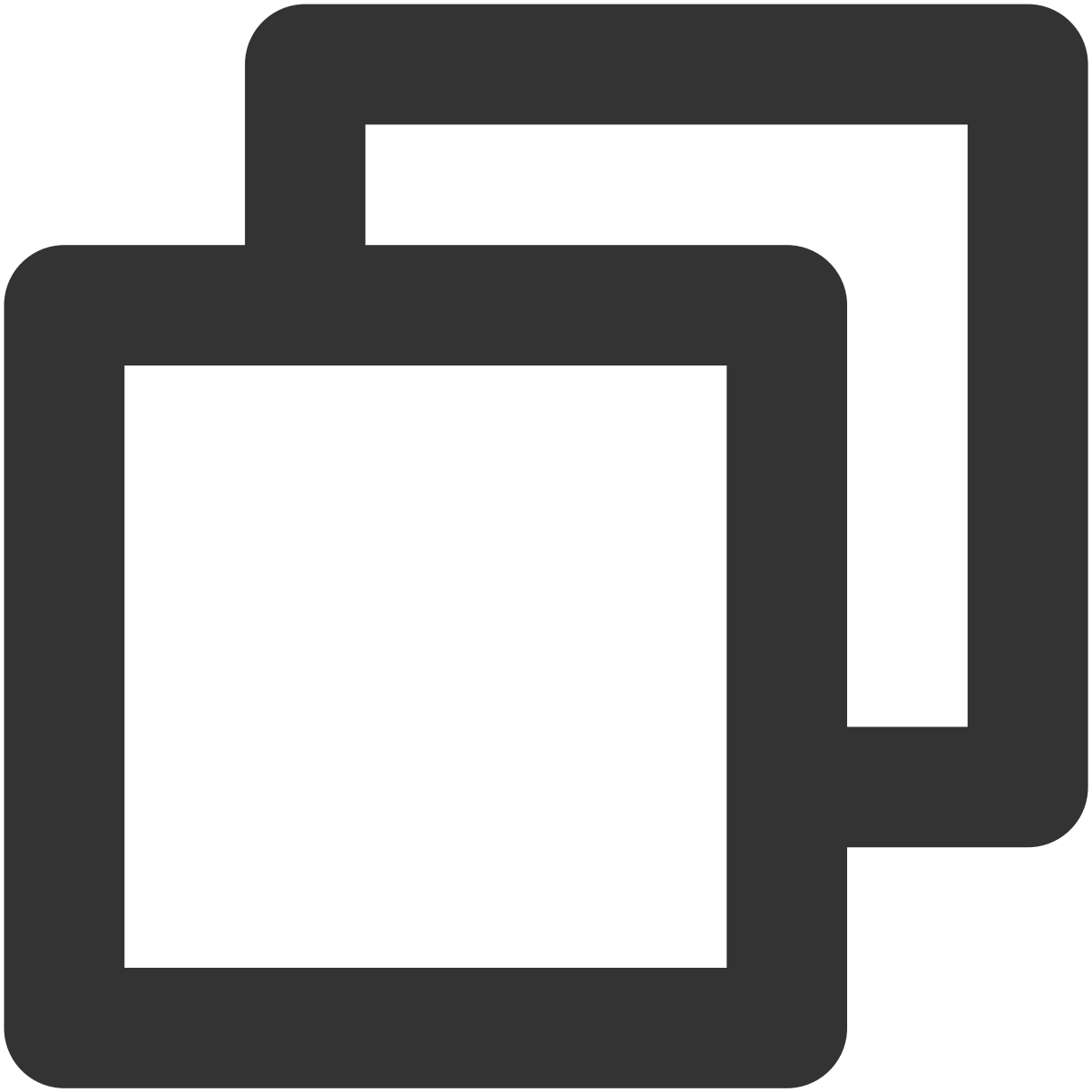


```
CanonicalRequest =
    HTTPRequestMethod + '\\n' +
    CanonicalURI + '\\n' +
    CanonicalQueryString + '\\n' +
    CanonicalHeaders + '\\n' +
    SignedHeaders + '\\n' +
    HashedRequestPayload
```

Field	Description

HTTPRequestMethod	HTTP request method (GET or POST). This example uses `POST` .
CanonicalURI	URI parameter. Slash ("/") is used for API 3.0.
CanonicalQueryString	Query string in the URL of the originating HTTP request. This is always an empty string POST requests and is the string after the question mark (?) for GET requests such as `Limit=10&Offset=0` . Note: `CanonicalQueryString` must be URL-encodec instructed in RFC 3986 with the UTF-8 character set. The applicable standard program language library is recommended. All special characters must be encoded and capitaliz
CanonicalHeaders	Header information for signature calculation, including at least `host` and `content type` . Custom headers can also be added to the signature process to improve the uniqueness and security of the request. Concatenation rules: both the key and value of header should be converted to lowercase with the leading and trailing spaces removed that they are concatenated in the `key:value\\n` format. If there are multiple heade they should be sorted in ASCII ascending order by header key (lowercase). The calcula result in this example is `content-type:application/json; charset=utf-8\\nhost:cvm.tencentcloudapi.com\\n` . Note: `content-type` must mat the content that is actually sent. In some programming languages, a `charset` value automatically added even if it is not specified. In this case, the request sent will be differ from the one signed, and the sever will return a signature verification failure.
SignedHeaders	Header information for signature calculation, indicating the request headers that are inv in the signature process. The request headers must correspond to the headers in `CanonicalHeaders` . `Content-type` and `host` are required headers. Concatenation rules: both the key and value of a header should be converted to lowerca there are multiple headers, they should be sorted in ASCII ascending order by header k (lowercase) and separated by semicolons (;). The value in this example is `content-type;host` .
HashedRequestPayload	Hash value of `Requestpayload` (i.e., the request body, such as `{"Limit": 1, "Filters": [{"Values": ["\\u672a\\u547d\\u540d"], "Name": "instance-name"}]}` in this example). The pseudo-code for calculation is `Lowercase(HexEncode(Hash.SHA256(RequestPayload)))` , which means t SHA256 hashing is performed on the payload of the HTTP request, then hexadecimal encoding is performed, and finally the encoded string is converted to lowercase letters. GET requests, `RequestPayload` is always an empty string. The calculation result i example is `35e9c5b0e3ae67532d3c9f17ead6c90222632e5b1ff7f6e89887f1398934f(`

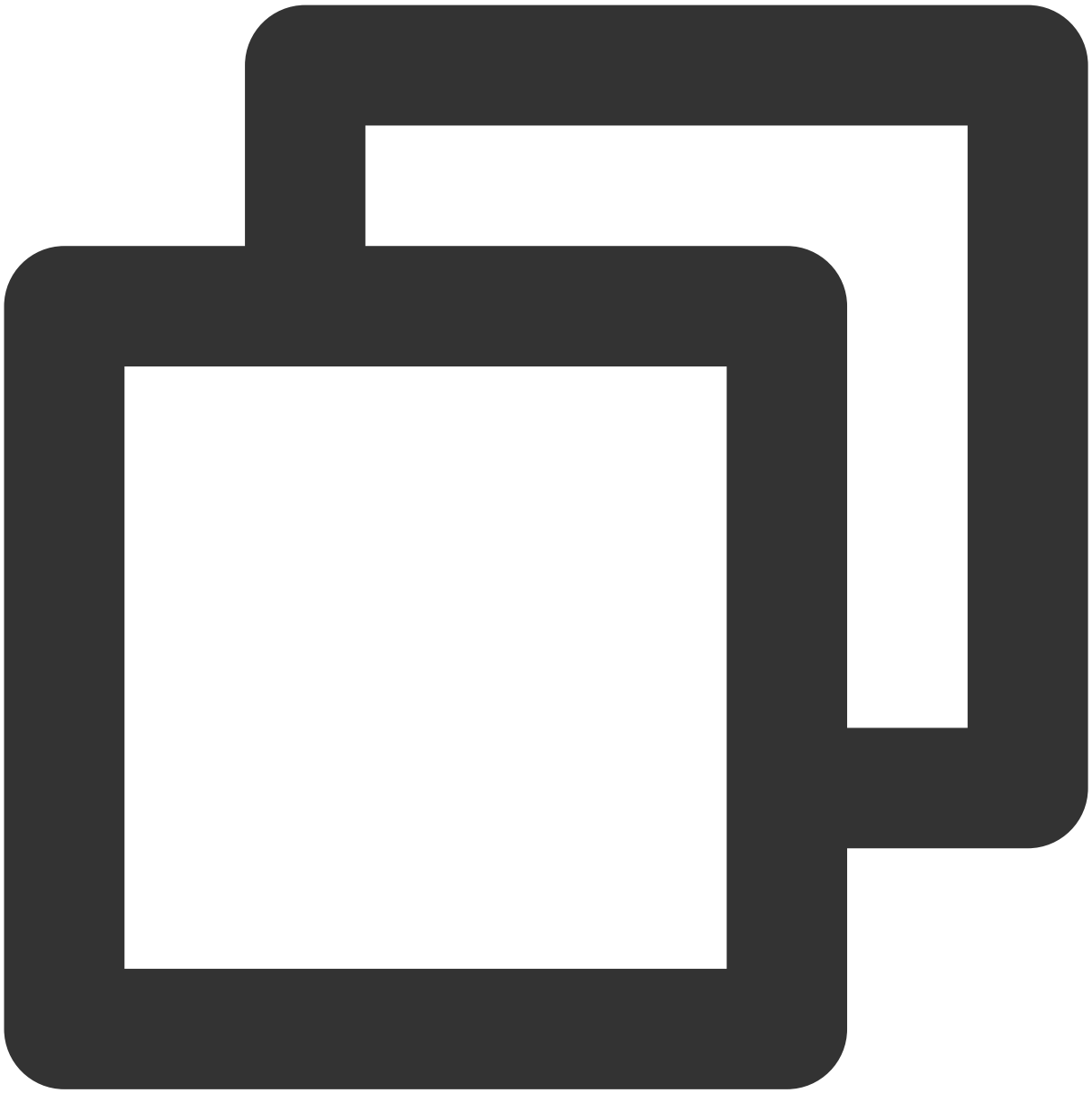
According to the rules above, the canonical request string obtained in the example is as follows:



```
POST
/

content-type:application/json; charset=utf-8
host:cvm.tencentcloudapi.com

content-type;host
35e9c5b0e3ae67532d3c9f17ead6c90222632e5b1ff7f6e89887f1398934f064
```

2. Concatenate the string to sign

Concatenate the string to sign in the following format:



```
StringToSign =
    Algorithm + \\n +
    RequestTimestamp + \\n +
    CredentialScope + \\n +
    HashedCanonicalRequest
```

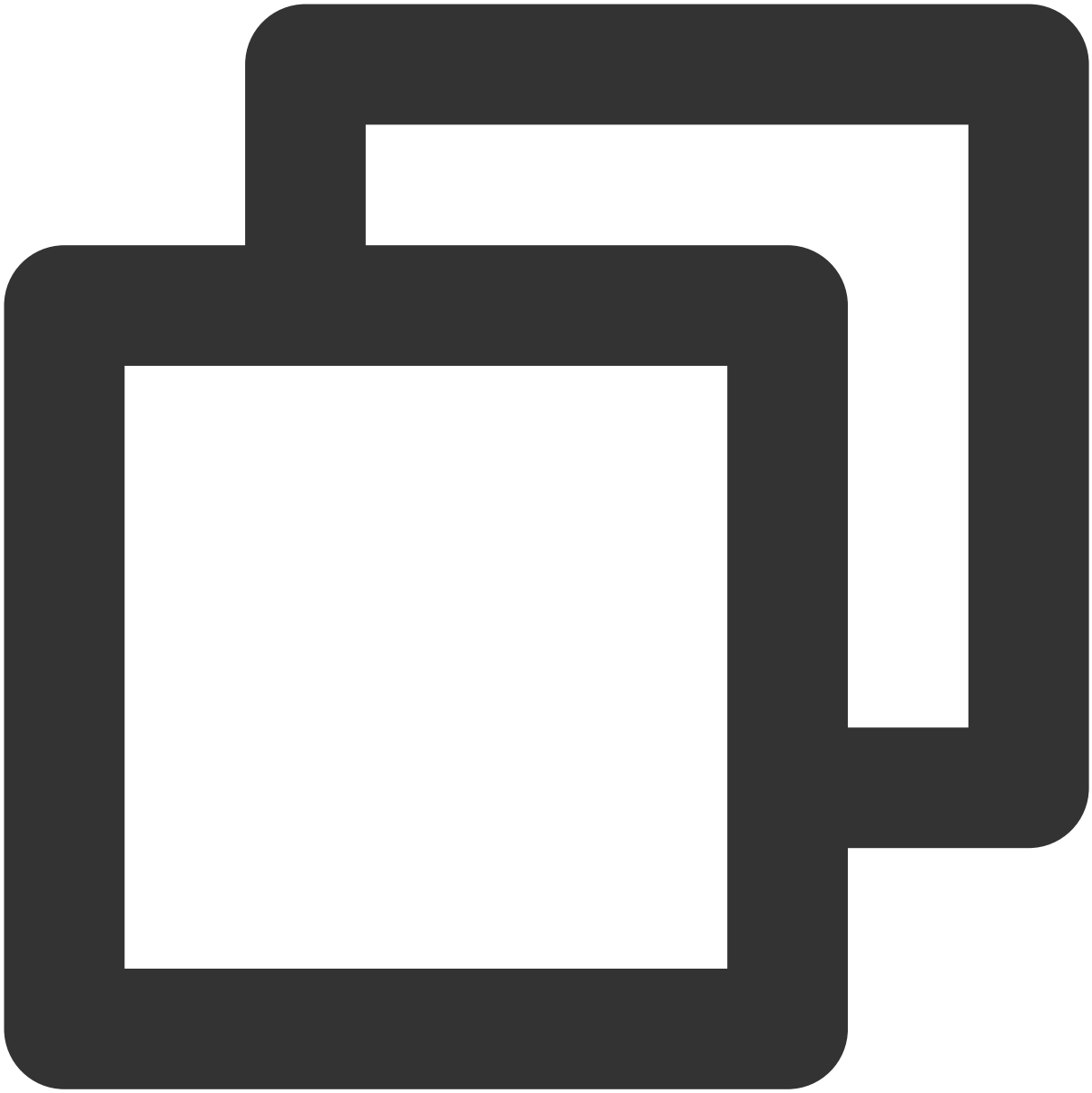
Field	Description
Algorithm	Signature algorithm, which is always `TC3-HMAC-SHA256` currently.

RequestTimestamp	Request timestamp, i.e., the value of the common parameter `X-TC-Timestamp` in request header. It is the UNIX timestamp of the current time in seconds, such as `1551113065` in this example.
CredentialScope	Scope of the credential in the format of `Date/service/tc3_request`, including date, requested service, and termination string (tc3_request). **`Date` indicates a U date, which should match the UTC date converted by the common parameter `TC-Timestamp`.** `service` is the service name, which should match the domain n of the service called. The calculation result in this example is `2019-02-25/cvm/tc3_request`.
HashedCanonicalRequest	Hash value of the canonical request string concatenated in the steps above. The pseu code for calculation is `Lowercase(HexEncode(Hash.SHA256(CanonicalRequest)))`. The calcu result in this example is `5ffe6a04c0664d6b969fab9a13bdab201d63ee709638e2749d62a09ca18d`

Note:

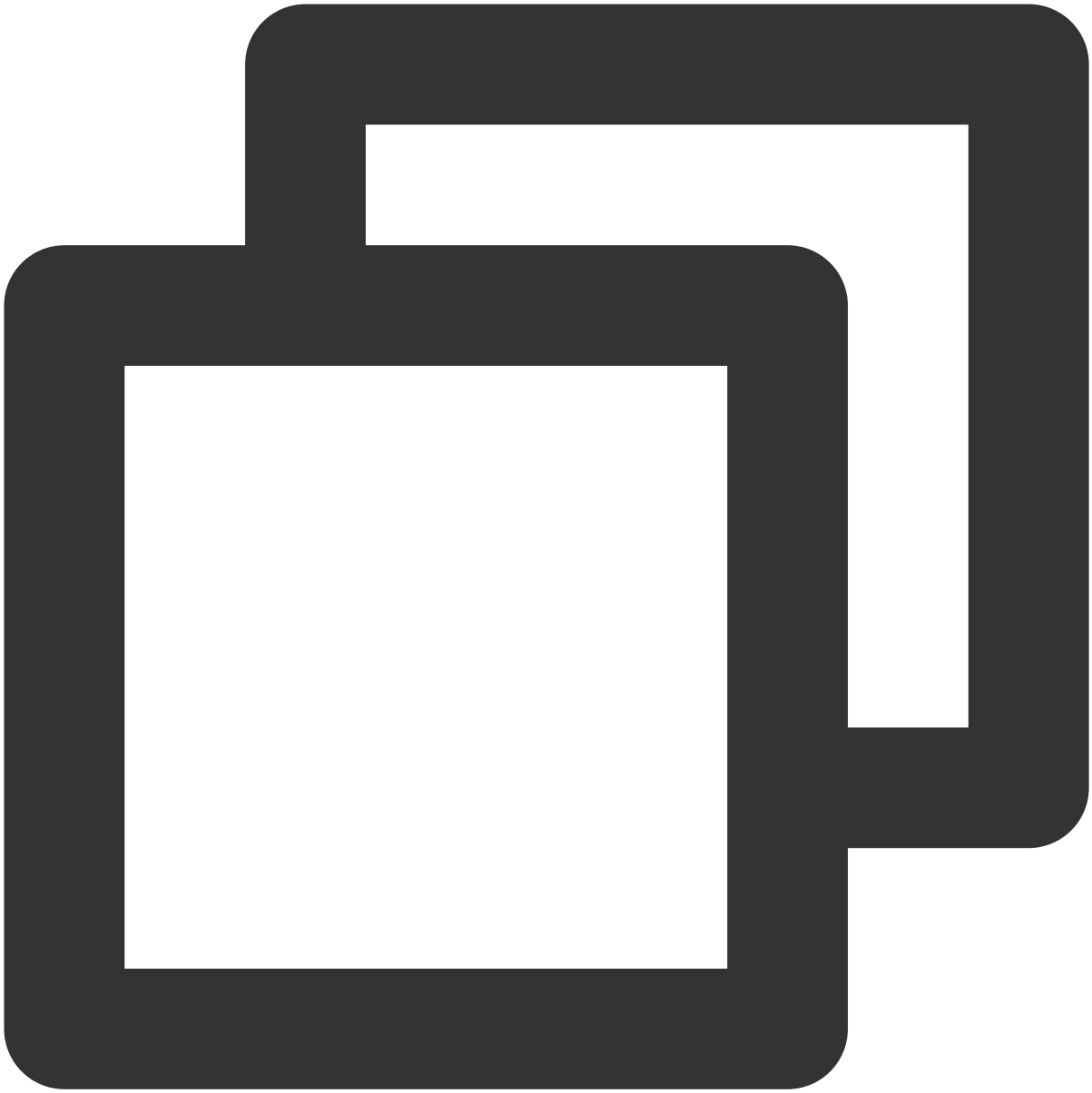
1. `Date` must be calculated from the timestamp `X-TC-Timestamp` and the time zone is UTC+0. If you add the local time zone information (such as UTC+8) in the system, calls can succeed both day and night but will definitely fail at 00:00. For example, if the timestamp is 1551113065 and the time in UTC+8 is 2019-02-26 00:44:25, the UTC+0 date in the calculated `Date` value should be 2019-02-25 instead of 2019-02-26.
2. `Timestamp` must be the same as your current system time, and your system time must be in sync with the UTC time. If the difference between the timestamp and your current system time is greater than five minutes, the request will fail. If your system time is out of sync with the UTC time for a prolonged period, the request will fail, and a signature expiration error will be returned.

According to the rules above, the string to sign obtained in the example is as follows:



```
TC3-HMAC-SHA256
1551113065
2019-02-25/cvm/tc3_request
5ffe6a04c0664d6b969fab9a13bdab201d63ee709638e2749d62a09ca18d7031
```

3. Calculate the signature (pseudocode)



```
const kDate = sha256(date, 'TC3' + SECRET_KEY)
const kService = sha256(service, kDate)
const kSigning = sha256('tc3_request', kService)
const signature = sha256(stringToSign, kSigning, 'hex')
```

The derived key `SecretDate` , `SecretService` , and `SecretSigning` are binary data and may contain non-printable characters. Intermediate results are not displayed here.

Field	Description
secretKey	Original `SecretKey` , i.e., `Gu5t9xGAR**********EXAMPLE` .

date	Value of the `Date` field in `Credential`, such as `2019-02-25` in this example.
service	Value of the `Service` field in `CredentialScope`, such as `cvm` in this example.

The calculation result in this example is `72e494ea8*****************************************a96525168`.

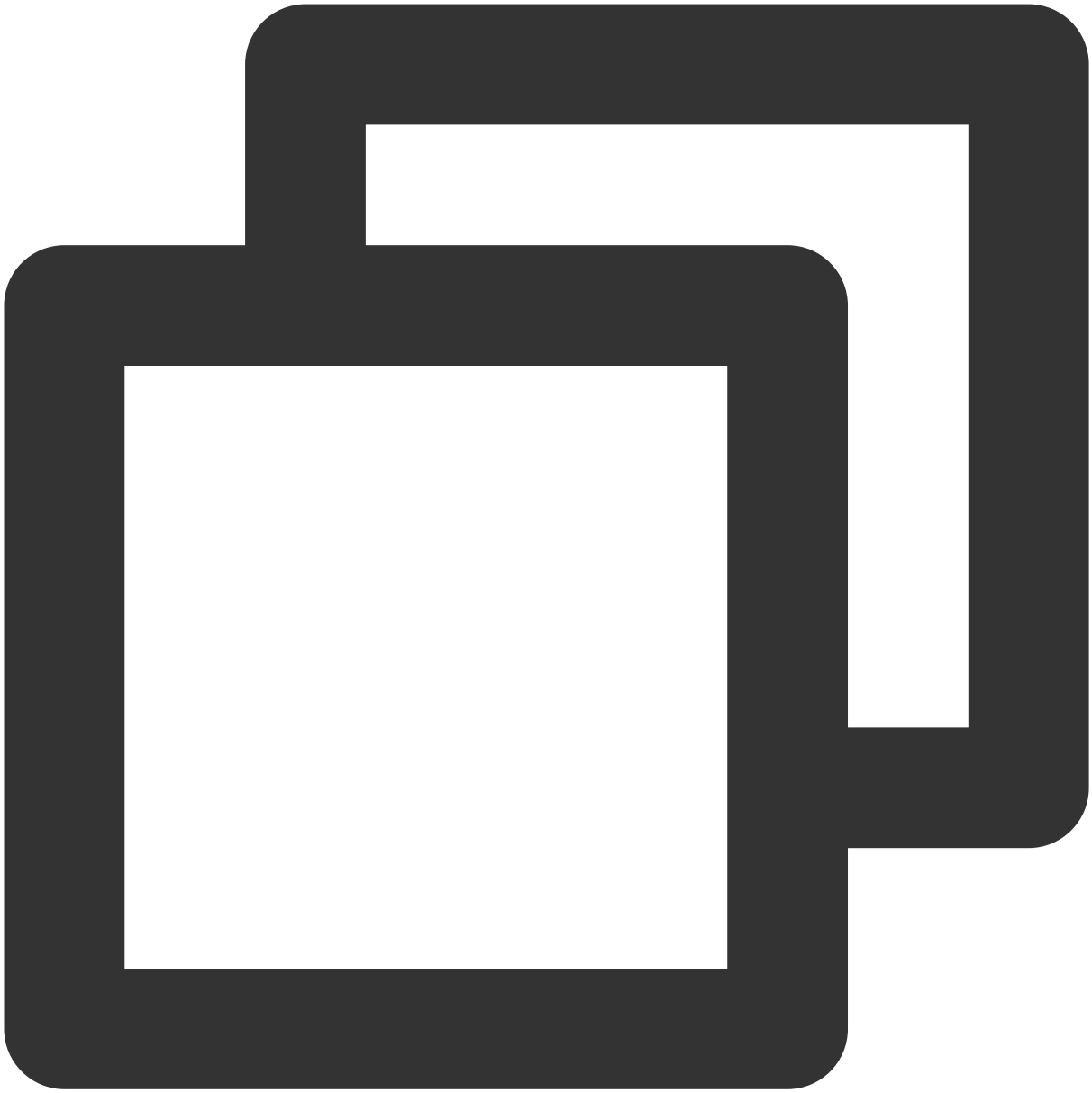
4. Concatenate the Authorization string

Concatenate the `Authorization` string in the following format:

```
Authorization =
    Algorithm + ' ' +
    'Credential=' + SecretId + '/' + CredentialScope + ', ' +
    'SignedHeaders=' + SignedHeaders + ', ' +
    'Signature=' + Signature
```

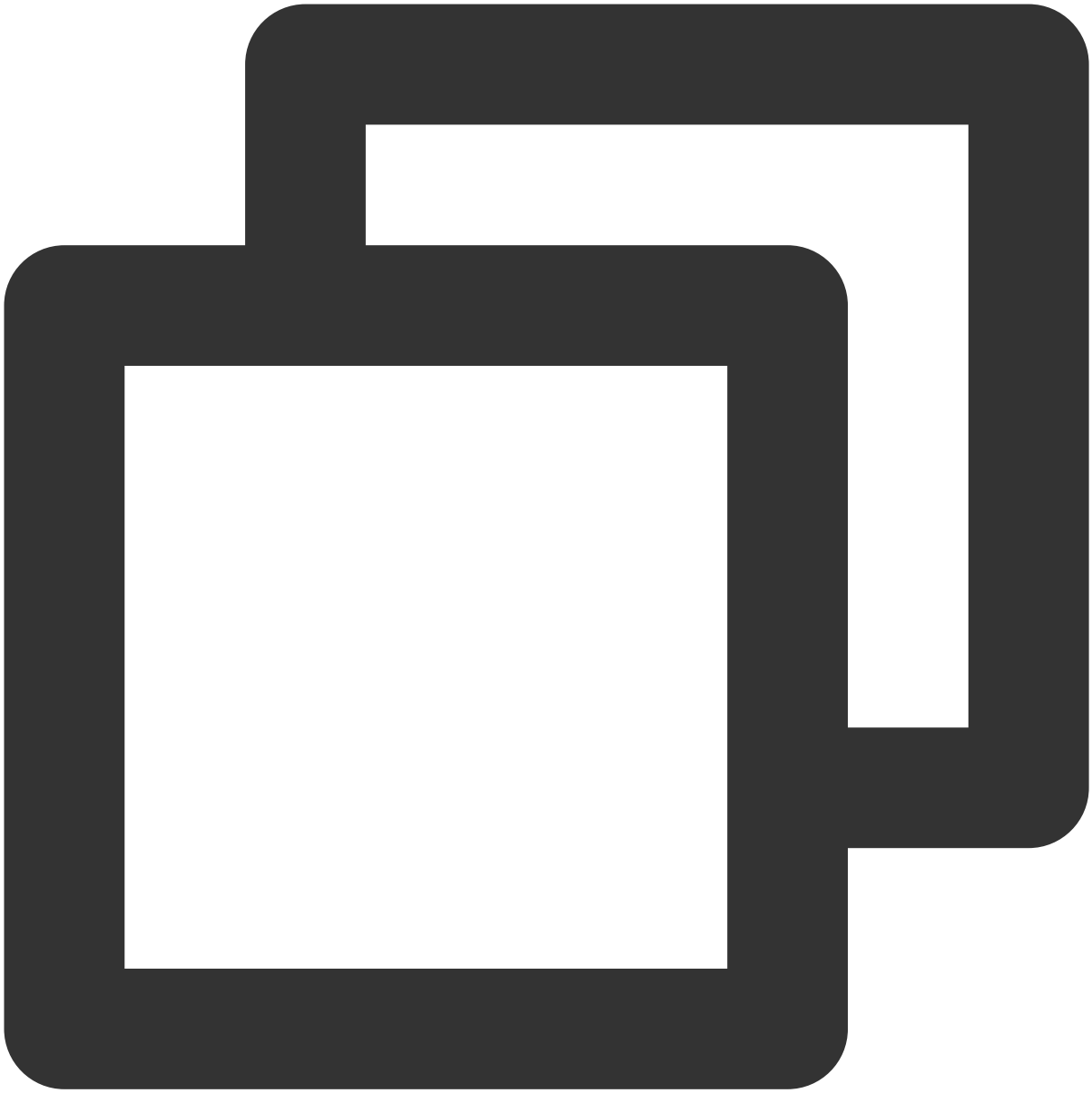
Field	Description
Algorithm	Signature algorithm, which is always `TC3-HMAC-SHA256` .
SecretId	`SecretId` in the key pair, i.e., `AKIDz8krbsJ5**********mLPx3EXAMPLE` .
CredentialScope	Credential scope (see above). The calculation result in this example is `2019-02-25/cvm/tc3_request` .
SignedHeaders	Header information for signature calculation (see above), such as `content-type;host` in this example.
Signature	Signature value. The calculation result in this example is `72e494ea8****************************************a96525168` .

According to the rules above, the values obtained in this example are:



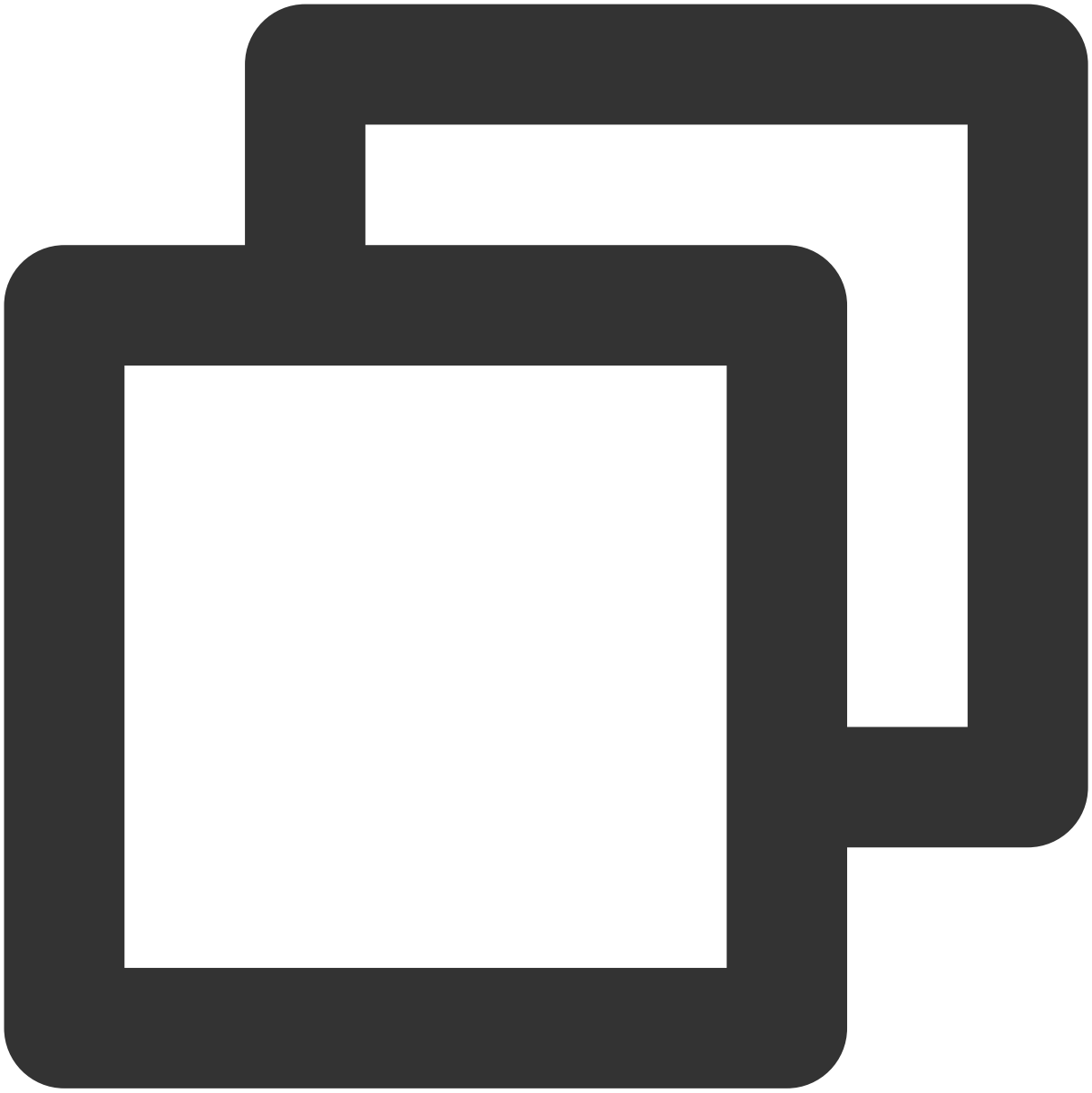
```
TC3-HMAC-SHA256 Credential=AKIDz8krbsJ5**********mLPx3EXAMPLE/2019-02-25/cvm/tc3_re
```

The complete call information is as follows:



```
POST https://cvm.tencentcloudapi.com/
Authorization: TC3-HMAC-SHA256 Credential=AKIDz8krbsJ5**********mLPx3EXAMPLE/2019-0
Content-Type: application/json; charset=utf-8
Host: cvm.tencentcloudapi.com
X-TC-Action: DescribeInstances
X-TC-Version: 2017-03-12
X-TC-Timestamp: 1551113065
X-TC-Region: ap-guangzhou

{"Limit": 1, "Filters": [{"Values": ["\\u672a\\u547d\\u540d"], "Name": "instance-na
```

5. Sample API 3.0 signature v3



```
const crypto = require('crypto');

function sha256(message, secret = '', encoding) {
    const hmac = crypto.createHmac('sha256', secret)
    return hmac.update(message).digest(encoding)
}
function getHash(message, encoding = 'hex') {
    const hash = crypto.createHash('sha256')
    return hash.update(message).digest(encoding)
```

```
}
function getDate(timestamp) {
    const date = new Date(timestamp * 1000)
    const year = date.getUTCFullYear()
    const month = ('0' + (date.getUTCMonth() + 1)).slice(-2)
    const day = ('0' + date.getUTCDate()).slice(-2)
    return `${year}-${month}-${day}`
}
function main(){
    // Key parameter
    const SECRET_ID = "AKIDz8krbsJ5**********mLPx3EXAMPLE"
    const SECRET_KEY = "Gu5t9xGAR***********EXAMPLE"

    const endpoint = "cvm.tencentcloudapi.com"
    const service = "cvm"
    const region = "ap-guangzhou"
    const action = "DescribeInstances"
    const version = "2017-03-12"
    //const timestamp = getTime()
    const timestamp = 1551113065
    // Process to get a UTC date
    const date = getDate(timestamp)

    // ************* Step 1. Concatenate the canonical request string *************
    const signedHeaders = "content-type;host"

    const payload = "{\\"Limit\\": 1, \\"Filters\\": [{\\"Values\\": [\\"\\\\u672a\

    const hashedRequestPayload = getHash(payload);
    const httpRequestMethod = "POST"
    const canonicalUri = "/"
    const canonicalQueryString = ""
    const canonicalHeaders = "content-type:application/json; charset=utf-8\\n" + "h

    const canonicalRequest = httpRequestMethod + "\\n"
                        + canonicalUri + "\\n"
                        + canonicalQueryString + "\\n"
                        + canonicalHeaders + "\\n"
                        + signedHeaders + "\\n"
                        + hashedRequestPayload
    console.log(canonicalRequest)
    console.log("----------------------------")

    // ************* Step 2. Concatenate the string to sign *************
    const algorithm = "TC3-HMAC-SHA256"
    const hashedCanonicalRequest = getHash(canonicalRequest);
    const credentialScope = date + "/" + service + "/" + "tc3_request"
```

```
    const stringToSign = algorithm + "\\n" +
                    timestamp + "\\n" +
                    credentialScope + "\\n" +
                    hashedCanonicalRequest
    console.log(stringToSign)
    console.log("----------------------------")

    // ************* Step 3. Calculate the signature *************
    const kDate = sha256(date, 'TC3' + SECRET_KEY)
    const kService = sha256(service, kDate)
    const kSigning = sha256('tc3_request', kService)
    const signature = sha256(stringToSign, kSigning, 'hex')
    console.log(signature)
    console.log("----------------------------")

    // ************* Step 4. Concatenate the `Authorization` string *************
    const authorization = algorithm + " " +
                    "Credential=" + SECRET_ID + "/" + credentialScope + ", " +
                    "SignedHeaders=" + signedHeaders + ", " +
                    "Signature=" + signature
    console.log(authorization)
    console.log("----------------------------")

    const Call_Information = 'curl -X POST ' + "https://" + endpoint
                          + ' -H "Authorization: ' + authorization + '"'
                          + ' -H "Content-Type: application/json; charset=utf-8"'
                          + ' -H "Host: ' + endpoint + '"'
                          + ' -H "X-TC-Action: ' + action + '"'
                          + ' -H "X-TC-Timestamp: ' + timestamp.toString() + '"'
                          + ' -H "X-TC-Version: ' + version + '"'
                          + ' -H "X-TC-Region: ' + region + '"'
                          + " -d '" + payload + "'"
    console.log(Call_Information)
}
main()
```

2. Get an API 3.0 signature v1

The signature algorithm v1 is simple and easy to use, but its functionality and security are not as good as the signature algorithm v3 which is therefore recommended.

Note:

If you are using the signature algorithm for the first time, we recommend you use the "signature string generation" feature in API Explorer and select "API 3.0 signature v1" as the signature version, which can generate a signature for demonstration and verification and provides signing examples for certain programming languages. Plus, it can also

generate SDK code directly. Seven common open-source programming language SDKs are available for TencentCloud API, including Python, Java, PHP, Go, Node.js, .NET, and C++.

For example, if you call the `DescribeInstances` API to query CVM instances, the request parameters may be as follows:

Parameter Name	Description	Value
Action	Method	DescribeInstances
SecretId	Key ID	`AKIDz8krbsJ5**********mLPx3EXAMPLE`
Timestamp	Current timestamp	1465185768
Nonce	Random positive integer	11886
Region	Instance region	ap-guangzhou
InstanceIds.0	ID of the instance to be queried	ins-09dx96dg
Offset	Offset	0
Limit	Allowed maximum number of output entries	20
Version	API version number	2017-03-12

1. Sort parameters

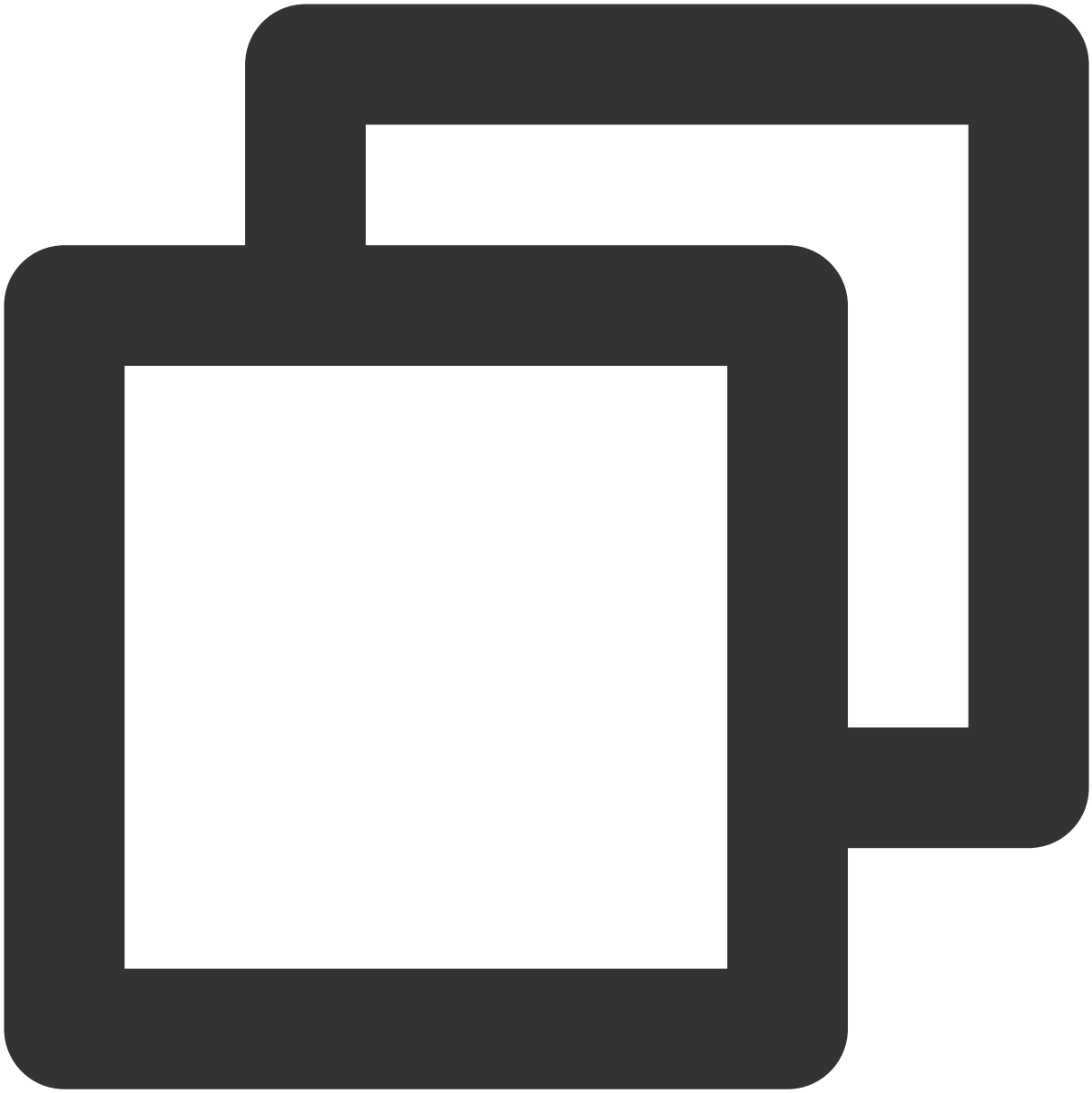
Sort all the request parameters in an ascending lexicographical order (ASCII code) by their names.

Note:

The parameters are sorted only by name but not by value.

The parameters are sorted based on ASCII code but not in an alphabetical order or by value. For example, `InstanceIds.2` should be arranged behind `InstanceIds.12`. You can complete sorting by using a sorting function in a programming language, such as the `ksort` function in PHP.

The parameters in the example are sorted as follows:



```
{
    'Action' : 'DescribeInstances',
    'InstanceIds.0' : 'ins-09dx96dg',
    'Limit' : 20,
    'Nonce' : 11886,
    'Offset' : 0,
    'Region' : 'ap-guangzhou',
    'SecretId' : 'AKIDz8krbsJ5**********mLPx3EXAMPLE',
    'Timestamp' : 1465185768,
    'Version': '2017-03-12',
}
```

Any other programming languages can be used to sort these parameters as long as the same result is produced.

2. Concatenate the canonical request string

This step generates a request string. Format the request parameters sorted in the previous step into the form of `parameter=value`. For example, for the `Action` parameter, its parameter is `Action` and its value is `DescribeInstances`; therefore, the parameter will be formatted into `Action=DescribeInstances`.

Note:

The `value` is the original value instead of the URL-encoded value.

Then, concatenate the formatted parameters with `&`. The generated request string will be as follows:

```
Action=DescribeInstances&InstanceIds.0=ins-09dx96dg&Limit=20&Nonce=11886&Offset=0&R
```

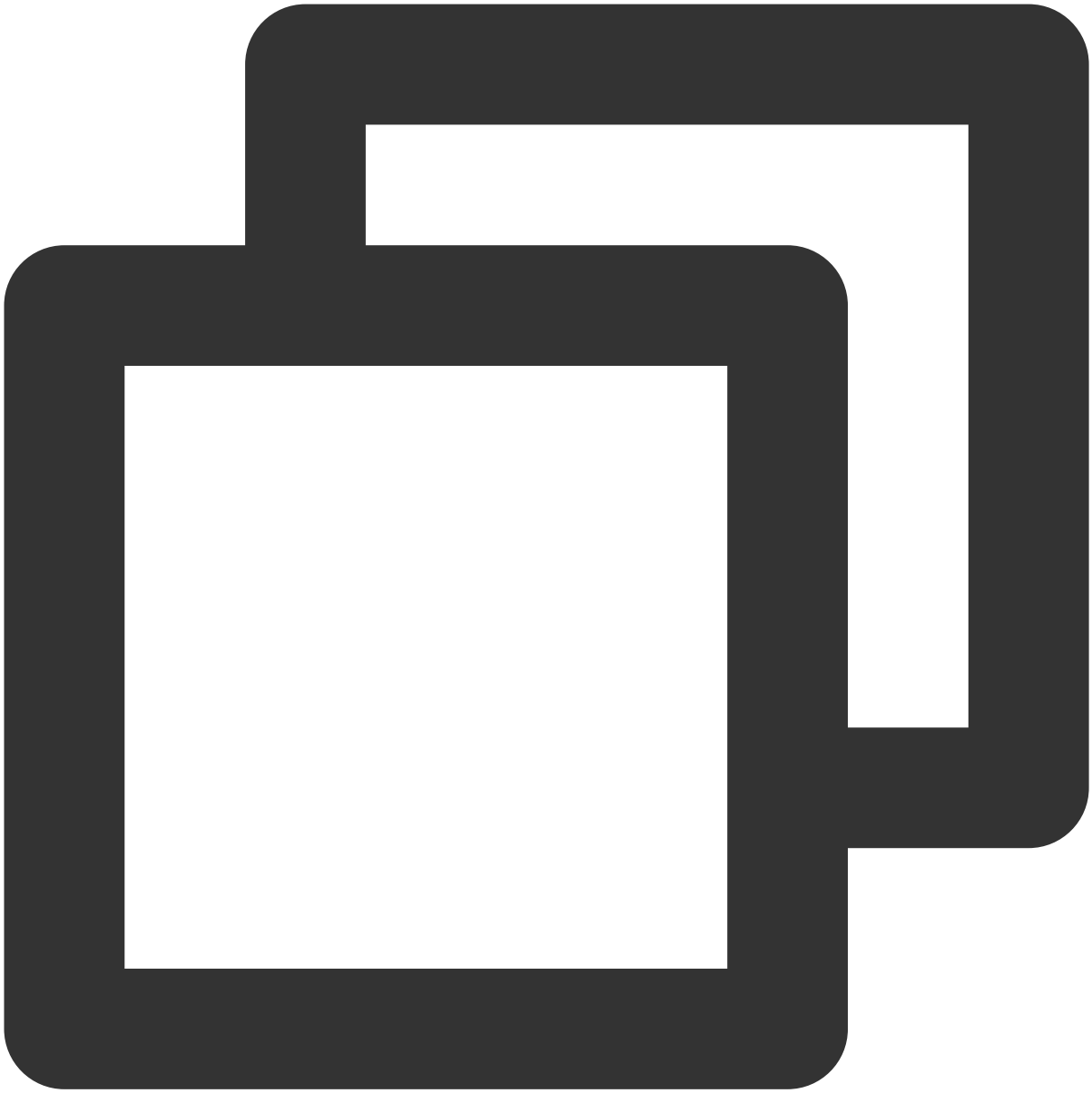
3. Concatenate the string to sign

This step generates the original signature string. The original signature string consists of the following parameters:

1. Request method: POST and GET methods are supported. GET is used here for the request. Please note that the method name should be in all capital letters.
2. Request server: the domain name of the request for querying instances (DescribeInstances) is `cvm.tencentcloudapi.com` . The actual request domain name varies by the module to which the API belongs. For more information, please see the specific API document.
3. Request path: the request path in the current version of TencentCloud API is fixed to `/` .
4. Request string: the request string generated in the previous step.

The rule for concatenating the original string of the signature is `request method + request server + request path + ? + request string` .

The concatenation result in the example is as follows:

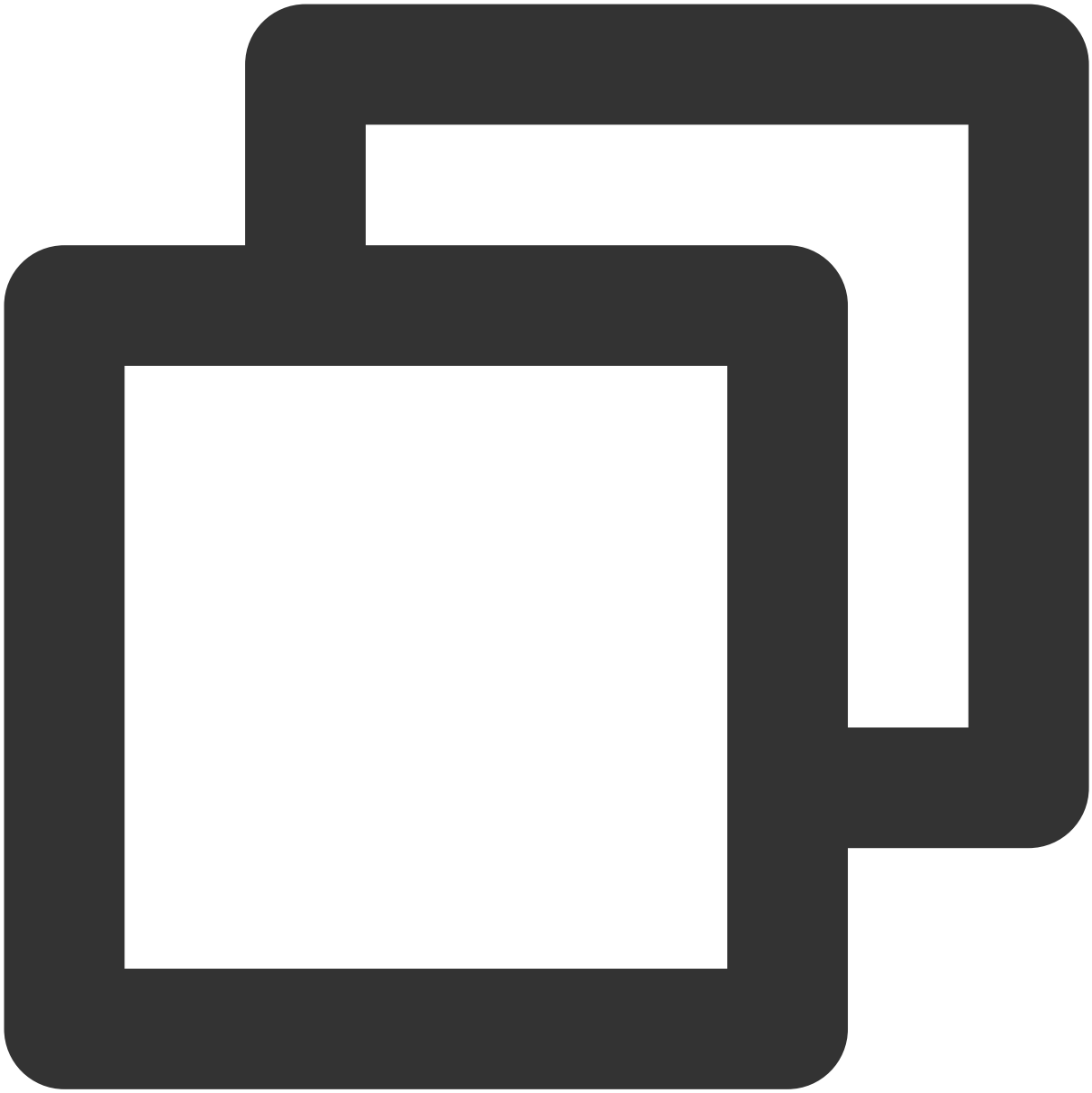


```
GETcvm.tencentcloudapi.com/?Action=DescribeInstances&InstanceIds.0=ins-09dx96dg&Lim
```

4. Calculate the signature

This step generates a signature string. Use the HMAC-SHA1 algorithm to sign the **original signature string** obtained in the previous step, and then Base64-encode the generated signature to get the final signature. For more information, please see the sample signature below.

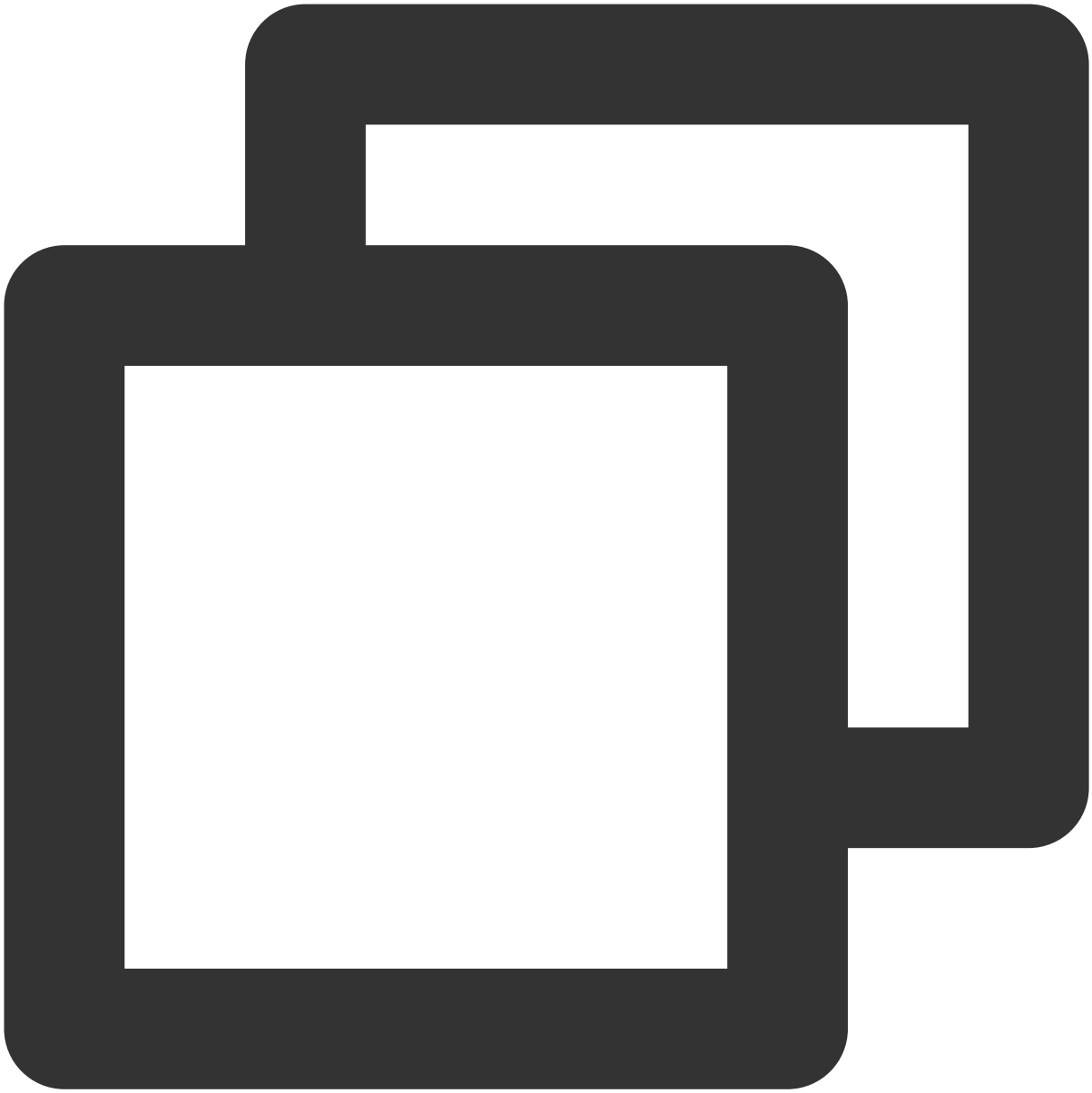
The obtained signature string is as follows:



```
Eli*****************cGeI=
```

5. Get the call information and send a request

```
# The API will be called actually, and fees will be incurred if it is a consumption
resp = requests.get("https://" + endpoint, params=data)
print(resp.url)
```



```
The obtained request string is as follows:
https://cvm.tencentcloudapi.com/?Action=DescribeInstances&InstanceIds.0=ins-09dx96d
```

Field	Description
endpoint	Service address, such as `cvm.tencentcloudapi.com` .
data	API parameter of the sample API 3.0 signature v1. **Note: you should add the calculated signature in the format of key-value pair to `data` .**

Note:

The key in the example is not real, and the timestamp is not the current system time. If you open this URL in the browser or call it by using commands such as `curl` , an authentication error `The signature expired` will be returned. To obtain a URL that works, you need to replace the `SecretId` and `SecretKey` in this example with your own credentials and use the current system time as the `Timestamp` .

To further explain the signing process, Node.js is used as examples below to implement the process as described above. The request domain name, API, and parameter values in the above example are used here. The code below is for demonstration only. Please use the SDK for actual development.

6. Encode a signature string

The generated signature string cannot be directly used as a request parameter and needs to be URL-encoded.

For example, if the signature string generated in the previous step is `Eli****************cGeI=` , the final value of the `Signature` request parameter will be `EliP**********************eI%3D` , which will be used to generate the final request URL.

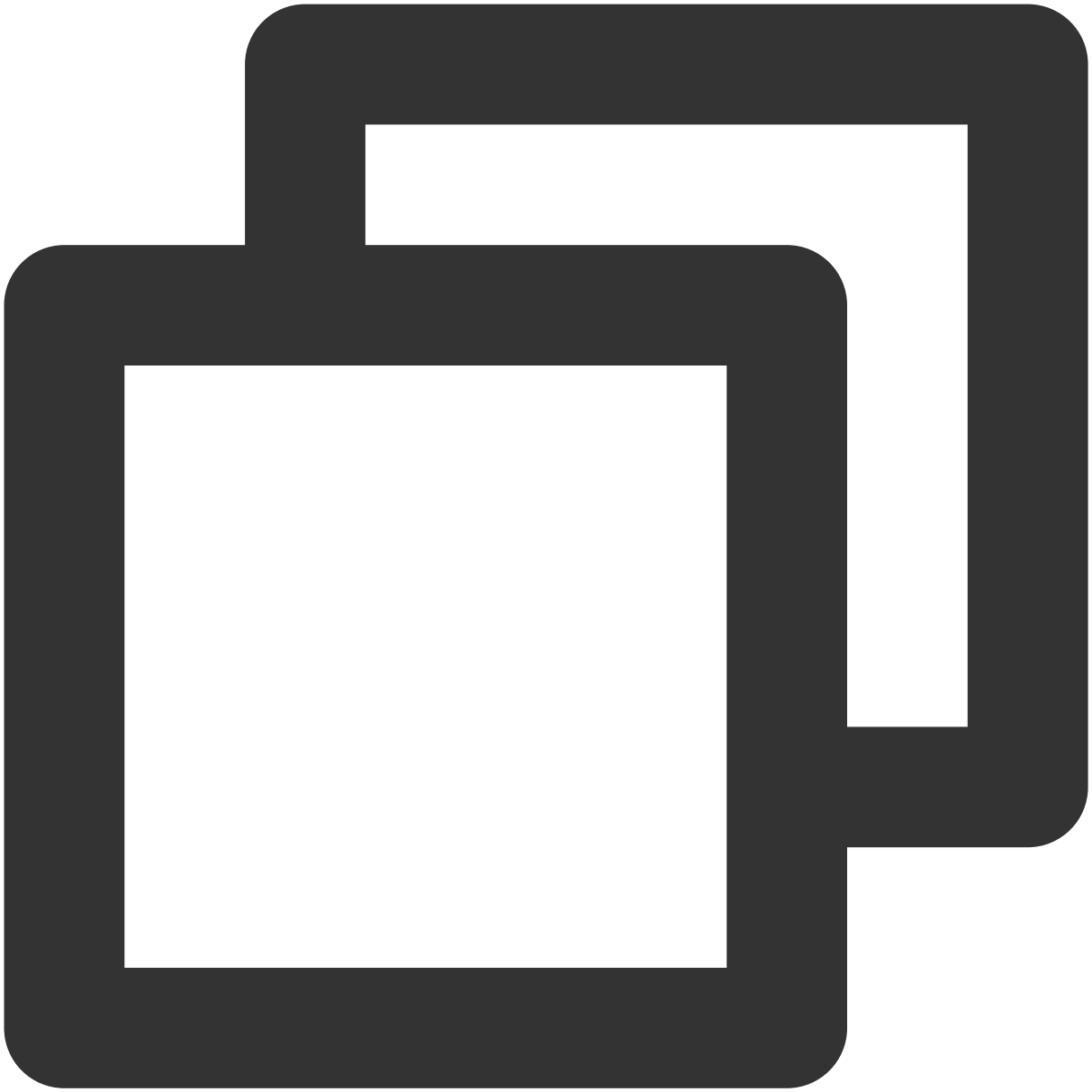
Note:

If you use the GET request method or use the POST request method with `Content-Type` of `application/x-www-form-urlencoded` , all the request parameter values must be URL-encoded (except the parameter key and the equal symbol (=)) before the request is sent. Non-ASCII characters must be encoded with UTF-8 before URL-encoding.

The network libraries of some programming languages automatically URL-encode all parameters. In this case, the signature string does not need to be URL-encoded again; otherwise, two rounds of URL-encoding will cause the signature to fail.

Other parameter values also need to be encoded with RFC 3986. Use `%XY` in percent-encoding for special characters such as Chinese characters, where "X" and "Y" are hexadecimal characters (0–9 and uppercase A–F). Using lowercase characters will cause an error.

7. Sample API 3.0 signature v1



```
const crypto = require('crypto');

function get_req_url(params, endpoint){
    params['Signature'] = escape(params['Signature']);
    const url_strParam = sort_params(params)
    return "https://" + endpoint + "/?" + url_strParam.slice(1);
}

function formatSignString(reqMethod, endpoint, path, strParam){
    let strSign = reqMethod + endpoint + path + "?" + strParam.slice(1);
    return strSign;
```

```
}
function sha1(secretKey, strsign){
    let signMethodMap = {'HmacSHA1': "sha1"};
    let hmac = crypto.createHmac(signMethodMap['HmacSHA1'], secretKey || "");
    return hmac.update(Buffer.from(strsign, 'utf8')).digest('base64')
}

function sort_params(params){
    let strParam = "";
    let keys = Object.keys(params);
    keys.sort();
    for (let k in keys) {
        //k = k.replace(/_/g, '.');
        strParam += ("&" + keys[k] + "=" + params[keys[k]]);
    }
    return strParam
}

function main(){
    // Key parameter
    const SECRET_ID = "AKIDz8krbsJ5**********mLPx3EXAMPLE"
    const SECRET_KEY = "Gu5t9xGAR***********EXAMPLE"

    const endpoint = "cvm.tencentcloudapi.com"
    const Region = "ap-guangzhou"
    const Version = "2017-03-12"
    const Action = "DescribeInstances"
    const Timestamp = 1465185768  // Timestamp: 2016-06-06 12:02:48. This parameter
    // const Timestamp = Math.round(Date.now() / 1000)
    const Nonce = 11886  // A random positive integer
    //const nonce = Math.round(Math.random() * 65535)

    let params = {};
    params['Action'] = Action;
    params['InstanceIds.0'] = 'ins-09dx96dg';
    params['Limit'] = 20;
    params['Offset'] = 0;
    params['Nonce'] = Nonce;
    params['Region'] = Region;
    params['SecretId'] = SECRET_ID;
    params['Timestamp'] = Timestamp;
    params['Version'] = Version;

    // 1. Sort the parameters and concatenate a request string
    strParam = sort_params(params)

    // 2. Concatenate an original signature string
```

```
    const reqMethod = "GET";
    const path = "/";
    strSign = formatSignString(reqMethod, endpoint, path, strParam)
    console.log(strSign)
    console.log("-----------------------")

    // 3. Generate a signature string
    params['Signature'] = sha1(SECRET_KEY, strSign)
    console.log(params['Signature'])
    console.log("-----------------------")

    // 4. Perform URL-encoding and concatenate a request URL
    const req_url = get_req_url(params, endpoint)
    console.log(params['Signature'])
    console.log("-----------------------")
    console.log(req_url)
}
main()
```

API 2.0 Signature

This signature version has been disused. We recommend you use **API 3.0 signature** with better performance. If you still need to use it, please go to API Explorer > **Signature Generation** and select **API 2.0 Signature** as the signature version.

Signature Failure

The following error codes may be returned for signature failure. Please resolve the errors accordingly.

Error Code	Error Description
AuthFailure.SignatureExpire	The signature expired. The difference between the `Timestamp` and the server time cannot be greater than five minutes.
AuthFailure.SecretIdNotFound	The key does not exist. Log in to the console and check whether it is disabled or you copied fewer or more characters.
AuthFailure.SignatureFailure	Signature error. It is possible that the signature is calculated incorrectly, the signature does not match the content that is actually sent, or the `SecretKey` is incorrect.
AuthFailure.TokenFailure	Temporary credential token error.

AuthFailure.InvalidSecretId	Invalid key (not TencentCloud API key type).

Returned Result

Successful response

For example, when calling the CVM API `DescribeInstancesStatus` (version: 2017-03-12) to view the status of instances, if the request succeeds, you may see the following response:

```
{
    "Response": {
        "TotalCount": 0,
        "InstanceStatusSet": [],
        "RequestId": "b5b41468-520d-4192-b42f-595cc34b6c1c"
    }
}
```

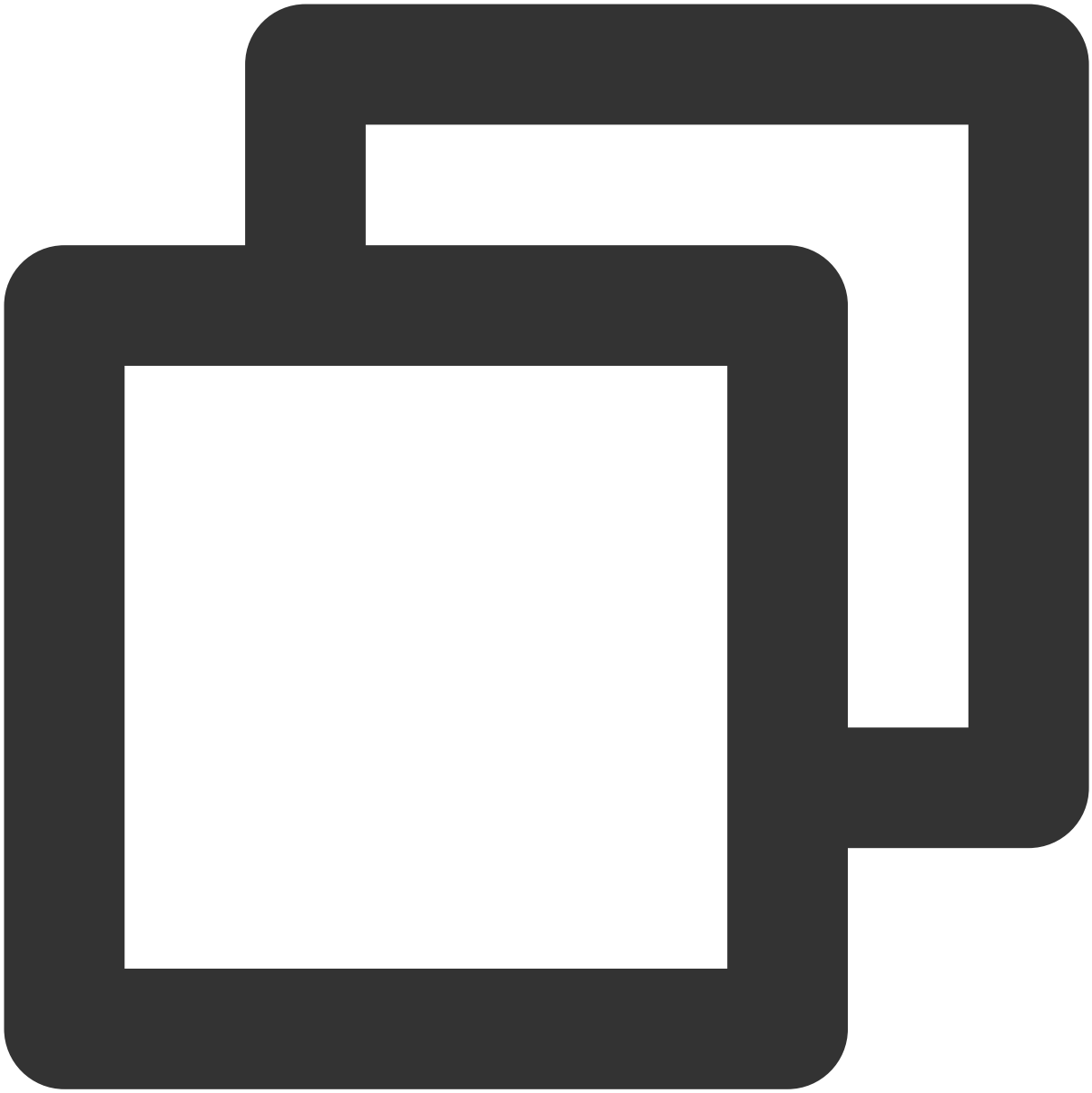
The API will return `Response`, which contains `RequestId`, as long as it processes the request, no matter whether the request is successful or not.

`RequestId` is the unique ID of an API request. It is required to troubleshoot issues.

Any fields other than the common fields are API-specific. For more information on such fields, please see the relevant API documentation. In this example, both `TotalCount` and `InstanceStatusSet` are specific to the `DescribeInstancesStatus` API. Since the user who initiated the request does not have a CVM instance yet, 0 is returned for `TotalCount` and `InstanceStatusSet` is empty.

Error response

If the call fails, you may see the following response:




```
{
    "Response": {
        "Error": {
            "Code": "AuthFailure.SignatureFailure",
            "Message": "The provided credentials could not be validated. Please che
        },
        "RequestId": "ed93f3cb-f35e-473f-b9f3-0d451b8b79c6"
    }
}
```

`Error` indicates that the request failed. A response for a failed request will always include the `Error`, `Code`, and `Message` fields.
`Code` indicates the specific error code, which is returned when an API request failed. You can use this code to locate the cause and solution of the error in the common or API-specific error code list.
`Message` explains the cause of the error. Note that the returned messages are subject to service updates. The information the messages provide may not be up-to-date and should not be the only source of reference.
`RequestId` is the unique ID of an API request. It is required to troubleshoot issues.

Common error codes

The `Error` field in a response indicates that the API call failed. The `Code` field in `Error` indicates the error code. The following table lists the common error codes that any services may return.

Error Code	Error Description
AuthFailure.InvalidSecretId	Invalid key (not TencentCloud API key type).
AuthFailure.MFAFailure	MFA failure.
AuthFailure.SecretIdNotFound	The key does not exist.
AuthFailure.SignatureExpire	The signature expired.
AuthFailure.SignatureFailure	Signature error.
AuthFailure.TokenFailure	Token error.
AuthFailure.UnauthorizedOperation	No CAM authorization.
DryRunOperation	DryRun Operation. It means that the request would have succeeded, but the DryRun parameter was used.
FailedOperation	The operation failed.
InternalError	Internal error.
InvalidAction	The API does not exist.
InvalidParameter	Incorrect parameter.
InvalidParameterValue	Invalid parameter value.
LimitExceeded	The quota limit is exceeded.
MissingParameter	A parameter is missing.
NoSuchVersion	The API version does not exist.

RequestLimitExceeded	The request rate limit is exceeded.
ResourceInUse	The resource is in use.
ResourceInsufficient	Insufficient resource.
ResourceNotFound	The resource does not exist.
ResourceUnavailable	The resource is unavailable.
UnauthorizedOperation	Unauthorized operation.
UnknownParameter	Unknown parameter error.
UnsupportedOperation	Unsupported operation.
UnsupportedProtocol	Unsupported HTTPS request method. Only GET and POST requests are supported.
UnsupportedRegion	Unsupported region.

API for PHP

Last updated：2023-03-07 18:16:40

TencentCloud API has been upgraded to v3.0. This version is optimized for performance and deployed in all regions. It supports nearby access and access by region for significantly reduced access latency. In addition, it features more detailed API descriptions and error codes and API-level comments for SDKs, enabling you to use Tencent Cloud services more conveniently and quickly. This document describes how to call APIs for PHP.
This version currently supports various Tencent Cloud services such as CVM, CBS, VPC, and TencentDB and will support more services in the future.

Request Structure

1. Service address (endpoint)

TencentCloud API supports access from either a nearby region (such as `cvm.tencentcloudapi.com` for CVM) or a specified region (such as `cvm.ap-guangzhou.tencentcloudapi.com` for CVM in the Guangzhou region). For values of the region parameter, please see the region list in the "Common Parameters" section below. To check whether a region is supported by a specific Tencent Cloud service, please see its "Request Structure" document.

Note:

For latency-sensitive businesses, we recommend you specify a domain name with a region.

2. Communications protocol

All TencentCloud APIs communicate over HTTPS, providing highly secure communications tunnels.

3. Request method

Supported HTTP request methods:
POST (recommended)
GET
`Content-Type` types supported by POST request:
application/json (recommended). The signature algorithm v3 (TC3-HMAC-SHA256) must be used.
application/x-www-form-urlencoded. The signature algorithm v1 (HmacSHA1 or HmacSHA256) must be used.
multipart/form-data (only supported by certain APIs). The signature algorithm v3 (TC3-HMAC-SHA256) must be used.
The size of a GET request packet cannot exceed 32 KB. The size of a POST request cannot exceed 1 MB for the signature algorithm v1 (HmacSHA1 or HmacSHA256) or 10 MB for the signature algorithm v3 (TC3-HMAC-SHA256).

4. Character encoding

UTF-8 encoding is always used.

Common Parameters

Note:
The common parameters are used to identity the user and API signature. They should be carried by each request to initiate properly.

Signature algorithm v3

The signature algorithm v3 (sometimes referred to as "TC3-HMAC-SHA256") is more secure than the signature algorithm v1 (referred to as signature algorithm in certain documents), supports larger request packets and POST JSON format, and has a higher performance. We recommend you use it to calculate signatures. For more information on how to use it, please see below.

Parameter Name	Type	Required	Description
X-TC-Action	String	**Yes**	Name of the API for the desired operation. For the specific value, please see the description of common parameter `Action` in the input parameters in the related API document. For example, the API for querying CVM instance list is `DescribeInstances` .
X-TC-Region	String	-	Region parameter, which is used to identify the region where the data you want to manipulate resides. For values supported for an API, please see the description of common parameter `Region` in the input parameters in related API documentation. **Note: this parameter is not required for some APIs (which will be indicated in related API documentation) and will not take effect even if it is passed.**
X-TC-Timestamp	Integer	**Yes**	The current UNIX timestamp that records the time when the API request was initiated, such as 1529223702. **Note: if the difference between the UNIX timestamp and the server time is greater than 5 minutes, a signature expiration error may occur.**
X-TC-Version	String	**Yes**	Version of the API for the desired operation, such as 2017-03-12 for CVM. For the specific value, please see the description of common parameter `Version` in the input parameters in related API documentation.
Authorization	String	**Yes**	HTTP authentication request header, such as TC3-HMAC-SHA256 Credential=AKIDEXAMPLE/Date/service/tc3_request,

			SignedHeaders=content-type;host, Signature=72e494ea8**********a96525168 Here, TC3-HMAC-SHA256: signature algorithm, currently fixed as this value. Credential: signature credential. `AKIDEXAMPLE` indicates the `SecretId`. `Date` indicates a UTC date which must match the value of `X-TC-Timestamp` (a common parameter) in UTC format. `service` indicates the name of the service and is generally a domain name prefix; for example, the domain name `cvm.tencentcloudapi.com` means the CVM service, and the value for this service is `cvm`. SignedHeaders: the headers that contain the authentication information. `content-type` and `host` are required. Signature: signature digest. For the calculation process, please see below.
X-TC-Token	String	No	Token used for temporary credentials. It must be used with a temporary key. You can get the temporary key and token by calling a CAM API. No token is required for a long-term key.

Signature algorithm v1

When the signature algorithm v1 (sometimes referred to as "HmacSHA256" or "HmacSHA1") is used, the common parameters should be uniformly placed in the request string.

Parameter Name	Type	Required	Description
Action	String	Yes	Name of the API for the desired operation. For the specific value, please see the description of common parameter `Action` in the input parameters in the related API document. For example, the API for querying CVM instance list is `DescribeInstances`.
Region	String	-	Region parameter, which is used to identify the region where the data you want to manipulate resides. For values supported for an API, please see the description of common parameter `Region` in the input parameters in related API documentation. **Note: this parameter is not required for some APIs (which will be indicated in related API documentation) and will not take effect even if it is passed.**
Timestamp	Integer	Yes	The current UNIX timestamp that records the time when the API request was initiated, such as 1529223702. If the

			difference between the UNIX timestamp and the current time is too large, a signature expiration error may occur.
Nonce	Integer	Yes	A random positive integer used in conjunction with `Timestamp` to prevent replay attacks.
SecretId	String	Yes	The identifying `SecretId` obtained on the TencentCloud API Key page. A `SecretId` corresponds to a unique `SecretKey` which is used to generate the request signature (`Signature`).
Signature	String	Yes	Request signature, which is used to verify the validity of the request. It is generated based on input parameters. For more information on how to calculate the signature, please see below.
Version	String	Yes	Version of the API for the desired operation, such as 2017-03-12 for CVM. For the specific value, please see the description of common parameter `Version` in the input parameters in related API documentation.
SignatureMethod	String	No	Signature algorithm. Currently, only HmacSHA256 and HmacSHA1 are supported. The HmacSHA256 algorithm is used to verify the signature only when this parameter is specified as HmacSHA256. In other cases, the signature is verified with HmacSHA1.
Token	String	No	Token used for temporary credentials. It must be used with a temporary key. You can get the temporary key and token by calling a CAM API. No token is required for a long-term key.

Region list

As the supported regions vary by service, please refer to the region list in each service's product documentation for specific details.
For example, you can see the region list of CVM.

API Call Method for PHP

TencentCloud API authenticates every request, that is, the request must be signed with the security credentials in the designated steps. Each request must contain the signature information in the common request parameters and be sent in the specified way and format.

Step 1. Apply for security credentials

In this document, the security credential used is a key pair, which consists of a `SecretId` and a `SecretKey`. Each user can have up to two key pairs.

SecretId: identifies the user that calls an API, which is similar to a username.

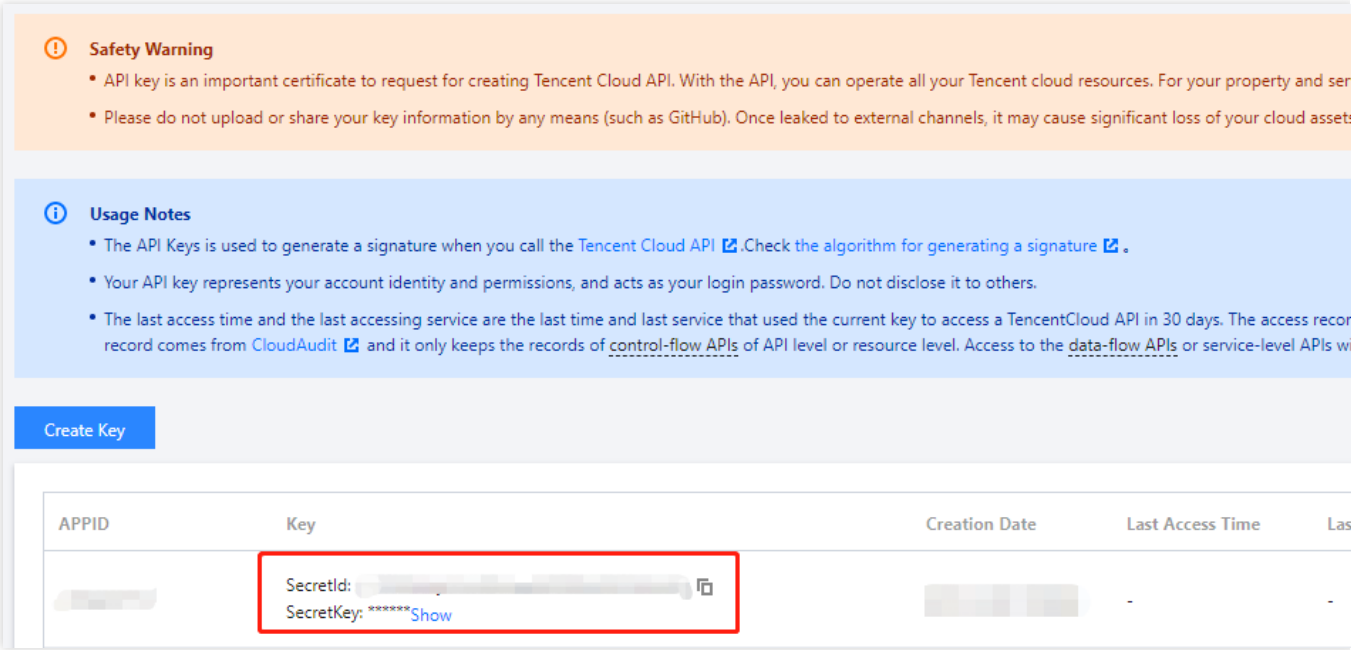
SecretKey: authenticates the user that calls the API, which is similar to a password.

Note:

You must keep your security credentials private and avoid disclosure; otherwise, your assets may be compromised. If they are disclosed, please disable them as soon as possible.

Suppose your `SecretId` and `SecretKey` are `AKIDz8krbsJ5*********mLPx3EXAMPLE` and `Gu5t9xGAR**********EXAMPLE`, respectively. If you want to view the status of an unnamed instance in the Guangzhou region and have only one data entry returned, the request may be:

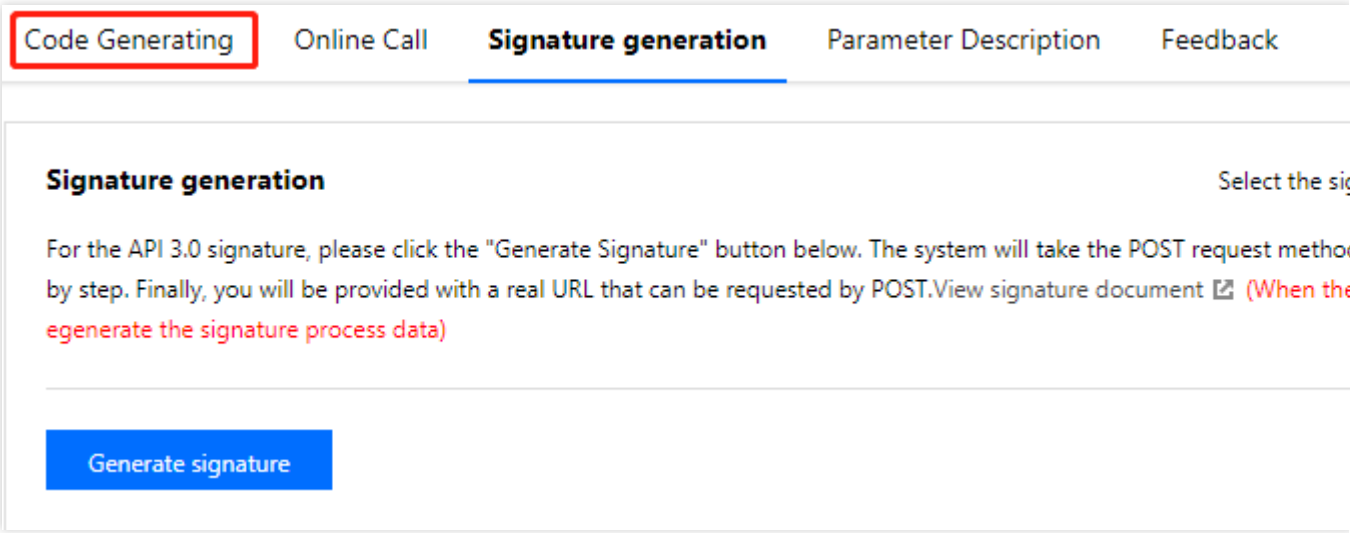
Go to the API key management page to get API keys as shown below:



Step 2

1. Get an API 3.0 signature v3

The signature algorithm v3 (TC3-HMAC-SHA256) is compatible with the previous signature algorithm v1 and more secure, supports larger request packets and POST JSON format, and has a higher performance. We recommend you use it to calculate signatures.



Note:

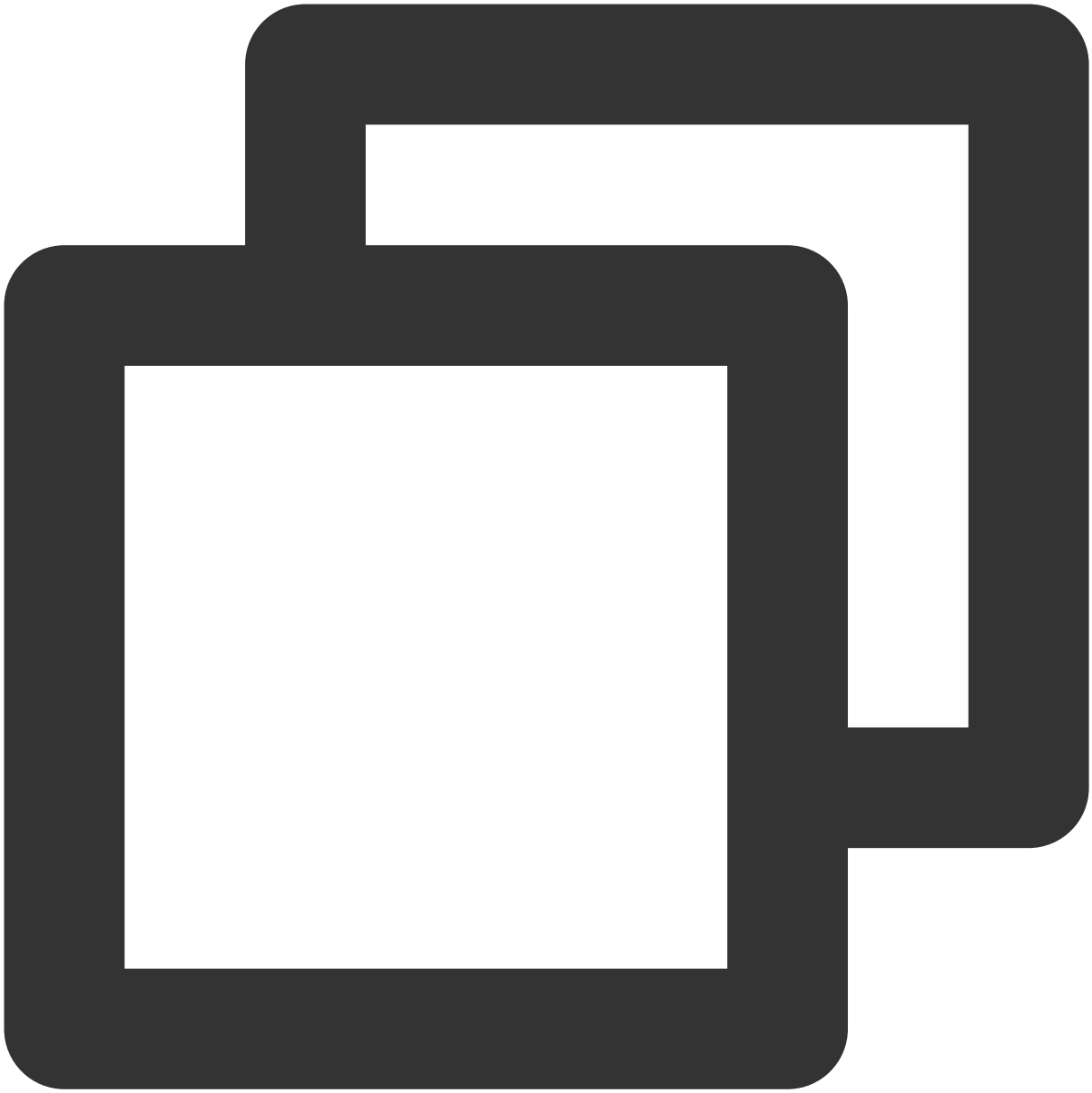
If you are using the signature algorithm for the first time, we recommend you use the "signature string generation" feature in API Explorer and select "API 3.0 signature v3" as the signature version, which can generate a signature for demonstration and verification. Plus, it can also generate SDK code directly. Seven common open-source programming language SDKs are available for TencentCloud API, including Python, Java, PHP, Go, Node.js, .NET, and C++.

TencentCloud API supports both GET and POST requests. For the GET method, only the `Content-Type: application/x-www-form-urlencoded` protocol format is supported. For the POST method, `Content-Type: application/json` and `Content-Type: multipart/form-data` are supported. The JSON format is supported by all business APIs, while the multipart format is supported only by specific APIs (in this case, an API cannot be called in JSON format). For more information, please see the specific business API document. We recommend you use the POST method because the two methods generate the same results, but the GET method only supports request packets below 32 KB in size.

The following describes how to calculate a signature by calling the DescribeInstances API. This API is chosen because:

1. The CVM API is enabled by default, and this API is often used.
2. It is read-only and does not change the status of existing resources.
3. It covers many types of parameters so that it is easy to show how to use an array that contains data structures.

1. Concatenate the canonical request string

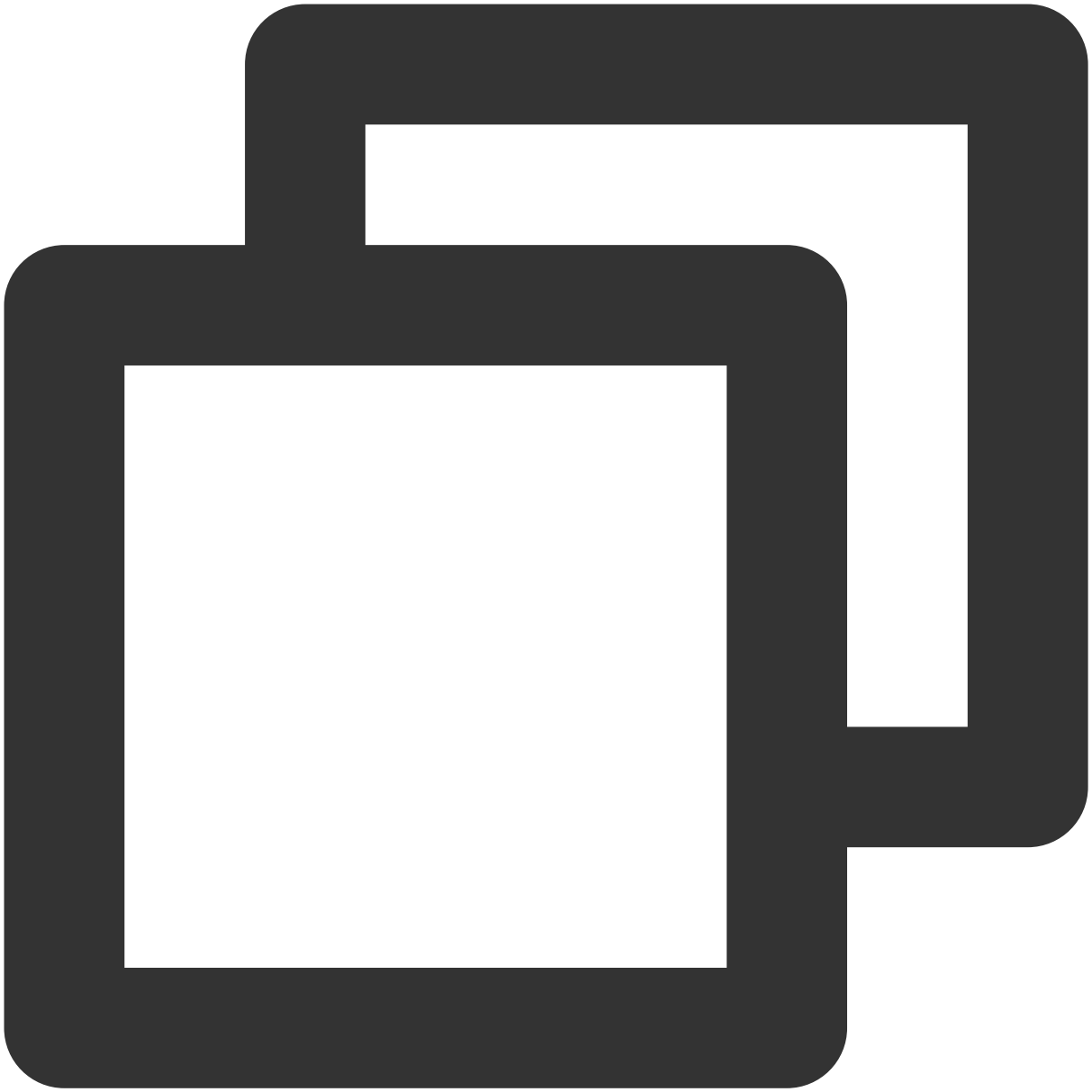


```
CanonicalRequest =
    HTTPRequestMethod + '\\n' +
    CanonicalURI + '\\n' +
    CanonicalQueryString + '\\n' +
    CanonicalHeaders + '\\n' +
    SignedHeaders + '\\n' +
    HashedRequestPayload
```

Field	Description

HTTPRequestMethod	HTTP request method (GET or POST). This example uses `POST` .
CanonicalURI	URI parameter. Slash ("/") is used for API 3.0.
CanonicalQueryString	Query string in the URL of the originating HTTP request. This is always an empty string POST requests and is the string after the question mark (?) for GET requests such as `Limit=10&Offset=0` . Note: `CanonicalQueryString` must be URL-encodec instructed in RFC 3986 with the UTF-8 character set. The applicable standard program language library is recommended. All special characters must be encoded and capitaliz
CanonicalHeaders	Header information for signature calculation, including at least `host` and `content type` . Custom headers can also be added to the signature process to improve the uniqueness and security of the request. Concatenation rules: both the key and value of header should be converted to lowercase with the leading and trailing spaces removed that they are concatenated in the `key:value\\n` format. If there are multiple heade they should be sorted in ASCII ascending order by header key (lowercase). The calcula result in this example is `content-type:application/json; charset=utf-8\\nhost:cvm.tencentcloudapi.com\\n` . Note: `content-type` must mat the content that is actually sent. In some programming languages, a `charset` value automatically added even if it is not specified. In this case, the request sent will be differ from the one signed, and the sever will return a signature verification failure.
SignedHeaders	Header information for signature calculation, indicating the request headers that are inv in the signature process. The request headers must correspond to the headers in `CanonicalHeaders` . `Content-type` and `host` are required headers. Concatenation rules: both the key and value of a header should be converted to lowerca there are multiple headers, they should be sorted in ASCII ascending order by header k (lowercase) and separated by semicolons (;). The value in this example is `content-type;host` .
HashedRequestPayload	Hash value of `Requestpayload` (i.e., the request body, such as `{"Limit": 1, "Filters": [{"Values": ["\\u672a\\u547d\\u540d"], "Name": "instance-name"}]}` in this example). The pseudo-code for calculation is `Lowercase(HexEncode(Hash.SHA256(RequestPayload)))` , which means t SHA256 hashing is performed on the payload of the HTTP request, then hexadecimal encoding is performed, and finally the encoded string is converted to lowercase letters. GET requests, `RequestPayload` is always an empty string. The calculation result i example is `35e9c5b0e3ae67532d3c9f17ead6c90222632e5b1ff7f6e89887f1398934f(`

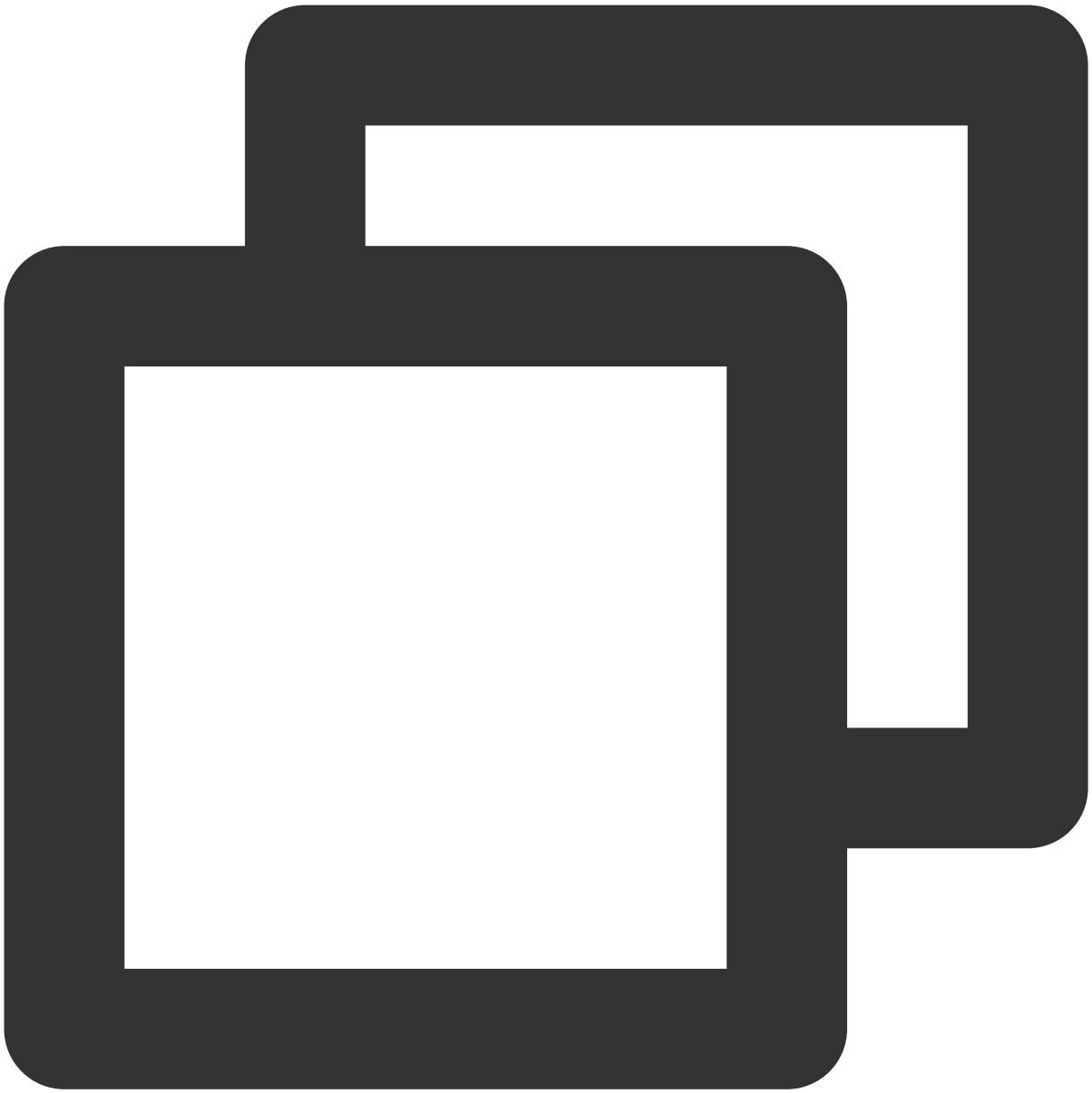
According to the rules above, the canonical request string obtained in the example is as follows:



```
POST
/

content-type:application/json; charset=utf-8
host:cvm.tencentcloudapi.com

content-type;host
35e9c5b0e3ae67532d3c9f17ead6c90222632e5b1ff7f6e89887f1398934f064
```

2. Concatenate the string to sign

Concatenate the string to sign in the following format:



```
StringToSign =
    Algorithm + \\n +
    RequestTimestamp + \\n +
    CredentialScope + \\n +
    HashedCanonicalRequest
```

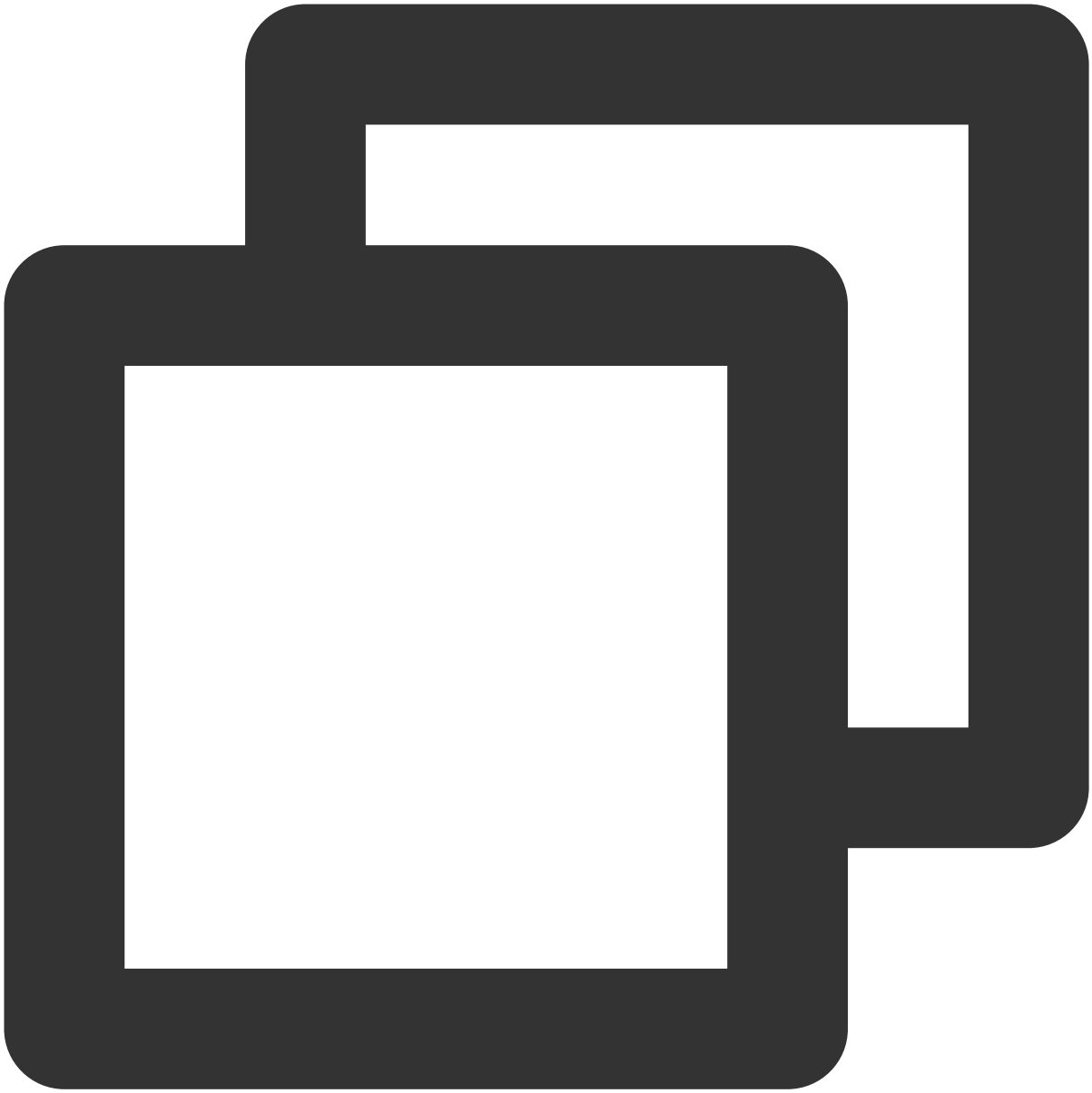
Field	Description
Algorithm	Signature algorithm, which is always `TC3-HMAC-SHA256` currently.

RequestTimestamp	Request timestamp, i.e., the value of the common parameter `X-TC-Timestamp` in request header. It is the UNIX timestamp of the current time in seconds, such as `1551113065` in this example.
CredentialScope	Scope of the credential in the format of `Date/service/tc3_request`, including date, requested service, and termination string (tc3_request). **`Date` indicates a U date, which should match the UTC date converted by the common parameter `TC-Timestamp`.** `service` is the service name, which should match the domain n of the service called. The calculation result in this example is `2019-02-25/cvm/tc3_request`.
HashedCanonicalRequest	Hash value of the canonical request string concatenated in the steps above. The pseu code for calculation is `Lowercase(HexEncode(Hash.SHA256(CanonicalRequest)))`. The calcu result in this example is `5ffe6a04c0664d6b969fab9a13bdab201d63ee709638e2749d62a09ca18d`

Note:

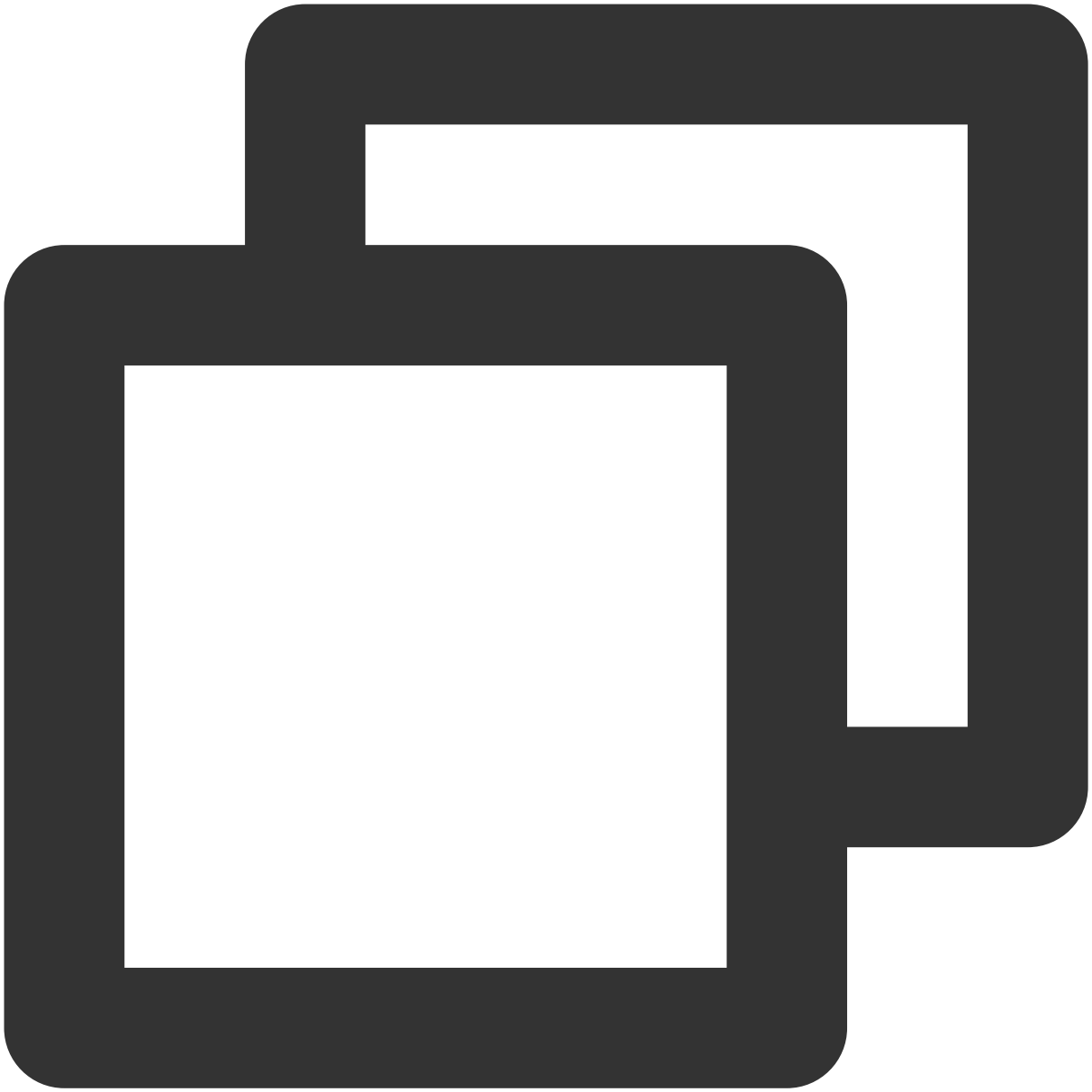
1. `Date` must be calculated from the timestamp `X-TC-Timestamp` and the time zone is UTC+0. If you add the local time zone information (such as UTC+8) in the system, calls can succeed both day and night but will definitely fail at 00:00. For example, if the timestamp is 1551113065 and the time in UTC+8 is 2019-02-26 00:44:25, the UTC+0 date in the calculated `Date` value should be 2019-02-25 instead of 2019-02-26.
2. `Timestamp` must be the same as your current system time, and your system time must be in sync with the UTC time. If the difference between the timestamp and your current system time is greater than five minutes, the request will fail. If your system time is out of sync with the UTC time for a prolonged period, the request will fail, and a signature expiration error will be returned.

According to the rules above, the string to sign obtained in the example is as follows:



```
TC3-HMAC-SHA256
1551113065
2019-02-25/cvm/tc3_request
5ffe6a04c0664d6b969fab9a13bdab201d63ee709638e2749d62a09ca18d7031
```

3. Calculate the signature (pseudocode)



```
$secretDate = hash_hmac("SHA256", $date, "TC3".$secretKey, true);
$secretService = hash_hmac("SHA256", $service, $secretDate, true);
$secretSigning = hash_hmac("SHA256", "tc3_request", $secretService, true);
$signature = hash_hmac("SHA256", $stringToSign, $secretSigning);
echo $signature.PHP_EOL;
```

The derived key `SecretDate` , `SecretService` , and `SecretSigning` are binary data and may contain non-printable characters. Intermediate results are not displayed here.

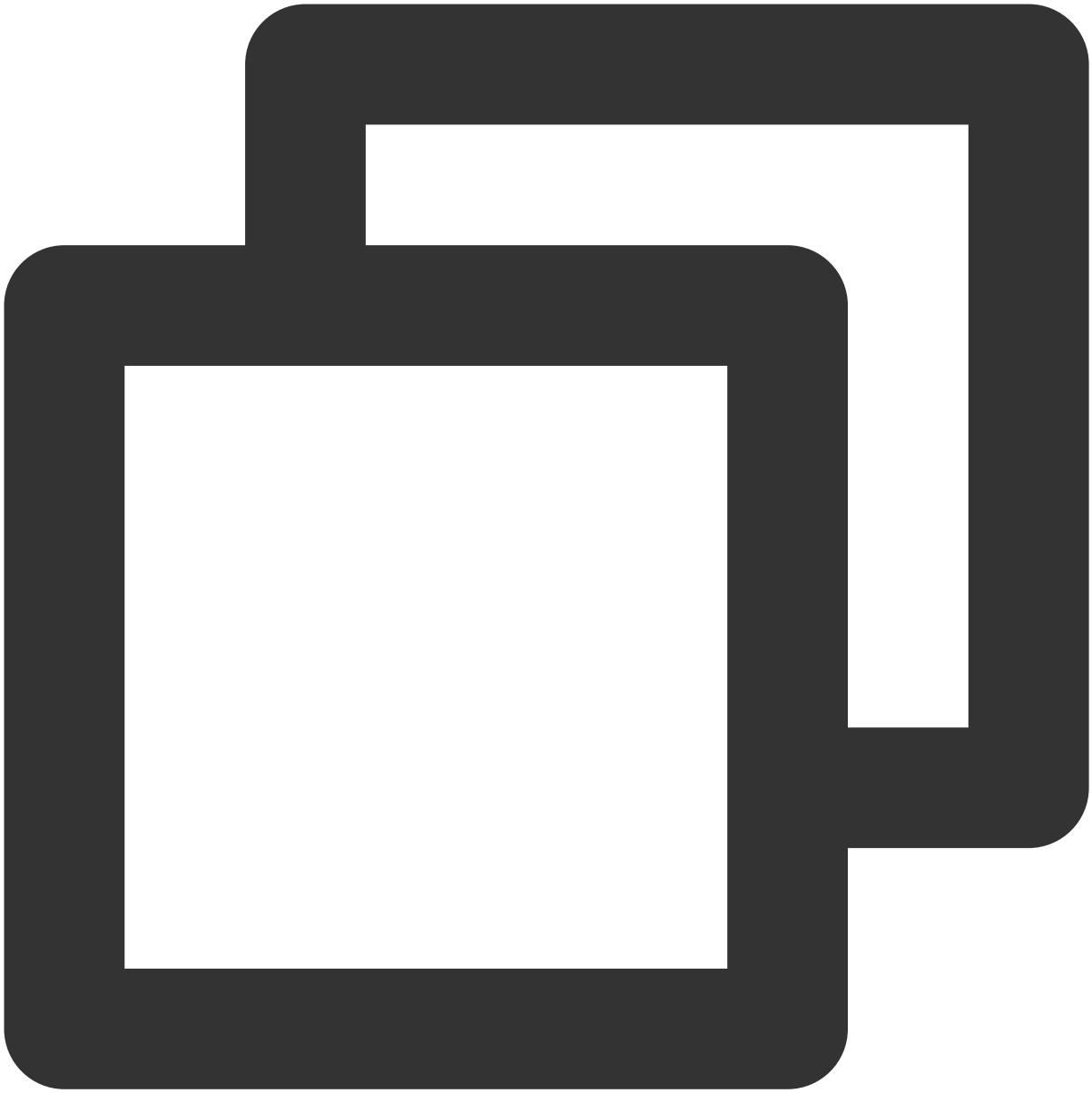
Field	Description

secretKey	Original `SecretKey` , i.e., `Gu5t9xGAR**********EXAMPLE` .
date	Value of the `Date` field in `Credential` , such as `2019-02-25` in this example.
service	Value of the `Service` field in `Credential` , such as `cvm` in this example.

The calculation result in this example is `72e494ea8****************************************a96525168` .

4. Concatenate the Authorization string

Concatenate the `Authorization` string in the following format:



```
$authorization = $algorithm
    ." Credential=".$secretId."/".$credentialScope
    ., SignedHeaders=content-type;host, Signature=".$signature;
echo $authorization.PHP_EOL;
```

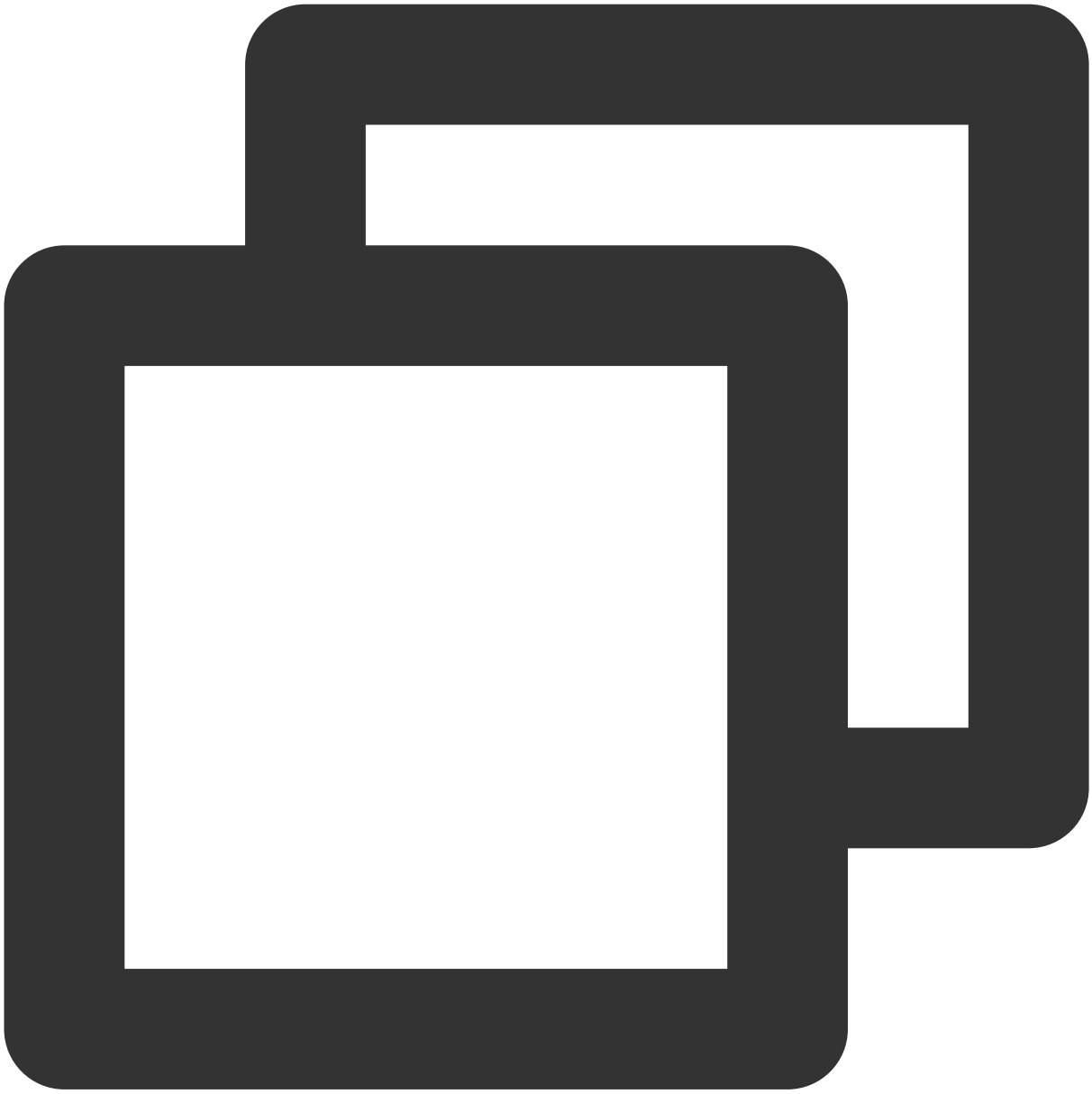
Field	Description
algorithm	Signature algorithm, which is always `TC3-HMAC-SHA256` .
secretId	`SecretId` in the key pair, i.e., `AKIDz8krbsJ5**********mLPx3EXAMPLE` .

credentialScope	Credential scope (see above). The calculation result in this example is `2019-02-25/cvm/tc3_request` .
SignedHeaders	Header information for signature calculation (see above), such as `content-type;host` in this example.
signature	Signature value. The calculation result in this example is `72e494ea8**********a96525168` .

According to the rules above, the values obtained in this example are:

```
TC3-HMAC-SHA256 Credential=AKIDz8krbsJ5**********mLPx3EXAMPLE/2019-02-25/cvm/tc3_re
```

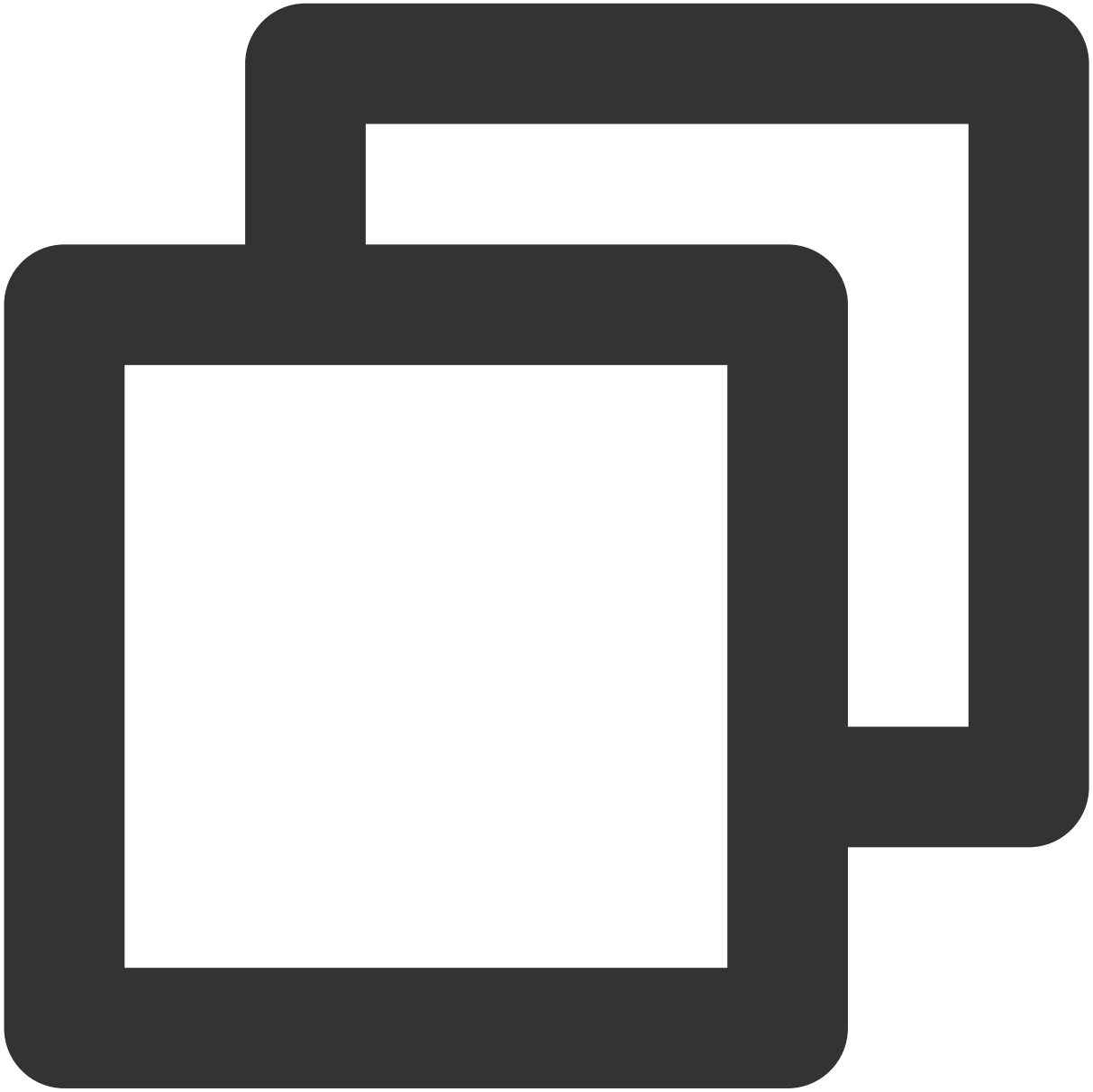
The complete call information is as follows:



```
POST https://cvm.tencentcloudapi.com/
Authorization: TC3-HMAC-SHA256 Credential=AKIDz8krbsJ5**********mLPx3EXAMPLE/2019-0
Content-Type: application/json; charset=utf-8
Host: cvm.tencentcloudapi.com
X-TC-Action: DescribeInstances
X-TC-Version: 2017-03-12
X-TC-Timestamp: 1551113065
```

```
X-TC-Region: ap-guangzhou

{"Limit": 1, "Filters": [{"Values": ["\\u672a\\u547d\\u540d"], "Name": "instance-na
```

5. Sample API 3.0 signature v3



```
<?php
$secretId = "AKIDz8krbsJ5**********mLPx3EXAMPLE";
$secretKey = "Gu5t9xGAR***********EXAMPLE";
$host = "cvm.tencentcloudapi.com";
$service = "cvm";
```

```
$version = "2017-03-12";
$action = "DescribeInstances";
$region = "ap-guangzhou";
// $timestamp = time();
$timestamp = 1551113065;
$algorithm = "TC3-HMAC-SHA256";

// step 1: build canonical request string
$httpRequestMethod = "POST";
$canonicalUri = "/";
$canonicalQueryString = "";
$canonicalHeaders = "content-type:application/json; charset=utf-8\\n"."host:".$host
$signedHeaders = "content-type;host";
$payload = '{"Limit": 1, "Filters": [{"Values": ["\\u672a\\u547d\\u540d"], "Name":
$hashedRequestPayload = hash("SHA256", $payload);
$canonicalRequest = $httpRequestMethod."\\n"
    .$canonicalUri."\\n"
    .$canonicalQueryString."\\n"
    .$canonicalHeaders."\\n"
    .$signedHeaders."\\n"
    .$hashedRequestPayload;
echo $canonicalRequest.PHP_EOL;

// step 2: build string to sign
$date = gmdate("Y-m-d", $timestamp);
$credentialScope = $date."/".$service."/tc3_request";
$hashedCanonicalRequest = hash("SHA256", $canonicalRequest);
$stringToSign = $algorithm."\\n"
    .$timestamp."\\n"
    .$credentialScope."\\n"
    .$hashedCanonicalRequest;
echo $stringToSign.PHP_EOL;

// step 3: sign string
$secretDate = hash_hmac("SHA256", $date, "TC3".$secretKey, true);
$secretService = hash_hmac("SHA256", $service, $secretDate, true);
$secretSigning = hash_hmac("SHA256", "tc3_request", $secretService, true);
$signature = hash_hmac("SHA256", $stringToSign, $secretSigning);
echo $signature.PHP_EOL;

// step 4: build authorization
$authorization = $algorithm
    ." Credential=".$secretId."/".$credentialScope
    ., SignedHeaders=content-type;host, Signature=".$signature;
echo $authorization.PHP_EOL;

$curl = "curl -X POST https://".$host
```

```
    .' -H "Authorization: '.$authorization.'"'
    .' -H "Content-Type: application/json; charset=utf-8"'
    .' -H "Host: '.$host.'"'
    .' -H "X-TC-Action: '.$action.'"'
    .' -H "X-TC-Timestamp: '.$timestamp.'"'
    .' -H "X-TC-Version: '.$version.'"'
    .' -H "X-TC-Region: '.$region.'"'
    ." -d '".$payload."'";
echo $curl.PHP_EOL;
```

2. Get an API 3.0 signature v1

The signature algorithm v1 is simple and easy to use, but its functionality and security are not as good as the signature algorithm v3 which is therefore recommended.

Note:

If you are using the signature algorithm for the first time, we recommend you use the "signature string generation" feature in API Explorer and select "API 3.0 signature v1" as the signature version, which can generate a signature for demonstration and verification and provides signing examples for certain programming languages. Plus, it can also generate SDK code directly. Seven common open-source programming language SDKs are available for TencentCloud API, including Python, Java, PHP, Go, Node.js, .NET, and C++.

For example, if you call the `DescribeInstances` API to query CVM instances, the request parameters may be as follows:

Parameter Name	Description	Value
Action	Method	DescribeInstances
SecretId	Key ID	`AKIDz8krbsJ5*********mLPx3EXAMPLE`
Timestamp	Current timestamp	1465185768
Nonce	Random positive integer	11886
Region	Instance region	ap-guangzhou
InstanceIds.0	ID of the instance to be queried	ins-09dx96dg
Offset	Offset	0
Limit	Allowed maximum number of output entries	20
Version	API version number	2017-03-12

1. Sort parameters

Sort all the request parameters in an ascending lexicographical order (ASCII code) by their names.

Note:

1. The parameters are sorted only by name but not by value.
2. The parameters are sorted based on ASCII code but not in an alphabetical order or by value. For example, `InstanceIds.2` should be arranged behind `InstanceIds.12`. You can complete sorting by using a sorting function in a programming language, such as the `ksort` function in PHP.

The parameters in the example are sorted as follows:

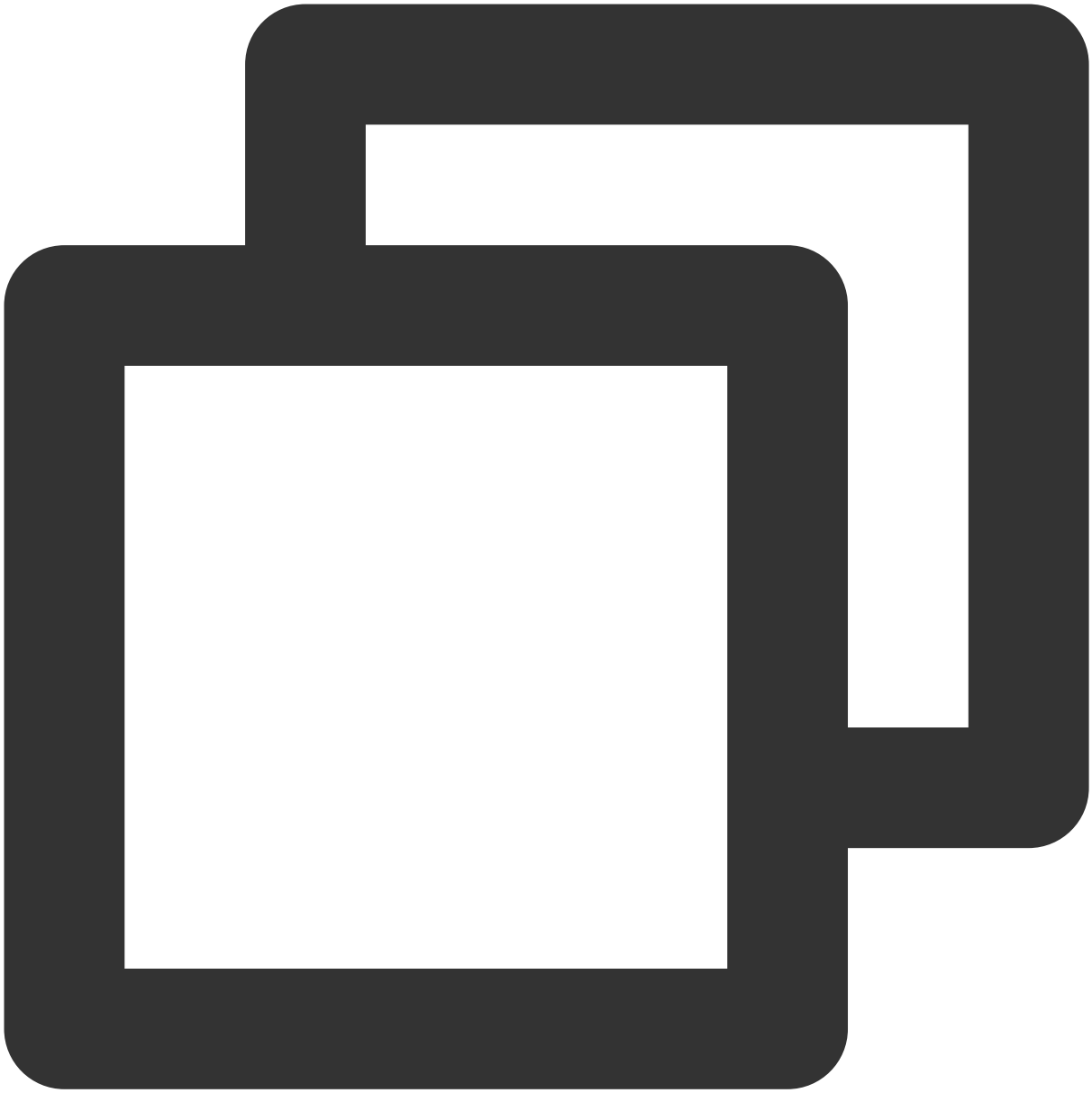
```
{
    'Action' : 'DescribeInstances',
    'InstanceIds.0' : 'ins-09dx96dg',
    'Limit' : 20,
    'Nonce' : 11886,
    'Offset' : 0,
    'Region' : 'ap-guangzhou',
    'SecretId' : 'AKIDz8krbsJ5**********mLPx3EXAMPLE',
    'Timestamp' : 1465185768,
    'Version': '2017-03-12',
}
```

Any other programming languages can be used to sort these parameters as long as the same result is produced.

2. Concatenate the canonical request string

This step generates a request string. Format the request parameters sorted in the previous step into the form of `parameter=value`. For example, for the `Action` parameter, its parameter is `Action` and its value is `DescribeInstances`; therefore, the parameter will be formatted into `Action=DescribeInstances`. **Note: the `value` is the original value instead of the URL-encoded value.**

Then, concatenate the formatted parameters with `&`. The generated request string will be as follows:



```
Action=DescribeInstances&InstanceIds.0=ins-09dx96dg&Limit=20&Nonce=11886&Offset=0&R
```

3. Concatenate the string to sign

This step generates the original signature string. The original signature string consists of the following parameters:

1. Request method: POST and GET methods are supported. GET is used here for the request. Please note that the method name should be in all capital letters.

2. Request server: the domain name of the request for querying instances (DescribeInstances) is `cvm.tencentcloudapi.com` . The actual request domain name varies by the module to which the API belongs.

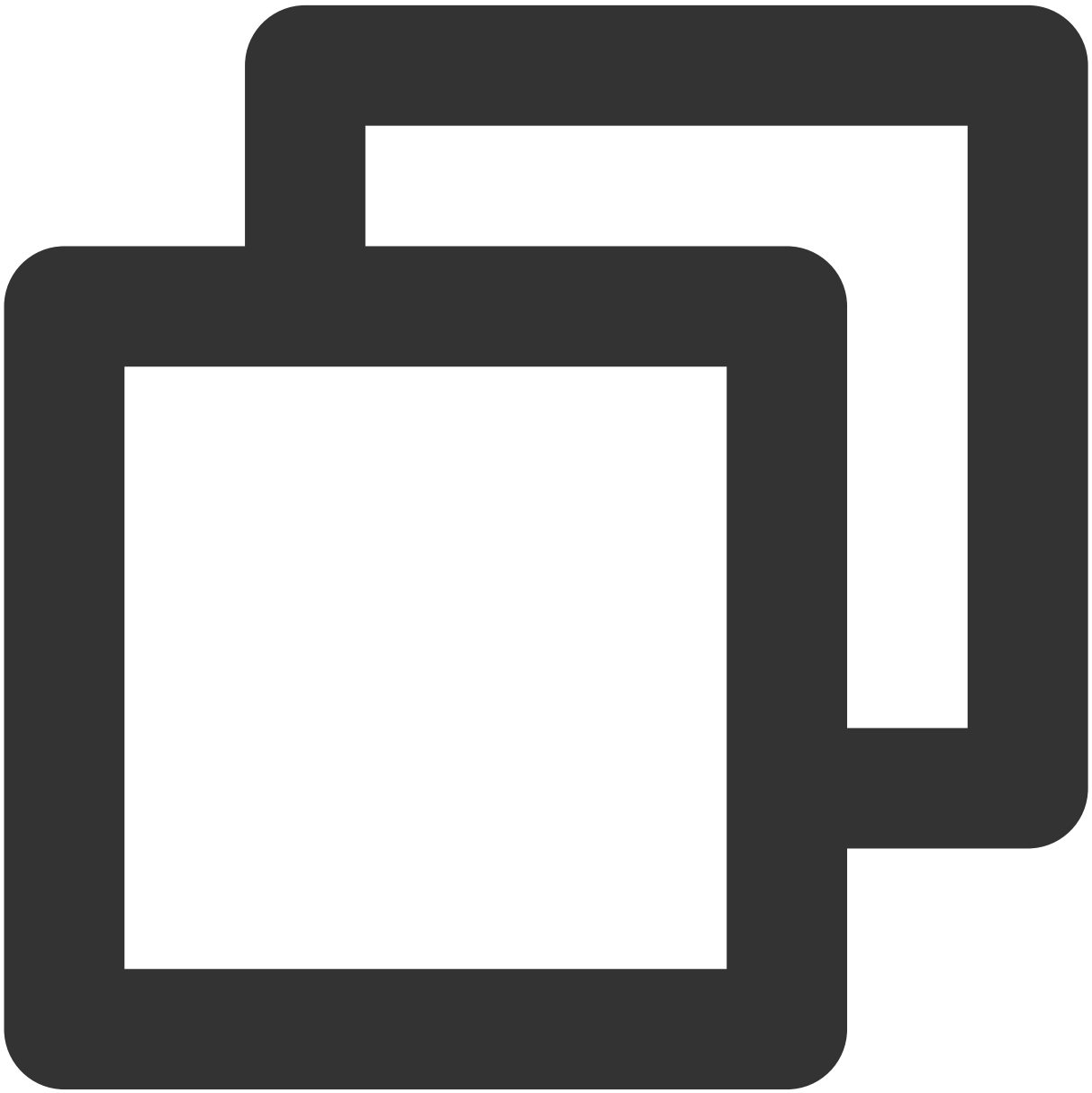
For more information, please see the specific API document.

3. Request path: the request path in the current version of TencentCloud API is fixed to `/` .

4. Request string: the request string generated in the previous step.

The rule for concatenating the original string of the signature is `request method + request server + request path + ? + request string` .

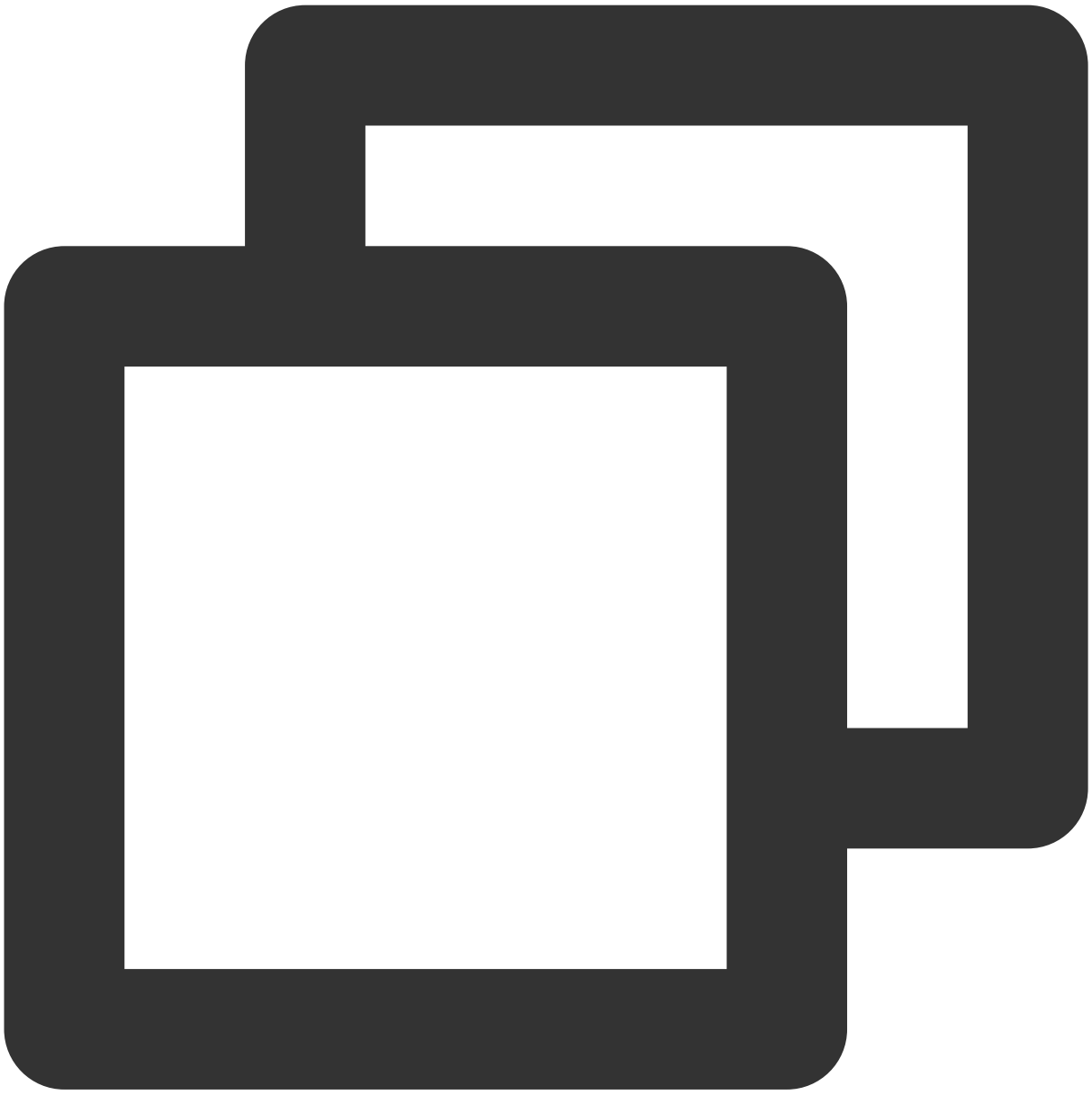
The concatenation result in the example is as follows:



```
GETcvm.tencentcloudapi.com/?Action=DescribeInstances&InstanceIds.0=ins-09dx96dg&Lim
```

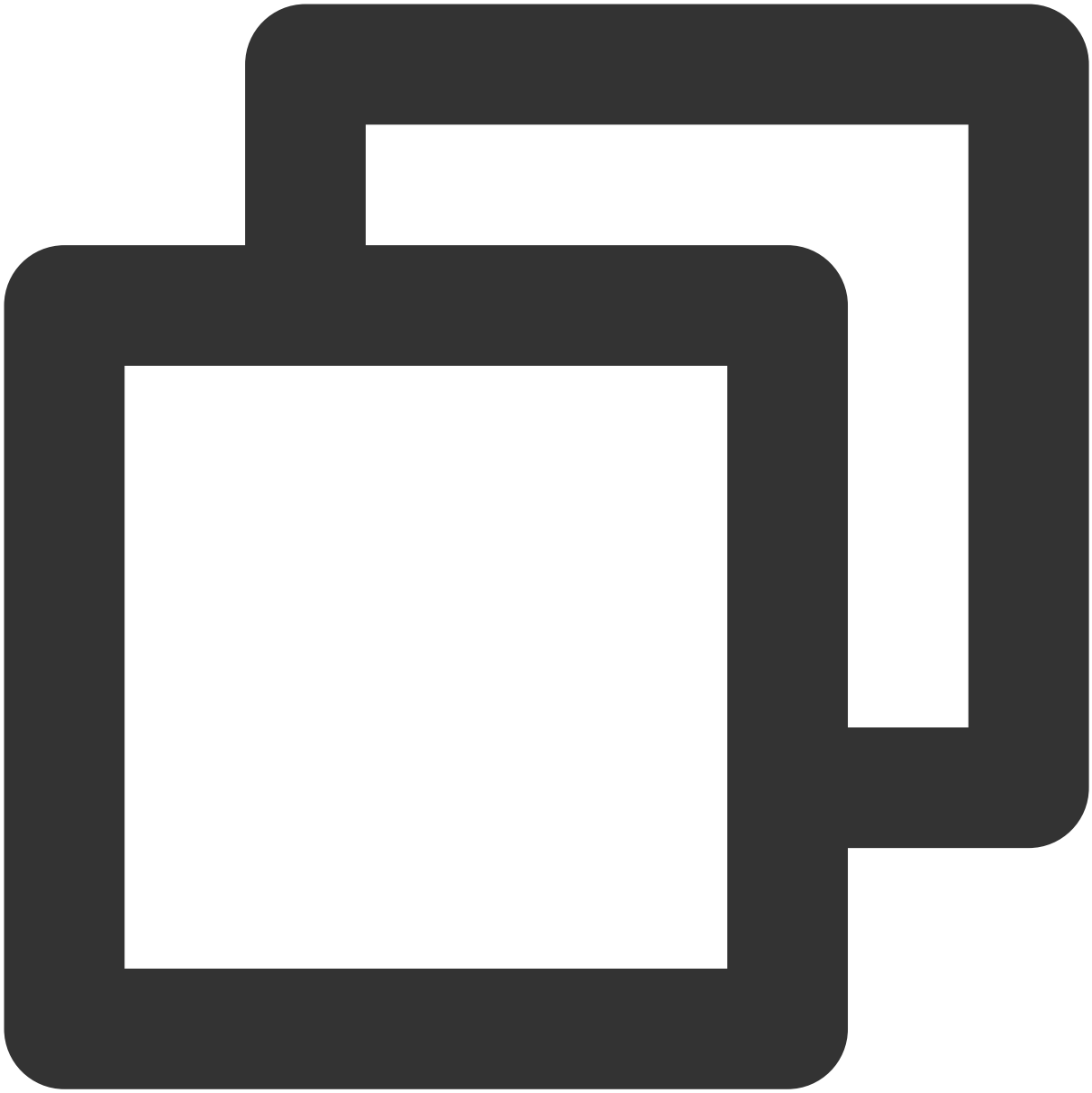
4. Calculate the signature (pseudocode)

This step generates a signature string. Use the HMAC-SHA1 algorithm to sign the **original signature string** obtained in the previous step, and then Base64-encode the generated signature to get the final signature.



```
$secretKey = 'Gu5t9xGAR***********EXAMPLE';
$srcStr = 'GETcvm.tencentcloudapi.com/?Action=DescribeInstances&InstanceIds.0=ins-0
$signStr = base64_encode(hash_hmac('sha1', $srcStr, $secretKey, true));
echo $signStr;
```

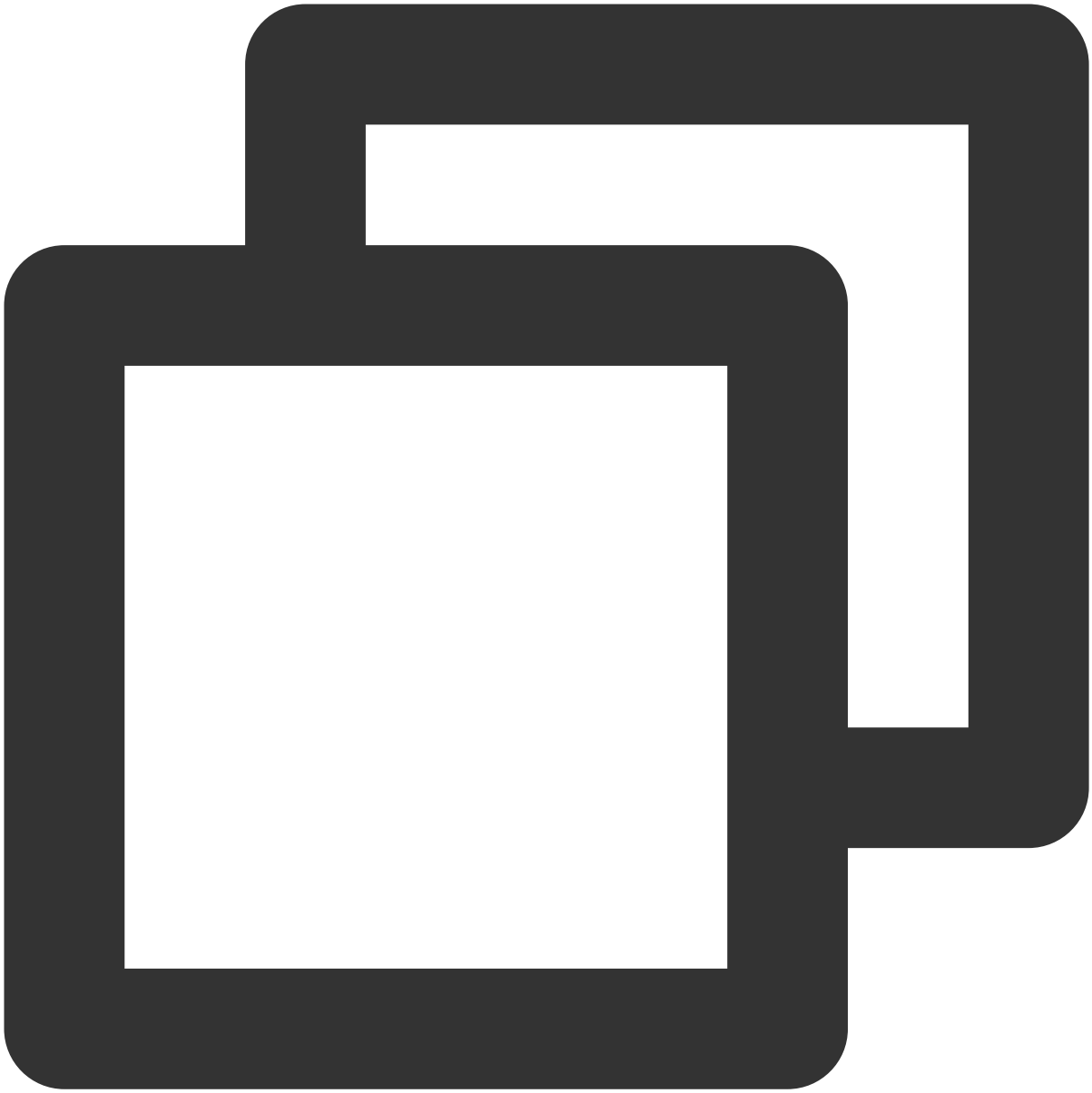
The obtained signature string is as follows:



```
EliP9YW3pW28FpsEdkXt/+WcGeI=
```

5. Get the call information and send a request

```
# The API will be called actually, and fees will be incurred if it is a consumption
resp = requests.get("https://" + endpoint, params=data)
# Print the request string of the sent request
print(resp.url)
```



```
The obtained request string is as follows:
https://cvm.tencentcloudapi.com/?Action=DescribeInstances&InstanceIds.0=ins-09dx96d
```

Field	Description
endpoint	Service address, such as `cvm.tencentcloudapi.com` .
data	API parameter of the sample API 3.0 signature v1. **Note: you should add the calculated signature in the format of key-value pair to `data` .**

Note:

The key in the example is not real, and the timestamp is not the current system time. If you open this URL in the browser or call it by using commands such as `curl` , an authentication error `The signature expired` will be returned. To obtain a URL that works, you need to replace the `SecretId` and `SecretKey` in this example with your own credentials and use the current system time as the `Timestamp` .

To further explain the signing process, PHP is used as examples below to implement the process as described above. The request domain name, API, and parameter values in the above example are used here. The code below is for demonstration only. Please use the SDK for actual development.

6. Encode a signature string

The generated signature string cannot be directly used as a request parameter and needs to be URL-encoded. For example, if the signature string generated in the previous step is `Eli****************cGeI=` , the final value of the `Signature` request parameter will be `EliP**********************eI%3D` , which will be used to generate the final request URL.

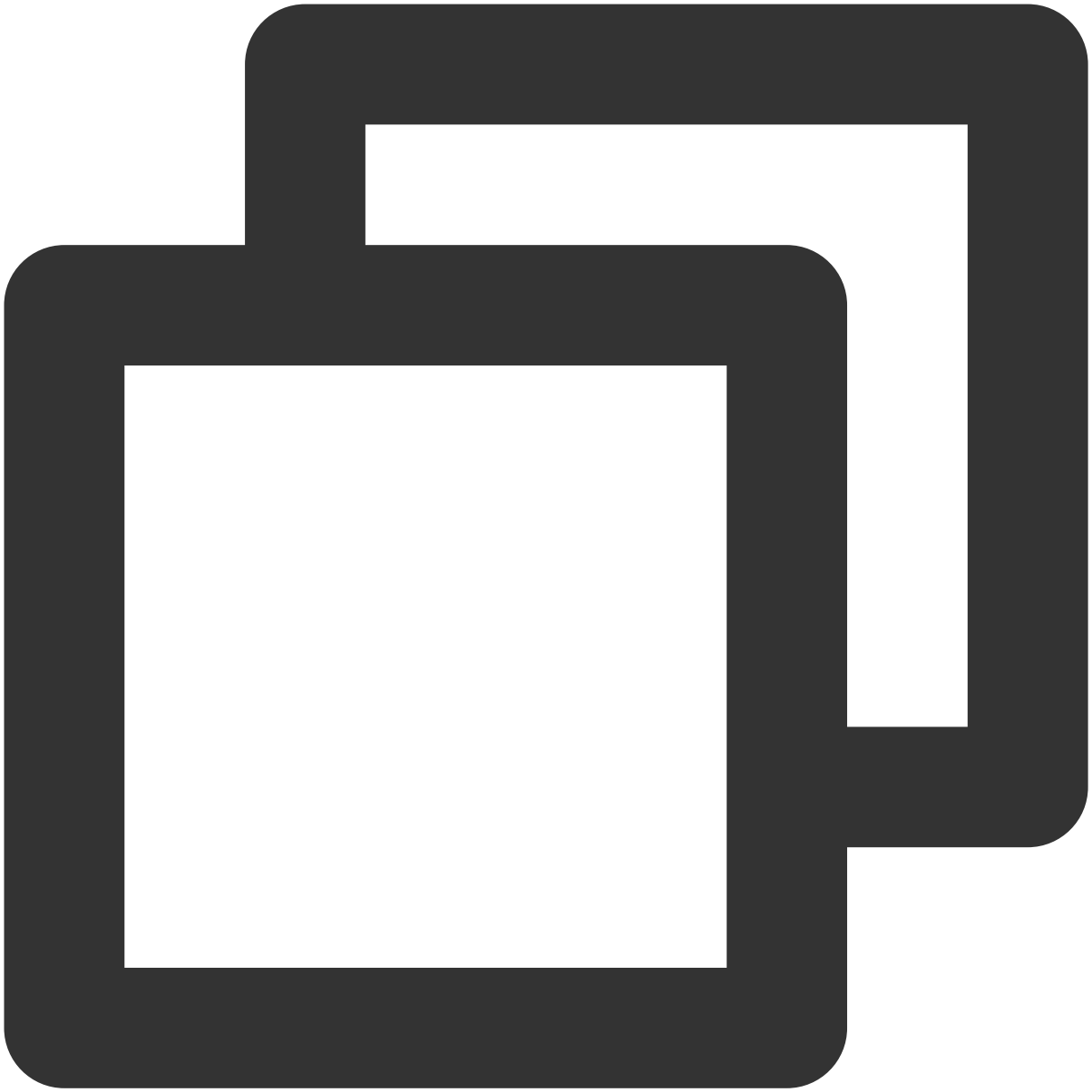
Note:

If you use the GET request method or use the POST request method with `Content-Type` of `application/x-www-form-urlencoded` , all the request parameter values must be URL-encoded (except the parameter key and the equal symbol (=)) before the request is sent. Non-ASCII characters must be encoded with UTF-8 before URL-encoding.

The network libraries of some programming languages automatically URL-encode all parameters. In this case, the signature string does not need to be URL-encoded again; otherwise, two rounds of URL-encoding will cause the signature to fail.

Other parameter values also need to be encoded with RFC 3986. Use %XY in percent-encoding for special characters such as Chinese characters, where "X" and "Y" are hexadecimal characters (0–9 and uppercase A–F). Using lowercase characters will cause an error.

7. Sample API 3.0 signature v1



```
<?php
$secretId = "AKIDz8krbsJ5**********mLPx3EXAMPLE";
$secretKey = "Gu5t9xGAR***********EXAMPLE";
$param["Nonce"] = 11886;//rand();
$param["Timestamp"] = 1465185768;//time();
$param["Region"] = "ap-guangzhou";
$param["SecretId"] = $secretId;
$param["Version"] = "2017-03-12";
$param["Action"] = "DescribeInstances";
$param["InstanceIds.0"] = "ins-09dx96dg";
$param["Limit"] = 20;
```

```
$param["Offset"] = 0;

ksort($param);

$signStr = "GETcvm.tencentcloudapi.com/?";
foreach ( $param as $key => $value ) {
    $signStr = $signStr . $key . "=" . $value . "&";
}
$signStr = substr($signStr, 0, -1);

$signature = base64_encode(hash_hmac("sha1", $signStr, $secretKey, true));
echo $signature.PHP_EOL;
// need to install and enable curl extension in php.ini
// $param["Signature"] = $signature;
// $url = "https://cvm.tencentcloudapi.com/?".http_build_query($param);
// echo $url.PHP_EOL;
// $ch = curl_init();
// curl_setopt($ch, CURLOPT_URL, $url);
// $output = curl_exec($ch);
// curl_close($ch);
// echo json_decode($output);
```

API 2.0 Signature

This signature version has been disused. We recommend you use **API 3.0 signature** with better performance. If you still need to use it, please go to API Explorer > **Signature Generation** and select **API 2.0 Signature** as the signature version.

Signature Failure

The following error codes may be returned for signature failure. Please resolve the errors accordingly.

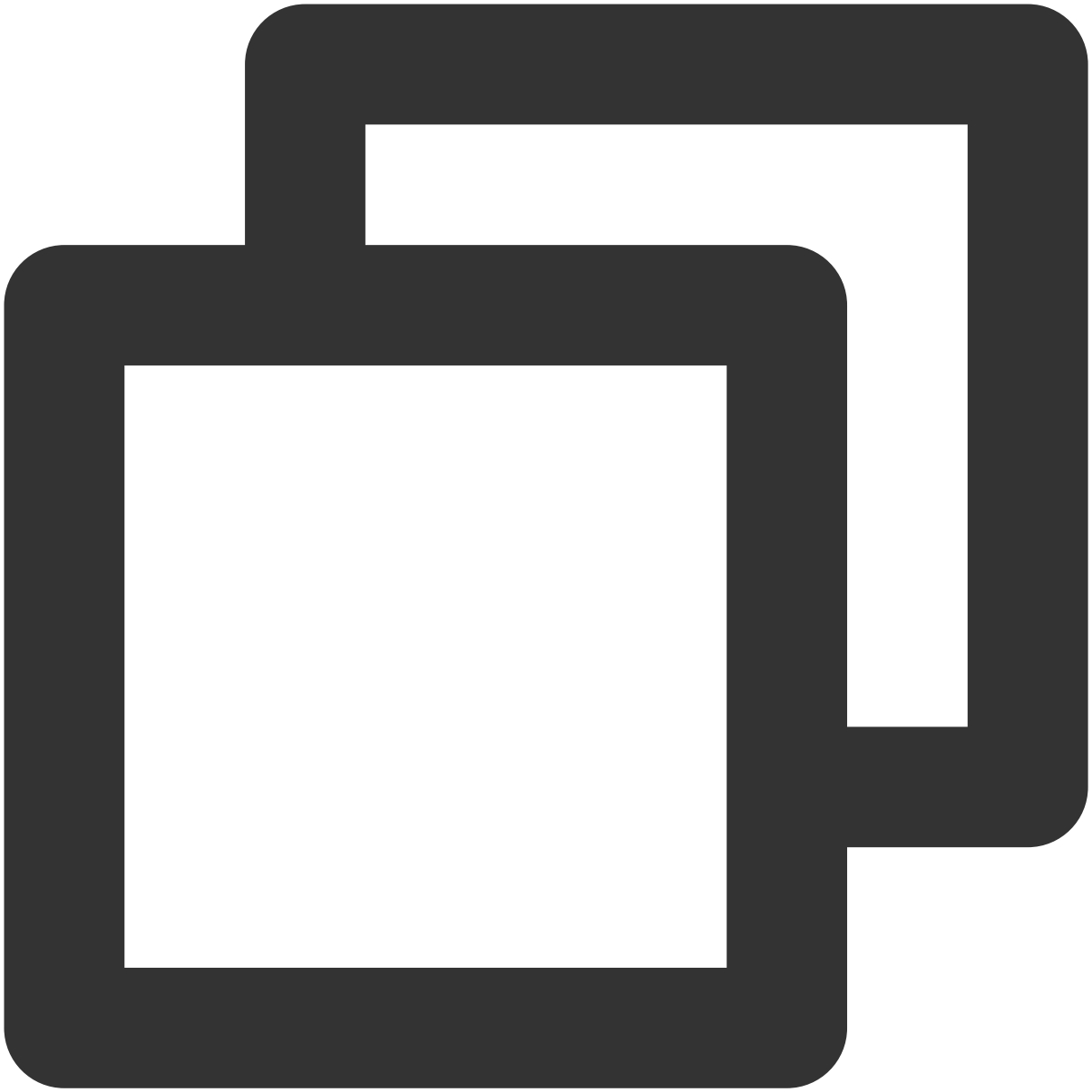
Error Code	Error Description
AuthFailure.SignatureExpire	The signature expired. The difference between the `Timestamp` and the server time cannot be greater than five minutes.
AuthFailure.SecretIdNotFound	The key does not exist. Log in to the console and check whether it is disabled or you copied fewer or more characters.
AuthFailure.SignatureFailure	Signature error. It is possible that the signature is calculated incorrectly, the signature does not match the content that is actually sent, or the `SecretKey` is incorrect.

AuthFailure.TokenFailure	Temporary credential token error.
AuthFailure.InvalidSecretId	Invalid key (not TencentCloud API key type).

Returned Result

Successful response

For example, when calling the CVM API `DescribeInstances` (version: 2017-03-12) to view the list of instances, if the request succeeds, you may see the following response:



```
{
    "Response": {
        "TotalCount": 0,
        "InstanceStatusSet": [],
        "RequestId": "b5b41468-520d-4192-b42f-595cc34b6c1c"
    }
}
```

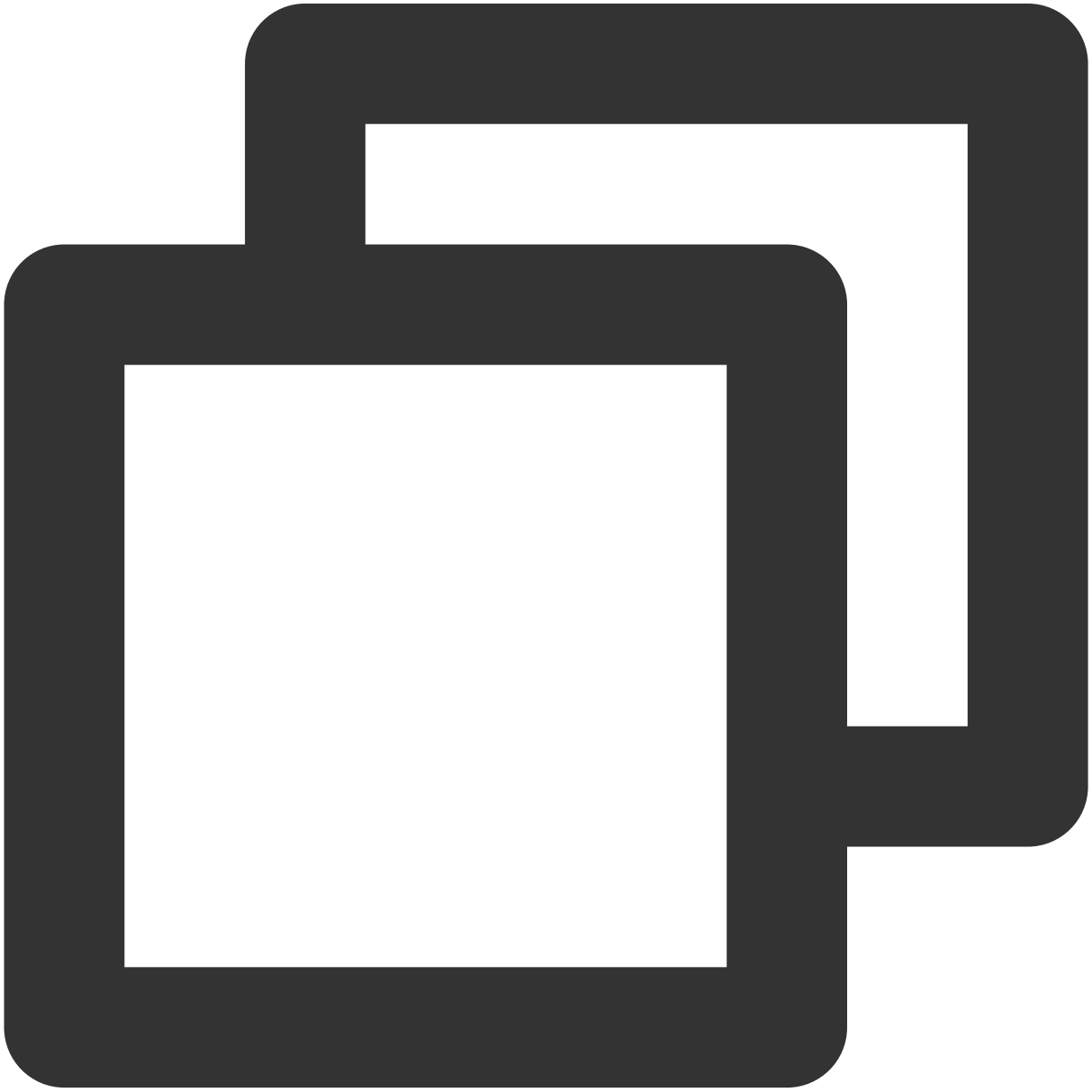
The API will return `Response` , which contains `RequestId` , as long as it processes the request, no matter whether the request is successful or not.

`RequestId` is the unique ID of an API request. It is required to troubleshoot issues.

Any fields other than the common fields are API-specific. For more information on such fields, please see the relevant API documentation. In this example, both `TotalCount` and `InstanceStatusSet` are specific to the `DescribeInstancesStatus` API. Since the user who initiated the request does not have a CVM instance yet, 0 is returned for `TotalCount` and `InstanceStatusSet` is empty.

Error response

If the call fails, you may see the following response:



```
{
```

```
    "Response": {
        "Error": {
            "Code": "AuthFailure.SignatureFailure",
            "Message": "The provided credentials could not be validated. Please che
        },
        "RequestId": "ed93f3cb-f35e-473f-b9f3-0d451b8b79c6"
    }
}
```

`Error` indicates that the request failed. A response for a failed request will always include the `Error`, `Code`, and `Message` fields.

`Code` indicates the specific error code, which is returned when an API request failed. You can use this code to locate the cause and solution of the error in the common or API-specific error code list.

`Message` explains the cause of the error. Note that the returned messages are subject to service updates. The information the messages provide may not be up-to-date and should not be the only source of reference.

`RequestId` is the unique ID of an API request. It is required to troubleshoot issues.

Common error codes

The `Error` field in a response indicates that the API call failed. The `Code` field in `Error` indicates the error code. The following table lists the common error codes that any services may return.

Error Code	Error Description
AuthFailure.InvalidSecretId	Invalid key (not TencentCloud API key type).
AuthFailure.MFAFailure	MFA failure.
AuthFailure.SecretIdNotFound	The key does not exist.
AuthFailure.SignatureExpire	The signature expired.
AuthFailure.SignatureFailure	Signature error.
AuthFailure.TokenFailure	Token error.
AuthFailure.UnauthorizedOperation	No CAM authorization.
DryRunOperation	DryRun Operation. It means that the request would have succeeded, but the DryRun parameter was used.
FailedOperation	The operation failed.
InternalError	Internal error.
InvalidAction	The API does not exist.

InvalidParameter	Incorrect parameter.
InvalidParameterValue	Invalid parameter value.
LimitExceeded	The quota limit is exceeded.
MissingParameter	A parameter is missing.
NoSuchVersion	The API version does not exist.
RequestLimitExceeded	The request rate limit is exceeded.
ResourceInUse	The resource is in use.
ResourceInsufficient	Insufficient resource.
ResourceNotFound	The resource does not exist.
ResourceUnavailable	The resource is unavailable.
UnauthorizedOperation	Unauthorized operation.
UnknownParameter	Unknown parameter error.
UnsupportedOperation	Unsupported operation.
UnsupportedProtocol	Unsupported HTTPS request method. Only GET and POST requests are supported.
UnsupportedRegion	Unsupported region.

API for Python

Last updated：2023-03-07 18:16:40

TencentCloud API has been upgraded to v3.0. This version is optimized for performance and deployed in all regions. It supports nearby access and access by region for significantly reduced access latency. In addition, it features more detailed API descriptions and error codes and API-level comments for SDKs, enabling you to use Tencent Cloud services more conveniently and quickly. This document describes how to call APIs for Python.
This version currently supports various Tencent Cloud services such as CVM, CBS, VPC, and TencentDB and will support more services in the future.

Request Structure

1. Service address (endpoint)

TencentCloud API supports access from either a nearby region (such as `cvm.tencentcloudapi.com` for CVM) or a specified region (such as `cvm.ap-guangzhou.tencentcloudapi.com` for CVM in the Guangzhou region). For values of the region parameter, please see the region list in the "Common Parameters" section below. To check whether a region is supported by a specific Tencent Cloud service, please see its "Request Structure" document.

Note:

For latency-sensitive businesses, we recommend you specify a domain name with a region.

2. Communications protocol

All TencentCloud APIs communicate over HTTPS, providing highly secure communications tunnels.

3. Request method

Supported HTTP request methods:
POST (recommended)
GET
`Content-Type` types supported by POST request:
application/json (recommended). The signature algorithm v3 (TC3-HMAC-SHA256) must be used.
application/x-www-form-urlencoded. The signature algorithm v1 (HmacSHA1 or HmacSHA256) must be used.
multipart/form-data (only supported by certain APIs). The signature algorithm v3 (TC3-HMAC-SHA256) must be used.
The size of a GET request packet cannot exceed 32 KB. The size of a POST request cannot exceed 1 MB for the signature algorithm v1 (HmacSHA1 or HmacSHA256) or 10 MB for the signature algorithm v3 (TC3-HMAC-SHA256).

4. Character encoding

UTF-8 encoding is always used.

Common Parameters

Note:
The common parameters are used to identity the user and API signature. They should be carried by each request to initiate properly.

Signature algorithm v3

The signature algorithm v3 (sometimes referred to as "TC3-HMAC-SHA256") is more secure than the signature algorithm v1 (referred to as signature algorithm in certain documents), supports larger request packets and POST JSON format, and has a higher performance. We recommend you use it to calculate signatures. For more information on how to use it, please see below.

Parameter Name	Type	Required	Description
X-TC-Action	String	Yes	Name of the API for the desired operation. For the specific value, please see the description of common parameter `Action` in the input parameters in the related API document. For example, the API for querying CVM instance list is `DescribeInstances` .
X-TC-Region	String	-	Region parameter, which is used to identify the region where the data you want to manipulate resides. For values supported for an API, please see the description of common parameter `Region` in the input parameters in related API documentation. **Note: this parameter is not required for some APIs (which will be indicated in related API documentation) and will not take effect even if it is passed.**
X-TC-Timestamp	Integer	Yes	The current UNIX timestamp that records the time when the API request was initiated, such as 1529223702. **Note: if the difference between the UNIX timestamp and the server time is greater than 5 minutes, a signature expiration error may occur.**
X-TC-Version	String	Yes	Version of the API for the desired operation, such as 2017-03-12 for CVM. For the specific value, please see the description of common parameter `Version` in the input parameters in related API documentation.
Authorization	String	Yes	HTTP authentication request header, such as TC3-HMAC-SHA256 Credential=AKIDEXAMPLE/Date/service/tc3_request,

			SignedHeaders=content-type;host, Signature=72e494ea8**********a96525168 Here, TC3-HMAC-SHA256: signature algorithm, currently fixed as this value. Credential: signature credential. `AKIDEXAMPLE` indicates the `SecretId`. `Date` indicates a UTC date which must match the value of `X-TC-Timestamp` (a common parameter) in UTC format. `service` indicates the name of the service and is generally a domain name prefix; for example, the domain name `cvm.tencentcloudapi.com` means the CVM service, and the value for this service is `cvm`. SignedHeaders: the headers that contain the authentication information. `content-type` and `host` are required. Signature: signature digest. For the calculation process, please see below.
X-TC-Token	String	No	Token used for temporary credentials. It must be used with a temporary key. You can get the temporary key and token by calling a CAM API. No token is required for a long-term key.

Signature algorithm v1

When the signature algorithm v1 (sometimes referred to as "HmacSHA256" or "HmacSHA1") is used, the common parameters should be uniformly placed in the request string.

Parameter Name	Type	Required	Description
Action	String	Yes	Name of the API for the desired operation. For the specific value, please see the description of common parameter `Action` in the input parameters in the related API document. For example, the API for querying CVM instance list is `DescribeInstances`.
Region	String	-	Region parameter, which is used to identify the region where the data you want to manipulate resides. For values supported for an API, please see the description of common parameter `Region` in the input parameters in related API documentation. **Note: this parameter is not required for some APIs (which will be indicated in related API documentation) and will not take effect even if it is passed.**
Timestamp	Integer	Yes	The current UNIX timestamp that records the time when the API request was initiated, such as 1529223702. If the

			difference between the UNIX timestamp and the current time is too large, a signature expiration error may occur.
Nonce	Integer	Yes	A random positive integer used in conjunction with `Timestamp` to prevent replay attacks.
SecretId	String	Yes	The identifying `SecretId` obtained on the TencentCloud API Key page. A `SecretId` corresponds to a unique `SecretKey` which is used to generate the request signature (`Signature`).
Signature	String	Yes	Request signature, which is used to verify the validity of the request. It is generated based on input parameters. For more information on how to calculate the signature, please see below.
Version	String	Yes	Version of the API for the desired operation, such as 2017-03-12 for CVM. For the specific value, please see the description of common parameter `Version` in the input parameters in related API documentation.
SignatureMethod	String	No	Signature algorithm. Currently, only HmacSHA256 and HmacSHA1 are supported. The HmacSHA256 algorithm is used to verify the signature only when this parameter is specified as HmacSHA256. In other cases, the signature is verified with HmacSHA1.
Token	String	No	Token used for temporary credentials. It must be used with a temporary key. You can get the temporary key and token by calling a CAM API. No token is required for a long-term key.

Region list

As the supported regions vary by service, please refer to the region list in each service's product documentation for specific details.
For example, you can see the region list of CVM.

API Call Method for Python

TencentCloud API authenticates every request, that is, the request must be signed with the security credentials in the designated steps. Each request must contain the signature information in the common request parameters and be sent in the specified way and format.

Suppose your `SecretId` and `SecretKey` are `AKIDz8krbsJ5*********mLPx3EXAMPLE` and `Gu5t9xGAR**********EXAMPLE`, respectively. If you want to view the status of an unnamed instance in the Guangzhou region and have only one data entry returned, the request may be:

```
curl -X POST https://cvm.tencentcloudapi.com \\
-H "Authorization: TC3-HMAC-SHA256 Credential=AKIDz8krbsJ5*********mLPx3EXAMPLE/20
-H "Content-Type: application/json; charset=utf-8" \\
-H "Host: cvm.tencentcloudapi.com" \\
-H "X-TC-Action: DescribeInstances" \\
-H "X-TC-Timestamp: 1551113065" \\
-H "X-TC-Version: 2017-03-12" \\
```

```
-H "X-TC-Region: ap-guangzhou" \\
-d '{"Limit": 1, "Filters": [{"Values": ["\\u672a\\u547d\\u540d"], "Name": "instanc
```

Step 1. Apply for security credentials

In this document, the security credential used is a key pair, which consists of a `SecretId` and a `SecretKey`.

Each user can have up to two key pairs.

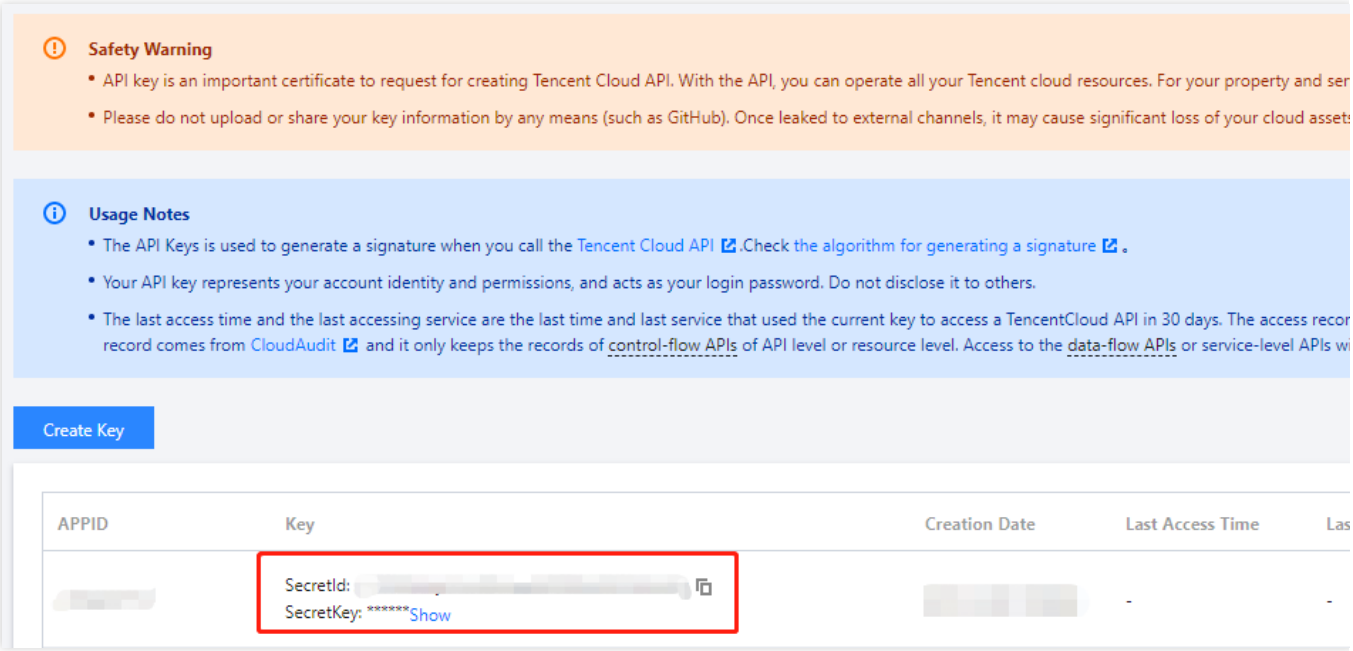
SecretId: identifies the user that calls an API, which is similar to a username.

SecretKey: authenticates the user that calls the API, which is similar to a password.

Note:

You must keep your security credentials private and avoid disclosure; otherwise, your assets may be compromised. If they are disclosed, please disable them as soon as possible.

Go to the API key management page to get API keys as shown below:



Step 2

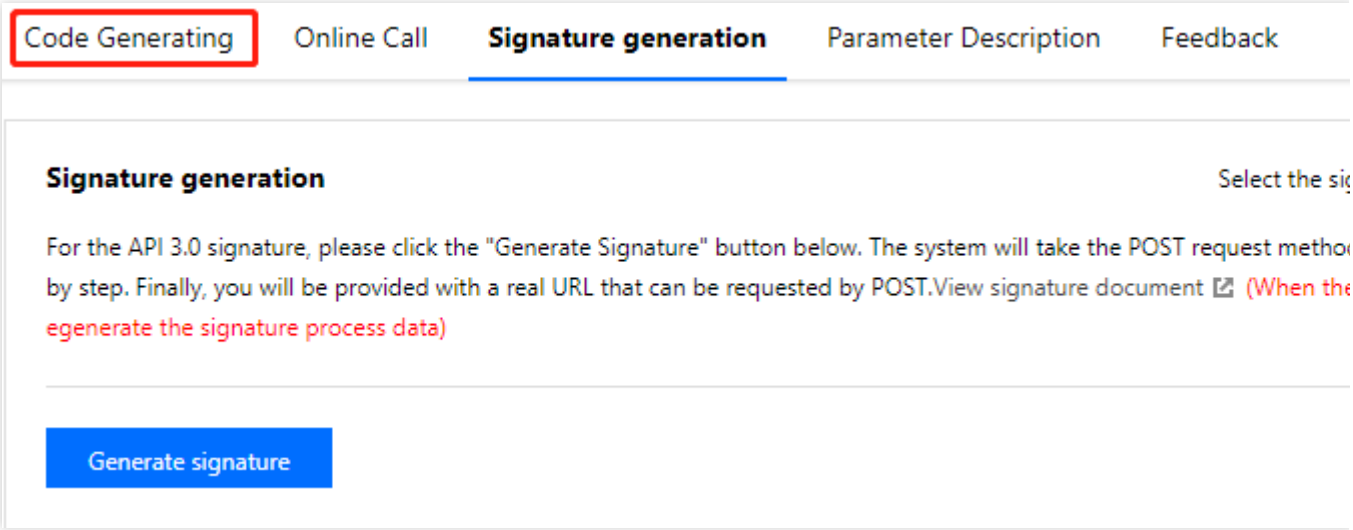
1. Get an API 3.0 signature v3

The signature algorithm v3 (TC3-HMAC-SHA256) is compatible with the previous signature algorithm v1 and more secure, supports larger request packets and POST JSON format, and has a higher performance. We recommend you use it to calculate signatures as shown below:

Note:

If you are using the signature algorithm for the first time, we recommend you use the "signature string generation" feature in API Explorer and select "API 3.0 signature v3" as the signature version, which can generate a signature for demonstration and verification. Plus, it can also generate SDK code directly. Seven common open-source

programming language SDKs are available for TencentCloud API, including Python, Java, PHP, Go, Node.js, .NET, and C++.

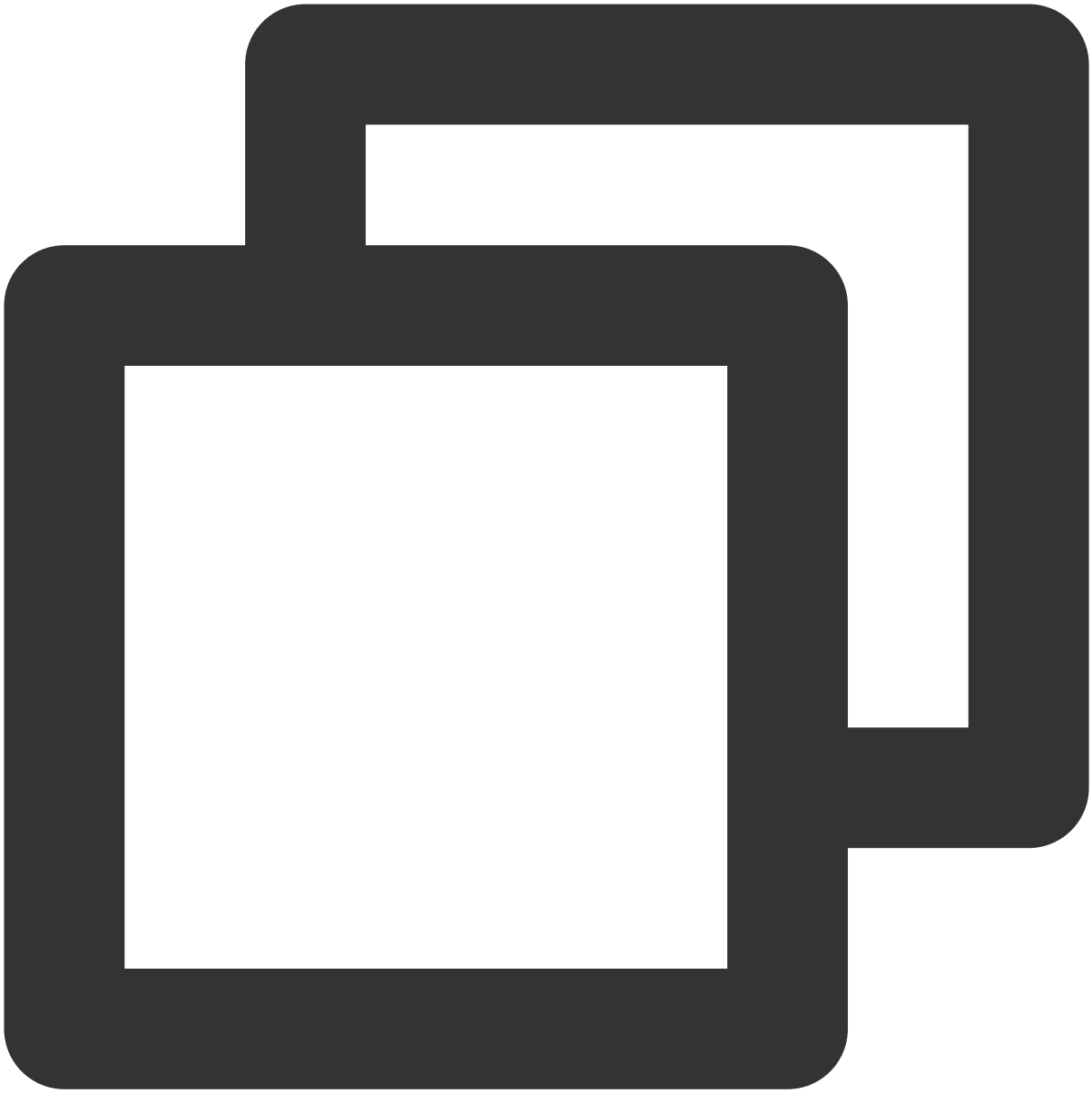


TencentCloud API supports both GET and POST requests. For the GET method, only the `Content-Type: application/x-www-form-urlencoded` protocol format is supported. For the POST method, `Content-Type: application/json` and `Content-Type: multipart/form-data` are supported. The JSON format is supported by all business APIs, while the multipart format is supported only by specific APIs (in this case, an API cannot be called in JSON format). For more information, please see the specific business API document. We recommend you use the POST method because the two methods generate the same results, but the GET method only supports request packets below 32 KB in size.

The following describes how to calculate a signature by calling the DescribeInstances API. This API is chosen because:

1. The CVM API is enabled by default, and this API is often used.
2. It is read-only and does not change the status of existing resources.
3. It covers many types of parameters so that it is easy to show how to use an array that contains data structures.

1. Concatenate the canonical request string

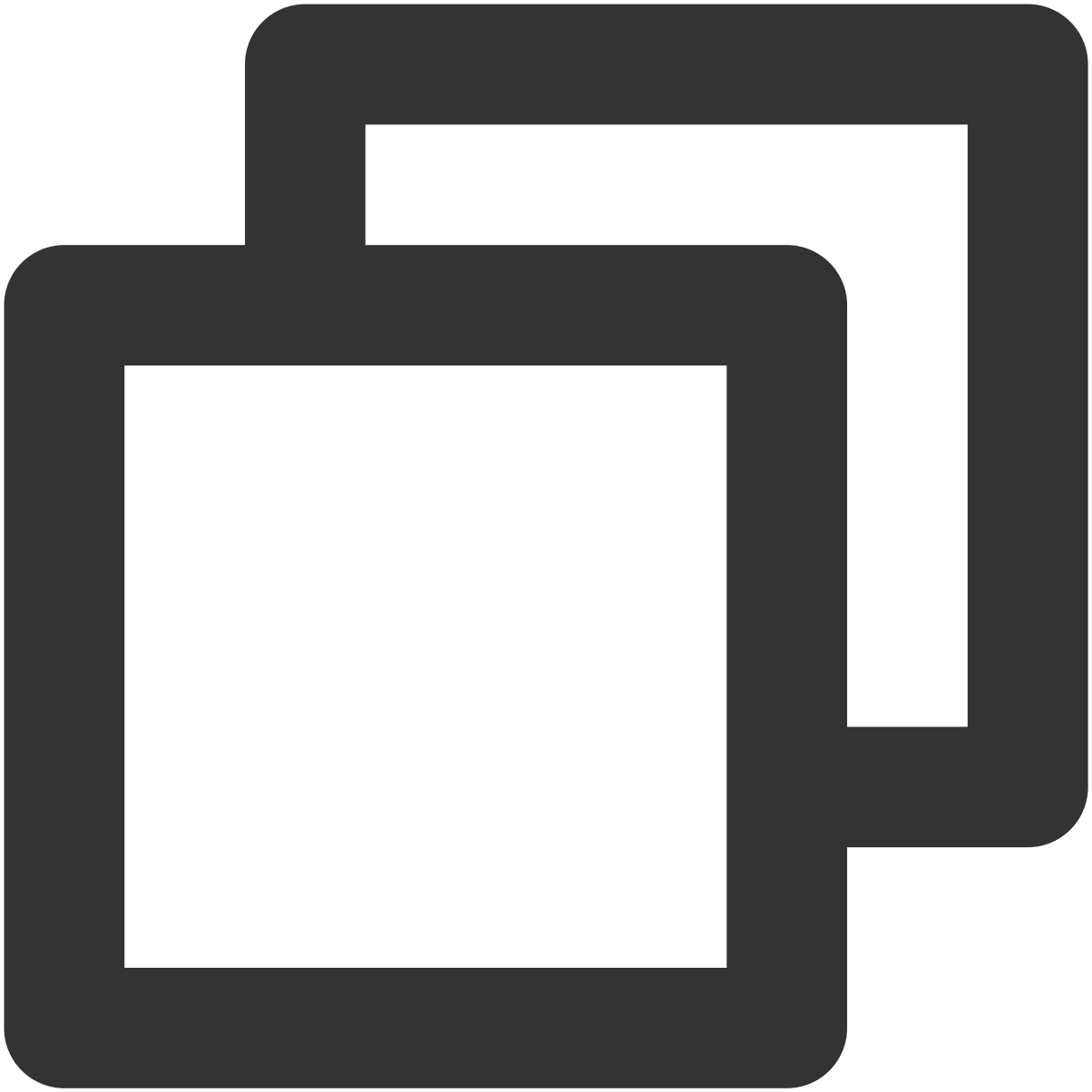


```
CanonicalRequest =
    HTTPRequestMethod + '\\n' +
    CanonicalURI + '\\n' +
    CanonicalQueryString + '\\n' +
    CanonicalHeaders + '\\n' +
    SignedHeaders + '\\n' +
    HashedRequestPayload
```

Field	Description

HTTPRequestMethod	HTTP request method (GET or POST). This example uses `POST` .
CanonicalURI	URI parameter. Slash ("/") is used for API 3.0.
CanonicalQueryString	Query string in the URL of the originating HTTP request. This is always an empty string POST requests and is the string after the question mark (?) for GET requests such as `Limit=10&Offset=0` . Note: `CanonicalQueryString` must be URL-encodec instructed in RFC 3986 with the UTF-8 character set. The applicable standard program language library is recommended. All special characters must be encoded and capitaliz
CanonicalHeaders	Header information for signature calculation, including at least `host` and `content type` . Custom headers can also be added to the signature process to improve the uniqueness and security of the request. Concatenation rules: both the key and value of header should be converted to lowercase with the leading and trailing spaces removed that they are concatenated in the `key:value\\n` format. If there are multiple heade they should be sorted in ASCII ascending order by header key (lowercase). The calcula result in this example is `content-type:application/json; charset=utf-8\\nhost:cvm.tencentcloudapi.com\\n` . Note: `content-type` must mat the content that is actually sent. In some programming languages, a `charset` value automatically added even if it is not specified. In this case, the request sent will be differ from the one signed, and the sever will return a signature verification failure.
SignedHeaders	Header information for signature calculation, indicating the request headers that are inv in the signature process. The request headers must correspond to the headers in `CanonicalHeaders` . `Content-type` and `host` are required headers. Concatenation rules: both the key and value of a header should be converted to lowerca there are multiple headers, they should be sorted in ASCII ascending order by header k (lowercase) and separated by semicolons (;). The value in this example is `content-type;host` .
HashedRequestPayload	Hash value of `Requestpayload` (i.e., the request body, such as `{"Limit": 1, "Filters": [{"Values": ["\\u672a\\u547d\\u540d"], "Name": "instance-name"}]}` in this example). The pseudo-code for calculation is `Lowercase(HexEncode(Hash.SHA256(RequestPayload)))` , which means t SHA256 hashing is performed on the payload of the HTTP request, then hexadecimal encoding is performed, and finally the encoded string is converted to lowercase letters. GET requests, `RequestPayload` is always an empty string. The calculation result i example is `35e9c5b0e3ae67532d3c9f17ead6c90222632e5b1ff7f6e89887f1398934f(`

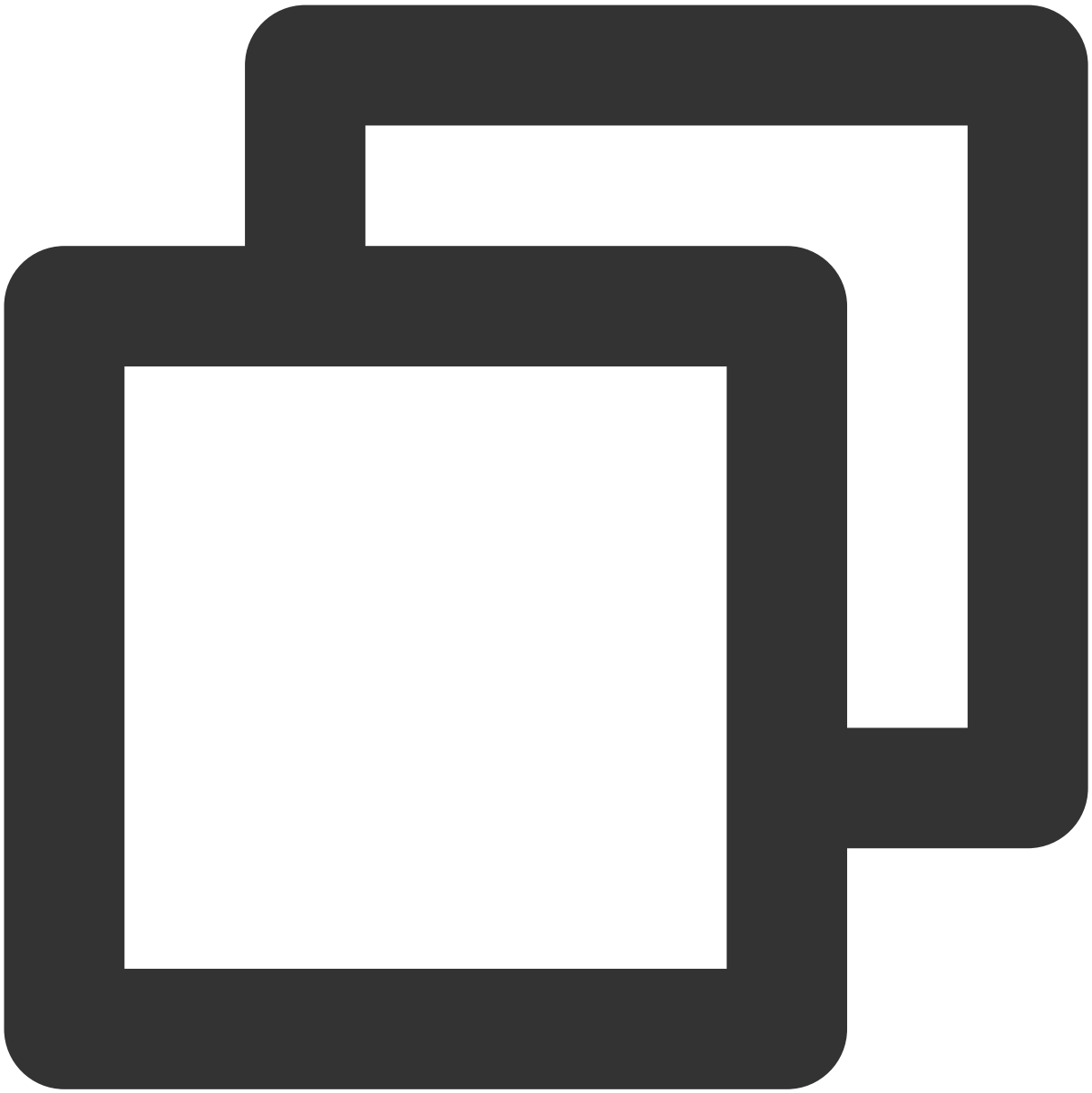
According to the rules above, the canonical request string obtained in the example is as follows:



```
POST
/

content-type:application/json; charset=utf-8
host:cvm.tencentcloudapi.com

content-type;host
35e9c5b0e3ae67532d3c9f17ead6c90222632e5b1ff7f6e89887f1398934f064
```

2. Concatenate the string to sign

Concatenate the string to sign in the following format:



```
StringToSign =
    Algorithm + \\n +
    RequestTimestamp + \\n +
    CredentialScope + \\n +
    HashedCanonicalRequest
```

Field	Description
Algorithm	Signature algorithm, which is always `TC3-HMAC-SHA256` currently.

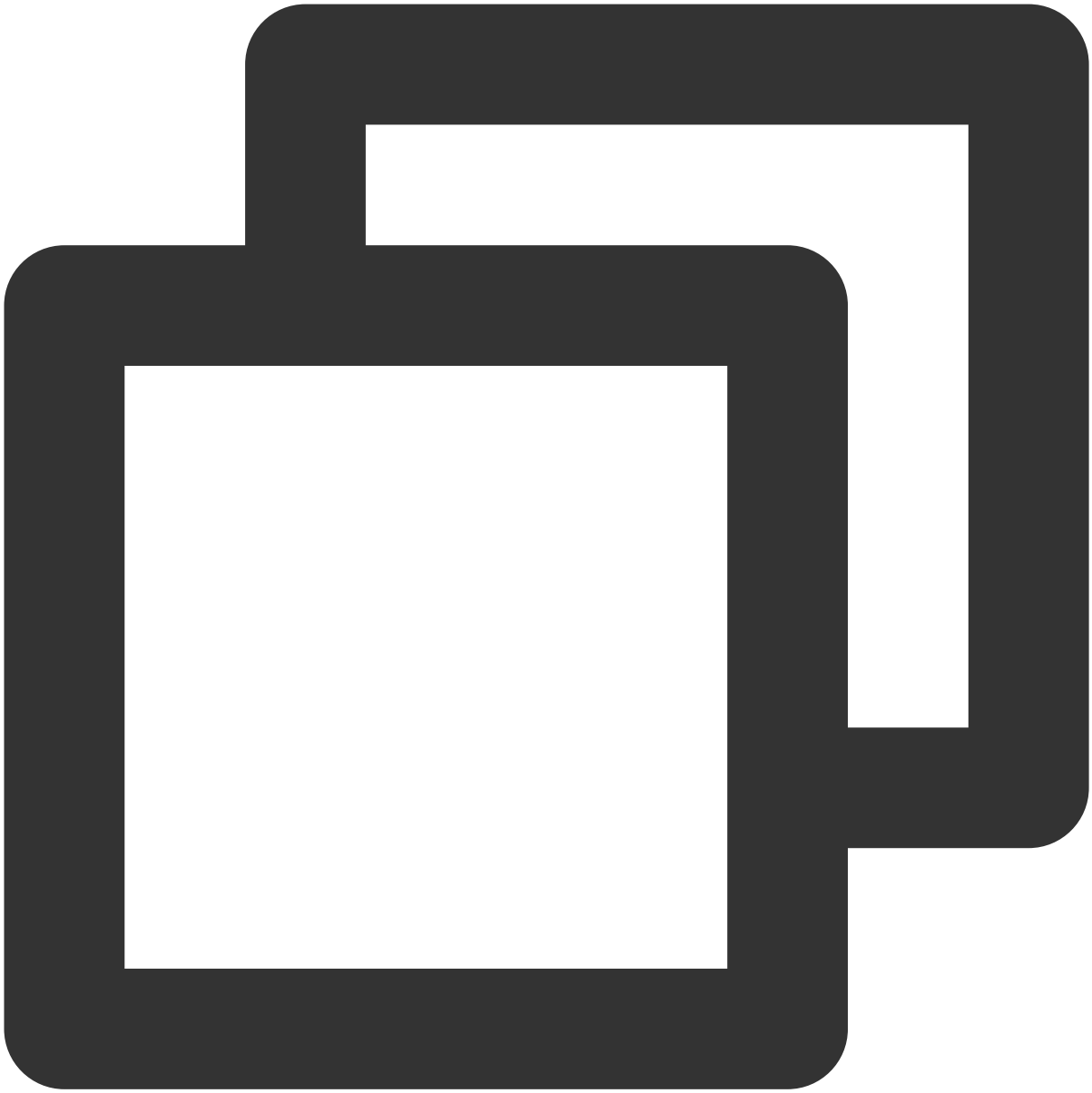
RequestTimestamp	Request timestamp, i.e., the value of the common parameter `X-TC-Timestamp` in request header. It is the UNIX timestamp of the current time in seconds, such as `1551113065` in this example.
CredentialScope	Scope of the credential in the format of `Date/service/tc3_request`, including date, requested service, and termination string (tc3_request). **`Date` indicates a U date, which should match the UTC date converted by the common parameter `TC-Timestamp`.** `service` is the service name, which should match the domain n of the service called. The calculation result in this example is `2019-02-25/cvm/tc3_request`.
HashedCanonicalRequest	Hash value of the canonical request string concatenated in the steps above. The pseu code for calculation is `Lowercase(HexEncode(Hash.SHA256(CanonicalRequest)))`. The calcu result in this example is `5ffe6a04c0664d6b969fab9a13bdab201d63ee709638e2749d62a09ca18d`

Note:

`Date` must be calculated from the timestamp `X-TC-Timestamp` and the time zone is UTC+0. If you add the local time zone information (such as UTC+8) in the system, calls can succeed both day and night but will definitely fail at 00:00. For example, if the timestamp is 1551113065 and the time in UTC+8 is 2019-02-26 00:44:25, the UTC+0 date in the calculated `Date` value should be 2019-02-25 instead of 2019-02-26.

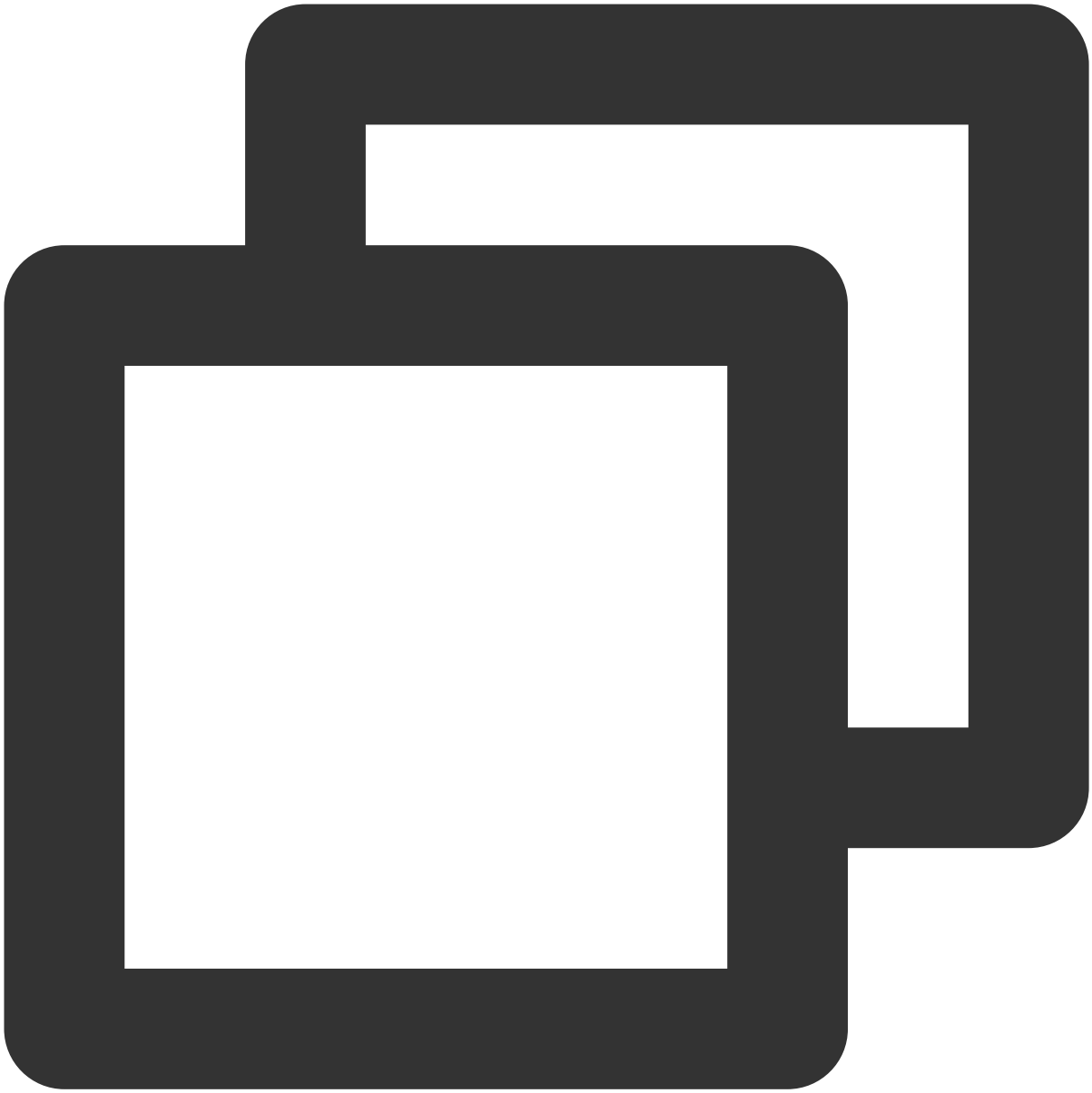
`Timestamp` must be the same as your current system time, and your system time must be in sync with the UTC time. If the difference between the timestamp and your current system time is greater than five minutes, the request will fail. If your system time is out of sync with the UTC time for a prolonged period, the request will fail, and a signature expiration error will be returned.

According to the rules above, the string to sign obtained in the example is as follows:



```
TC3-HMAC-SHA256
1551113065
2019-02-25/cvm/tc3_request
5ffe6a04c0664d6b969fab9a13bdab201d63ee709638e2749d62a09ca18d7031
```

3. Calculate the signature (pseudocode)

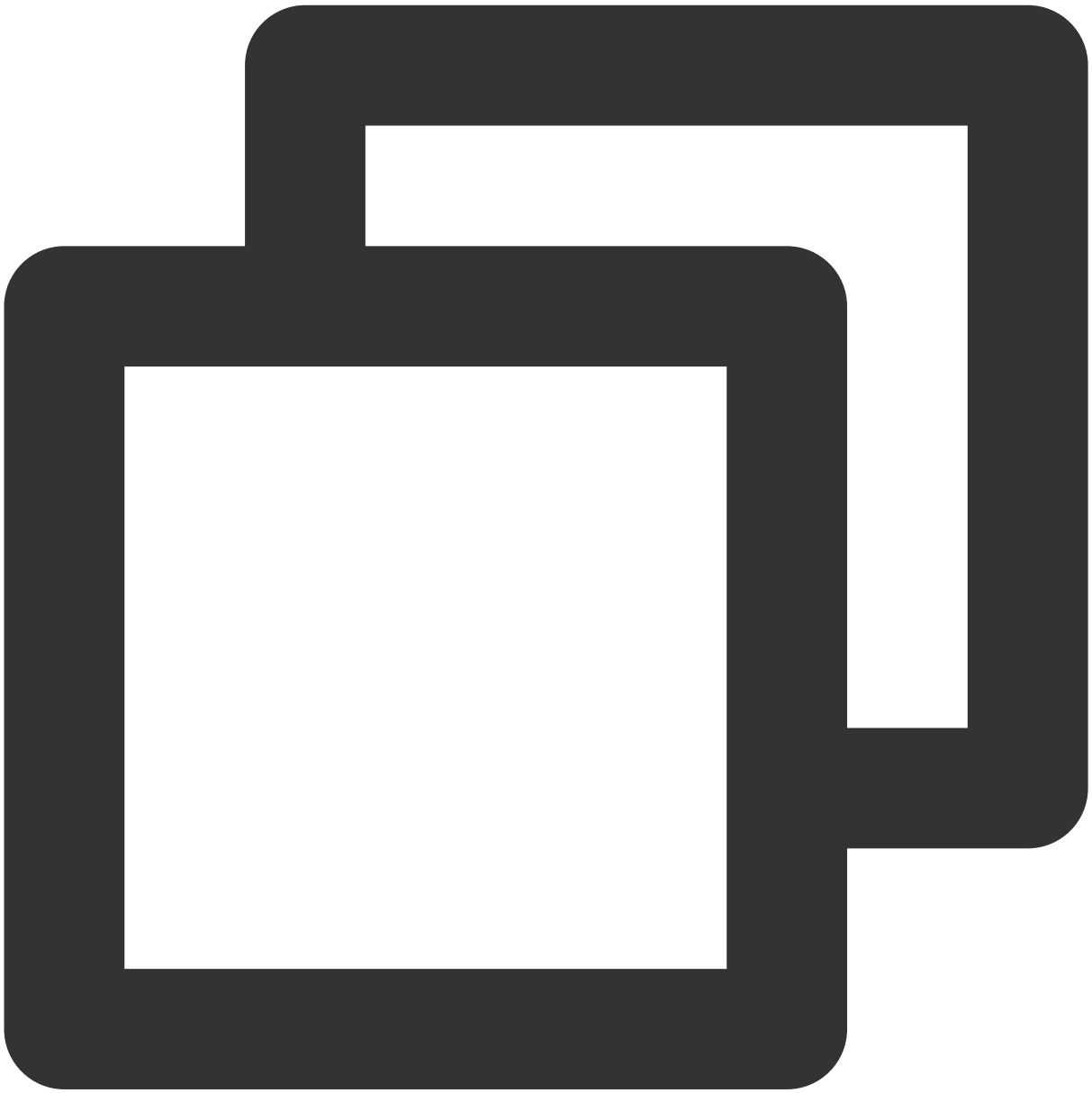


```
SecretKey = "Gu5t9xGARN**********QYCN3EXAMPLE"
SecretDate = HMAC_SHA256("TC3" + SecretKey, Date)
SecretService = HMAC_SHA256(SecretDate, Service)
SecretSigning = HMAC_SHA256(SecretService, "tc3_request")
```

Field	Description
SecretKey	Original `SecretKey` , i.e., `Gu5t9xGARN**********QYCN3EXAMPLE` .
Date	Value of the `Date` field in `Credential` , such as `2019-02-25` in this example.

Service	Value of the `Service` field in `Credential`, such as `cvm` in this example.

The signing result is as shown below:

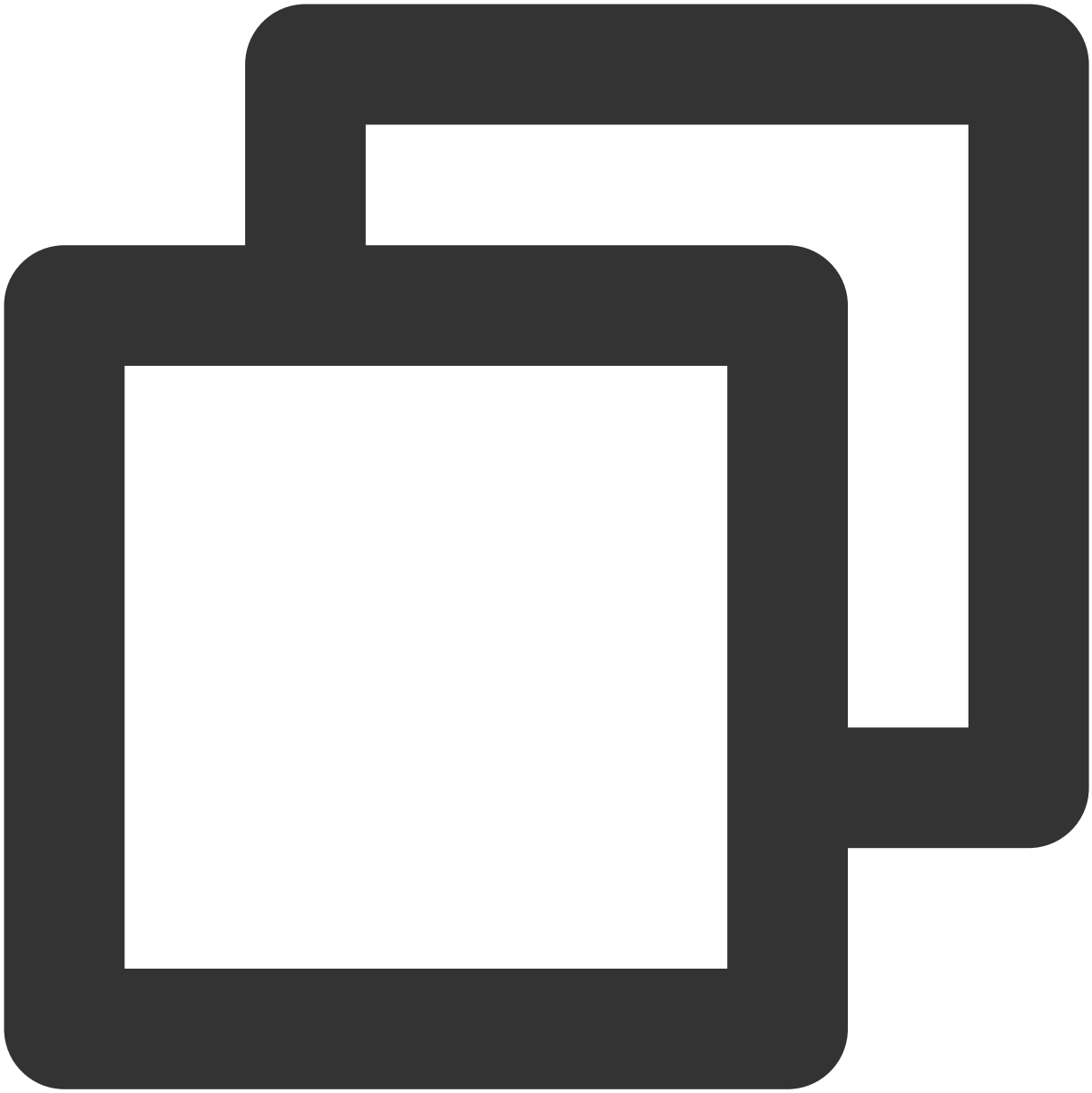


```
Signature = HexEncode(HMAC_SHA256(SecretSigning, StringToSign))
```

The calculation result in this example is `72e494ea8******************************c5a96525168`.

4. Get the call information

Concatenate the `Authorization` string in the following format:

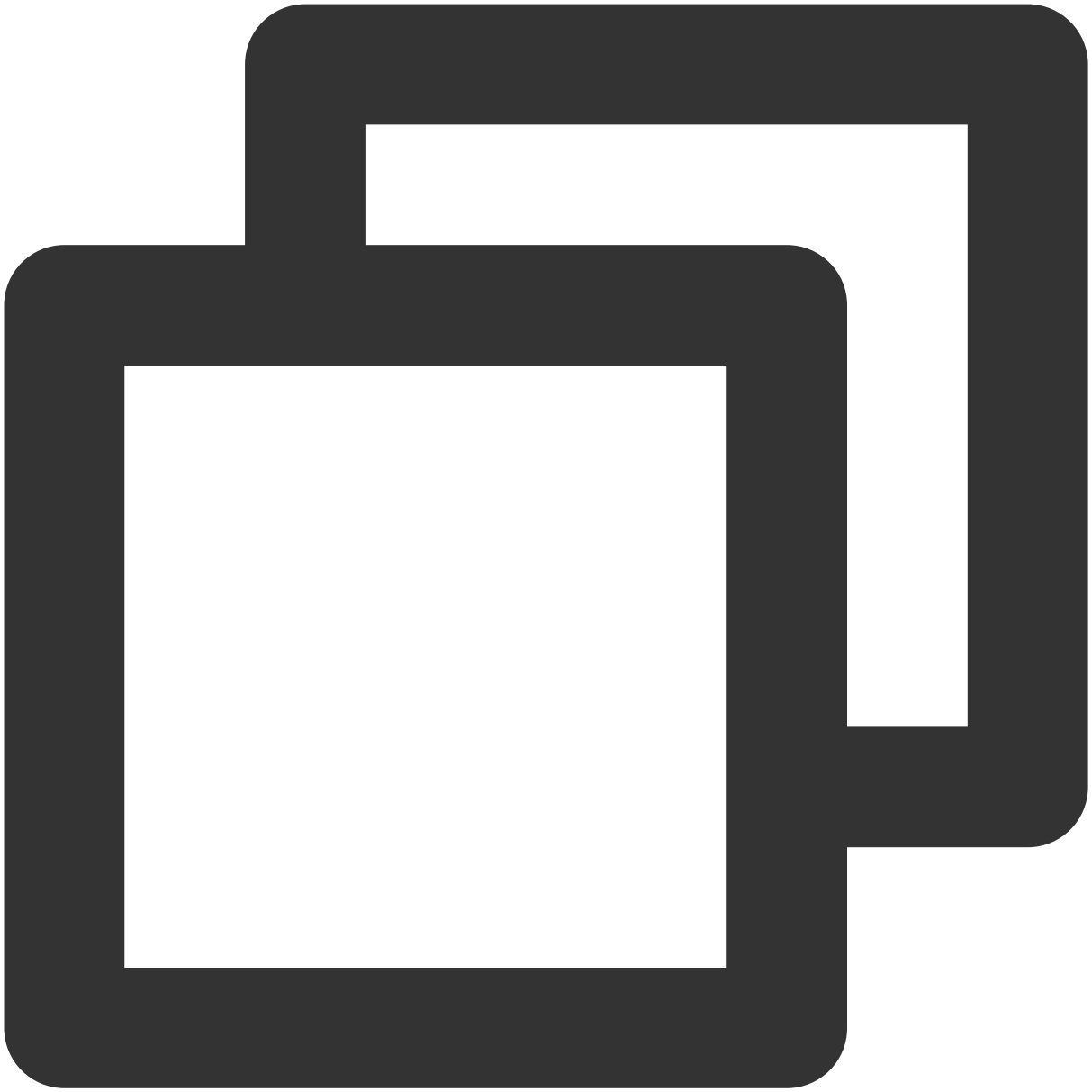


```
Authorization =
    Algorithm + ' ' +
    'Credential=' + SecretId + '/' + CredentialScope + ', ' +
    'SignedHeaders=' + SignedHeaders + ', ' +
    'Signature=' + Signature
```

Field	Description
Algorithm	Signature algorithm, which is always `TC3-HMAC-SHA256` .
SecretId	

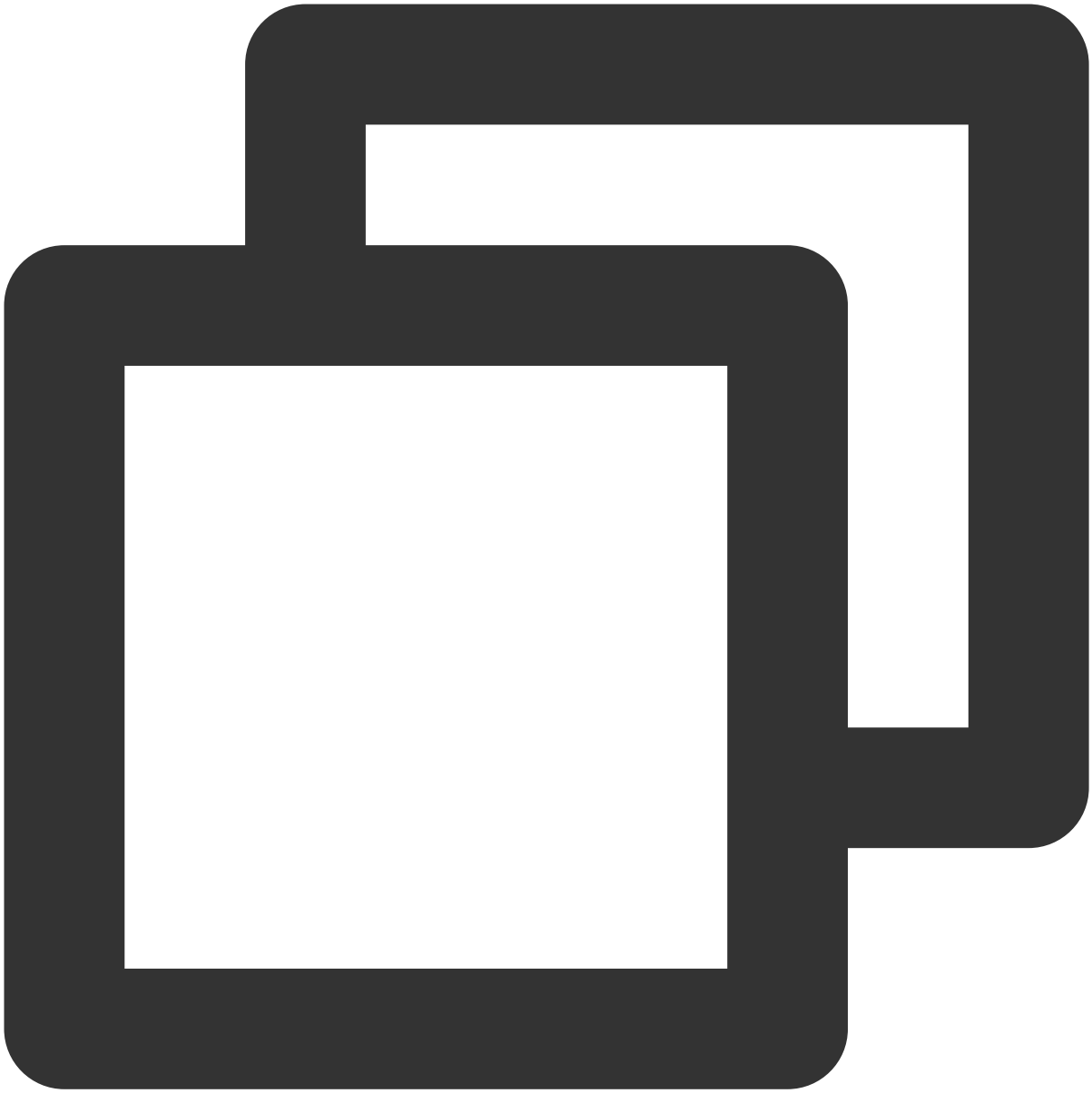
	`SecretId` in the key pair, i.e., `AKIDz8krbsJ5yK*********mLPx3EXAMPLE` .
CredentialScope	Credential scope (see above). The calculation result in this example is `2019-02-25/cvm/tc3_request` .
SignedHeaders	Header information for signature calculation (see above), such as `content-type;host` in this example.
Signature	Signature value. The calculation result in this example is `72e494ea809ad7a8c8f7a450**************f516e8da2f66e2c5a96525168` .

According to the rules above, the values obtained in this example are:



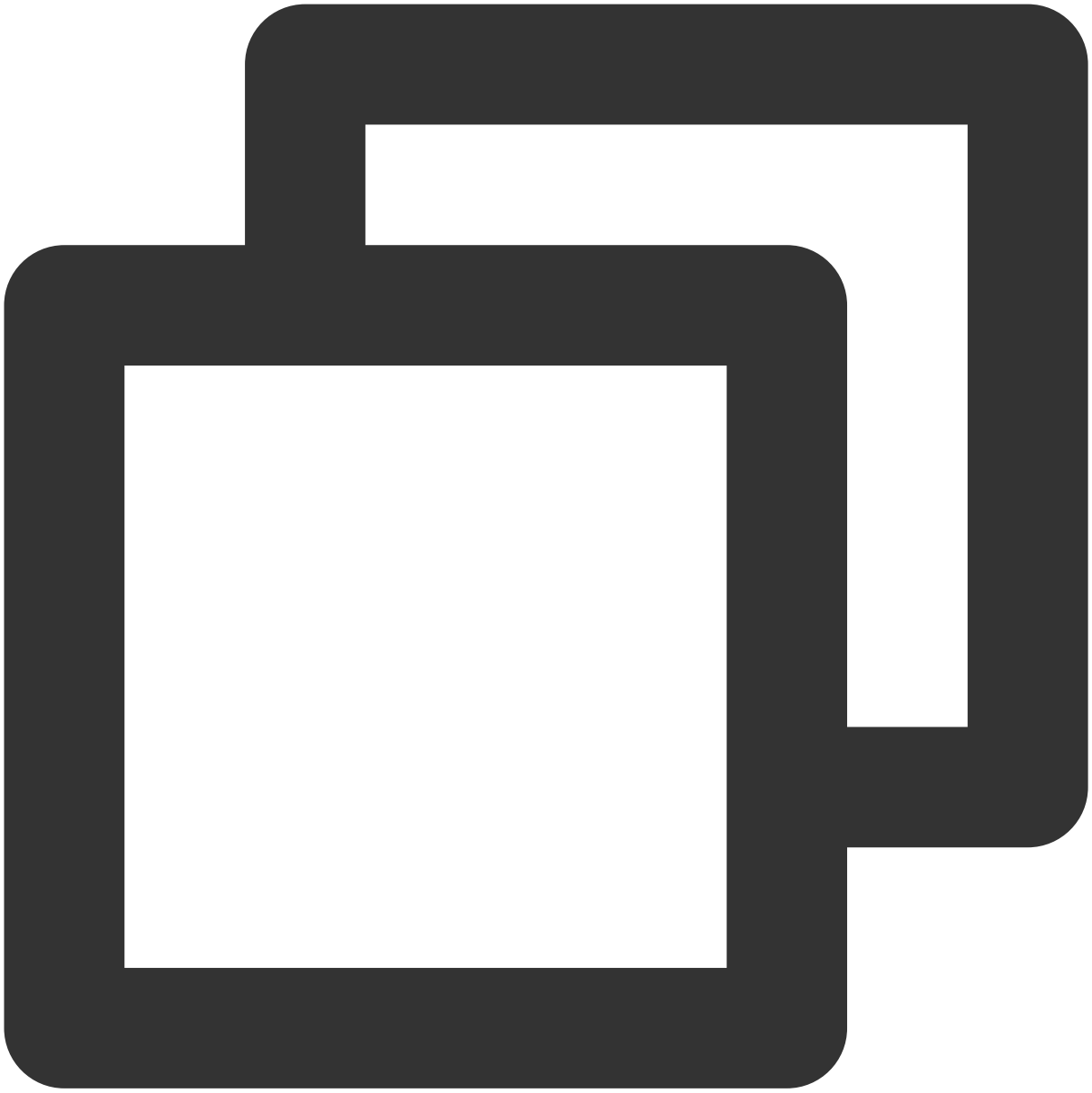
```
TC3-HMAC-SHA256 Credential=AKIDz8krbsJ5yK**********mLPx3EXAMPLE/2019-02-25/cvm/tc3_
```

The complete call information is as follows:



```
POST https://cvm.tencentcloudapi.com/
Authorization: TC3-HMAC-SHA256 Credential=AKIDz8krbsJ5yK**********mLPx3EXAMPLE/2019
Content-Type: application/json; charset=utf-8
Host: cvm.tencentcloudapi.com
X-TC-Action: DescribeInstances
X-TC-Version: 2017-03-12
X-TC-Timestamp: 1551113065
X-TC-Region: ap-guangzhou

{"Limit": 1, "Filters": [{"Values": ["\\u672a\\u547d\\u540d"], "Name": "instance-na
```

5. Sample API 3.0 signature v3



```
# -*- coding: utf-8 -*-
import hashlib, hmac, json, os, sys, time
from datetime import datetime

# Key parameter
secret_id = "`AKIDz8krbsJ5yK*********mLPx3EXAMPLE`"
secret_key = "`Gu5t9xGARN*********QYCN3EXAMPLE`"

service = "cvm"
```

```
host = "cvm.tencentcloudapi.com"
endpoint = "https://" + host
region = "ap-guangzhou"
action = "DescribeInstances"
version = "2017-03-12"
algorithm = "TC3-HMAC-SHA256"
#timestamp = int(time.time())
timestamp = 1551113065
date = datetime.utcfromtimestamp(timestamp).strftime("%Y-%m-%d")
params = {"Limit": 1, "Filters": [{"Name": "instance-name", "Values": [u"unnamed"]}

# ************* Step 1. Concatenate the canonical request string *************
http_request_method = "POST"
canonical_uri = "/"
canonical_querystring = ""
ct = "application/json; charset=utf-8"
payload = json.dumps(params)
canonical_headers = "content-type:%s\\nhost:%s\\n" % (ct, host)
signed_headers = "content-type;host"
hashed_request_payload = hashlib.sha256(payload.encode("utf-8")).hexdigest()
canonical_request = (http_request_method + "\\n" +
                     canonical_uri + "\\n" +
                     canonical_querystring + "\\n" +
                     canonical_headers + "\\n" +
                     signed_headers + "\\n" +
                     hashed_request_payload)
print(canonical_request)

# ************* Step 2. Concatenate the string to sign *************
credential_scope = date + "/" + service + "/" + "tc3_request"
hashed_canonical_request = hashlib.sha256(canonical_request.encode("utf-8")).hexdig
string_to_sign = (algorithm + "\\n" +
                  str(timestamp) + "\\n" +
                  credential_scope + "\\n" +
                  hashed_canonical_request)
print(string_to_sign)

# ************* Step 3. Calculate the signature *************
# Function for calculating signature digest
def sign(key, msg):
    return hmac.new(key, msg.encode("utf-8"), hashlib.sha256).digest()
secret_date = sign(("TC3" + secret_key).encode("utf-8"), date)
secret_service = sign(secret_date, service)
secret_signing = sign(secret_service, "tc3_request")
signature = hmac.new(secret_signing, string_to_sign.encode("utf-8"), hashlib.sha256
print(signature)
```

```
# ************* Step 4. Concatenate the `Authorization` string *************
authorization = (algorithm + " " +
                 "Credential=" + secret_id + "/" + credential_scope + ", " +
                 "SignedHeaders=" + signed_headers + ", " +
                 "Signature=" + signature)
print(authorization)

print('curl -X POST ' + endpoint
      + ' -H "Authorization: ' + authorization + '"'
      + ' -H "Content-Type: application/json; charset=utf-8"'
      + ' -H "Host: ' + host + '"'
      + ' -H "X-TC-Action: ' + action + '"'
      + ' -H "X-TC-Timestamp: ' + str(timestamp) + '"'
      + ' -H "X-TC-Version: ' + version + '"'
      + ' -H "X-TC-Region: ' + region + '"'
      + " -d '" + payload + "'")
```

2. Get an API 3.0 signature v1

The signature algorithm v1 (HmacSHA1 or HmacSHA256) is simple and easy to use, but its functionality and security are not as good as the signature algorithm v3 which is therefore recommended.

If you are using the signature algorithm for the first time, we recommend you use the "signature string generation" feature in API Explorer and select "API 3.0 signature v1" as the signature version, which can generate a signature for demonstration and verification and provides signing examples for certain programming languages. Plus, it can also generate SDK code directly. Seven common open-source programming language SDKs are available for TencentCloud API, including Python, Java, PHP, Go, Node.js, .NET, and C++.

For example, if you call the `DescribeInstances` API to query CVM instances, the request parameters may be as follows:

Parameter Name	Description	Value
Action	Method	DescribeInstances
SecretId	Key ID	`AKIDz8krbsJ5**********mLPx3EXAMPLE`
Timestamp	Current timestamp	1465185768
Nonce	Random positive integer	11886
Region	Instance region	ap-guangzhou
InstanceIds.0	ID of the instance to be queried	ins-09dx96dg
Offset	Offset	0

Limit	Allowed maximum number of output entries	20
Version	API version number	2017-03-12

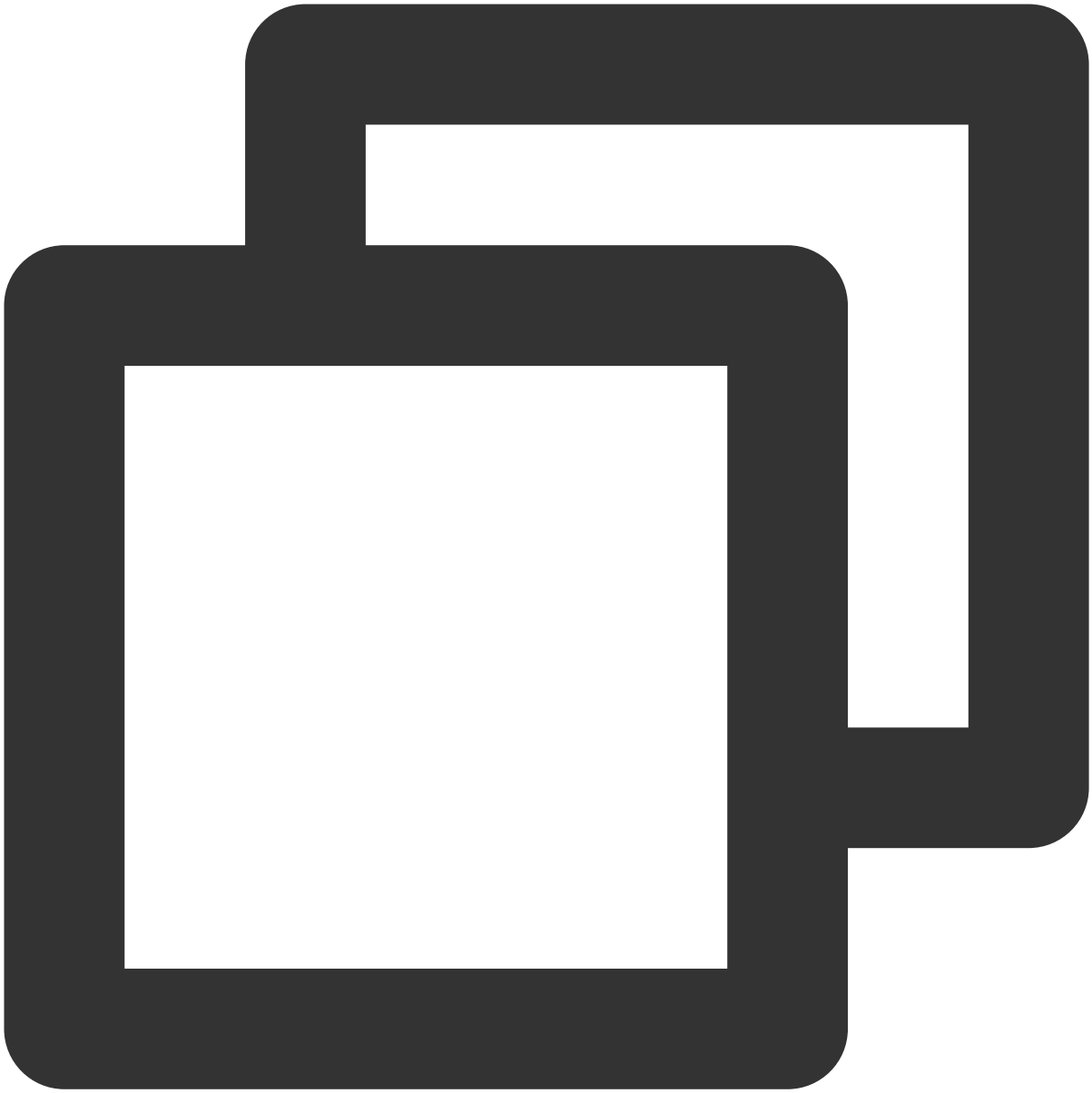
1. Sort parameters

Sort all the request parameters in an ascending lexicographical order (ASCII code) by their names.

Note:

1. The parameters are sorted only by name but not by value.
2. The parameters are sorted based on ASCII code but not in an alphabetical order or by value. For example, `InstanceIds.2` should be arranged behind `InstanceIds.12`. You can complete sorting by using a sorting function in a programming language, such as the `ksort` function in PHP.

The parameters in the example are sorted as follows:



```
{
    'Action' : 'DescribeInstances',
    'InstanceIds.0' : 'ins-09dx96dg',
    'Limit' : 20,
    'Nonce' : 11886,
    'Offset' : 0,
    'Region' : 'ap-guangzhou',
    'SecretId' : 'AKIDz8krbsJ5**********mLPx3EXAMPLE',
    'Timestamp' : 1465185768,
    'Version': '2017-03-12',
}
```

Any other programming languages can be used to sort these parameters as long as the same result is produced.

2. Concatenate the canonical request string

This step generates a request string. Format the request parameters sorted in the previous step into the form of `parameter=value`. For example, for the `Action` parameter, its parameter is `Action` and its value is `DescribeInstances`; therefore, the parameter will be formatted into `Action=DescribeInstances`.

Note:

The `value` is the original value instead of the URL-encoded value.

Then, concatenate the formatted parameters with `&`. The generated request string will be as follows:

```
Action=DescribeInstances&InstanceIds.0=ins-09dx96dg&Limit=20&Nonce=11886&Offset=0&R
```

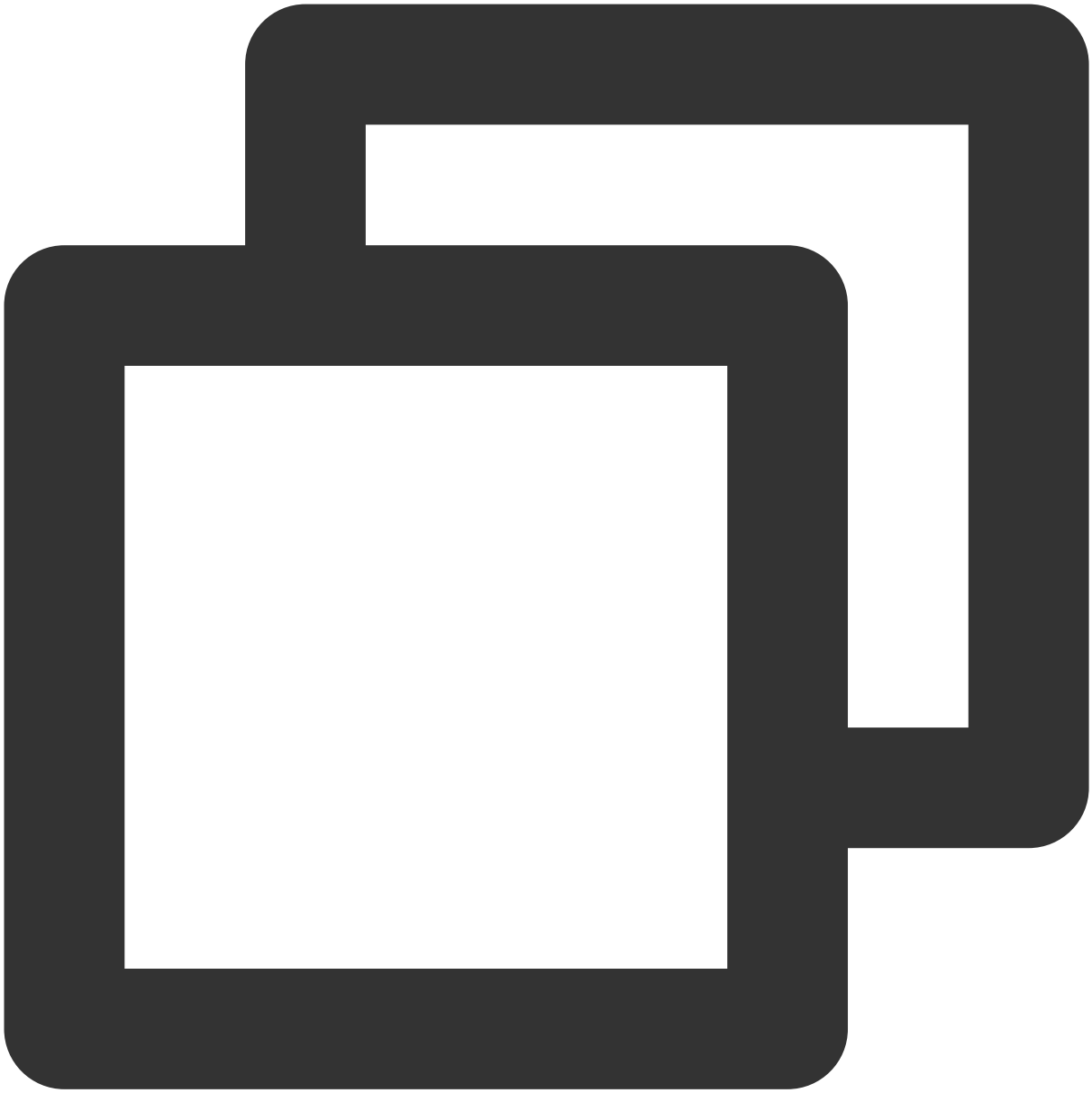
3. Concatenate the string to sign

This step generates the original signature string. The original signature string consists of the following parameters:

1. Request method: POST and GET methods are supported. GET is used here for the request. Please note that the method name should be in all capital letters.
2. Request server: the domain name of the request for querying instances (DescribeInstances) is cvm.tencentcloudapi.com. The actual request domain name varies by the module to which the API belongs. For more information, please see the specific API document.
3. Request path: the request path in the current version of TencentCloud API is fixed to `/` .
4. Request string: the request string generated in the previous step.

The rule for concatenating the original string of the signature is `request method + request server + request path + ? + request string` .

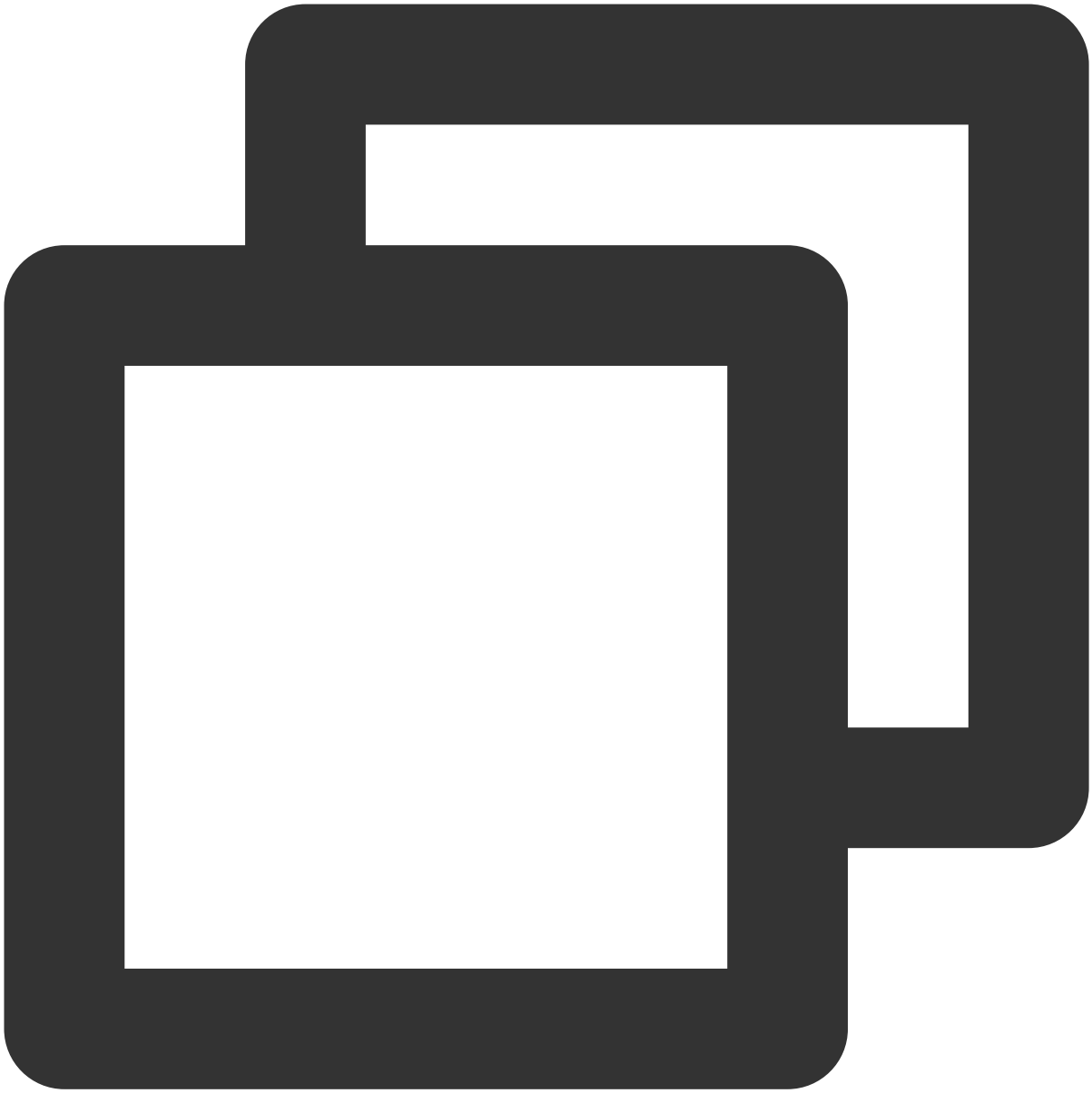
The concatenation result in the example is as follows:



```
GETcvm.tencentcloudapi.com/?Action=DescribeInstances&InstanceIds.0=ins-09dx96dg&Lim
```

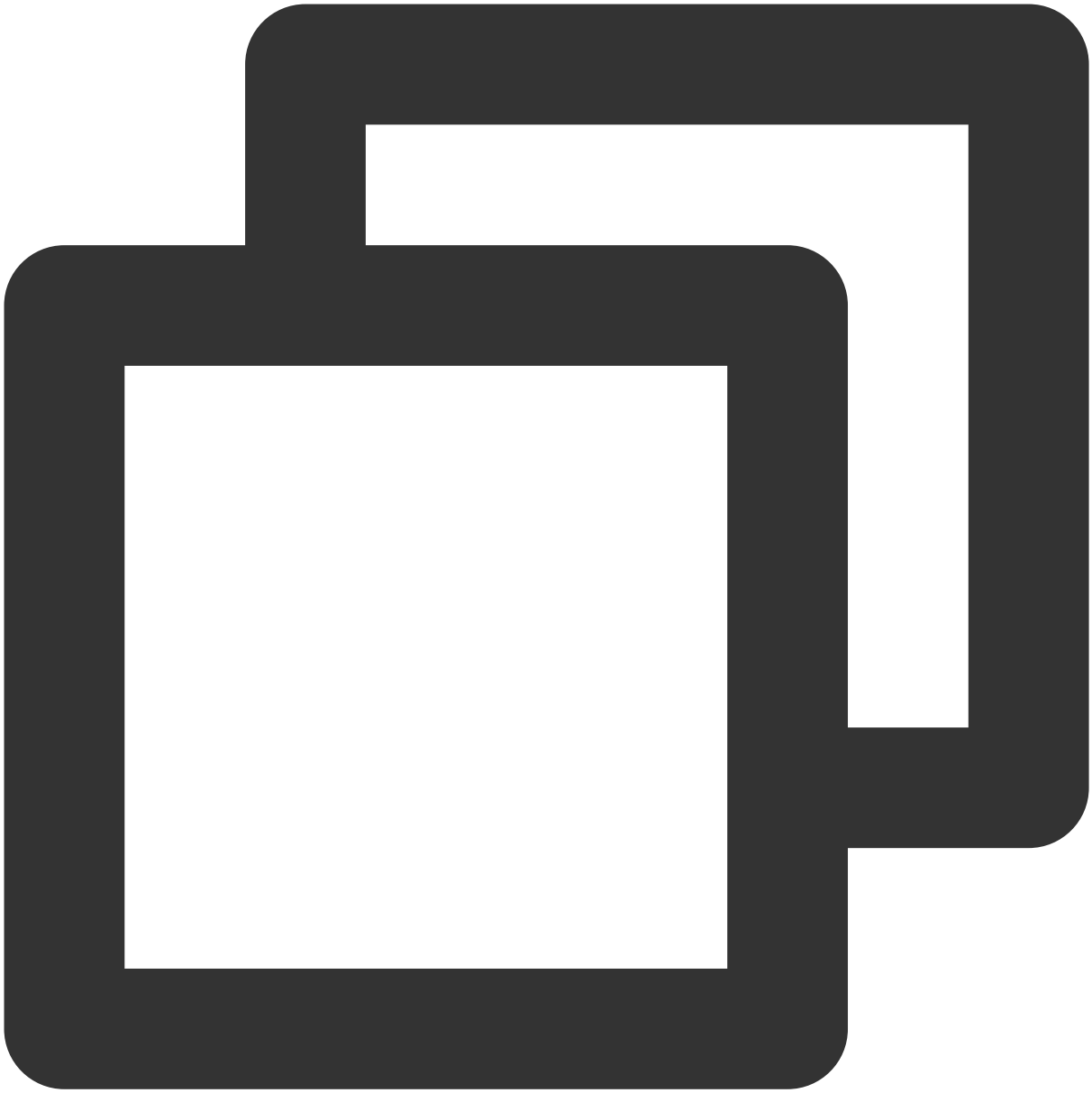
4. Calculate the signature (pseudocode)

This step generates a signature string. Use the HMAC-SHA1 algorithm to sign the **original signature string** obtained in the previous step, and then Base64-encode the generated signature to get the final signature.



```
secret_key = "Gu5t9xGAR********QYCN3EXAMPLE"
s = "GETcvm.tencentcloudapi.com/?Action=DescribeInstances&InstanceIds.0=ins-09dx96d
hmac_str = hmac.new(secret_key.encode("utf8"), s.encode("utf8"),hashlib.sha1).diges
# Final signature string
Signature = base64.b64encode(hmac_str)
```

5. Get the call information and send a request



```
data["Signature"] = base64.b64encode(hmac_str)
print(data["Signature"])  # Final signature string
# The API will be called actually, and fees may be incurred if the call is successf
resp = requests.get("https://" + endpoint, params=data)
print(resp.url)
```

Field	Description
endpoint	Service address, such as `cvm.tencentcloudapi.com` .
data	API parameter of the sample API 3.0 signature v1. **Note:** you should add the calculated

	signature in the format of key-value pair to `data` .

Note:

The key in the example is not real, and the timestamp is not the current system time. If you open this URL in the browser or call it by using commands such as `curl` , an authentication error `The signature expired` will be returned. To obtain a URL that works, you need to replace the `SecretId` and `SecretKey` in this example with your own credentials and use the current system time as the `Timestamp` .

To further explain the signing process, Python is used as examples below to implement the process as described above. The request domain name, API, and parameter values in the above example are used here. The code below is for demonstration only. Please use the SDK for actual development.

6. Encode a signature string

The generated signature string cannot be directly used as a request parameter and needs to be URL-encoded.

For example, if the signature string generated in the previous step is `Eli****************cGeI=` , the final value of the `Signature` request parameter will be `EliP**********************eI%3D` , which will be used to generate the final request URL.

Note:

If you use the GET request method or use the POST request method with `Content-Type` of `application/x-www-form-urlencoded` , all the request parameter values must be URL-encoded (except the parameter key and the equal symbol (=)) before the request is sent. Non-ASCII characters must be encoded with UTF-8 before URL-encoding.

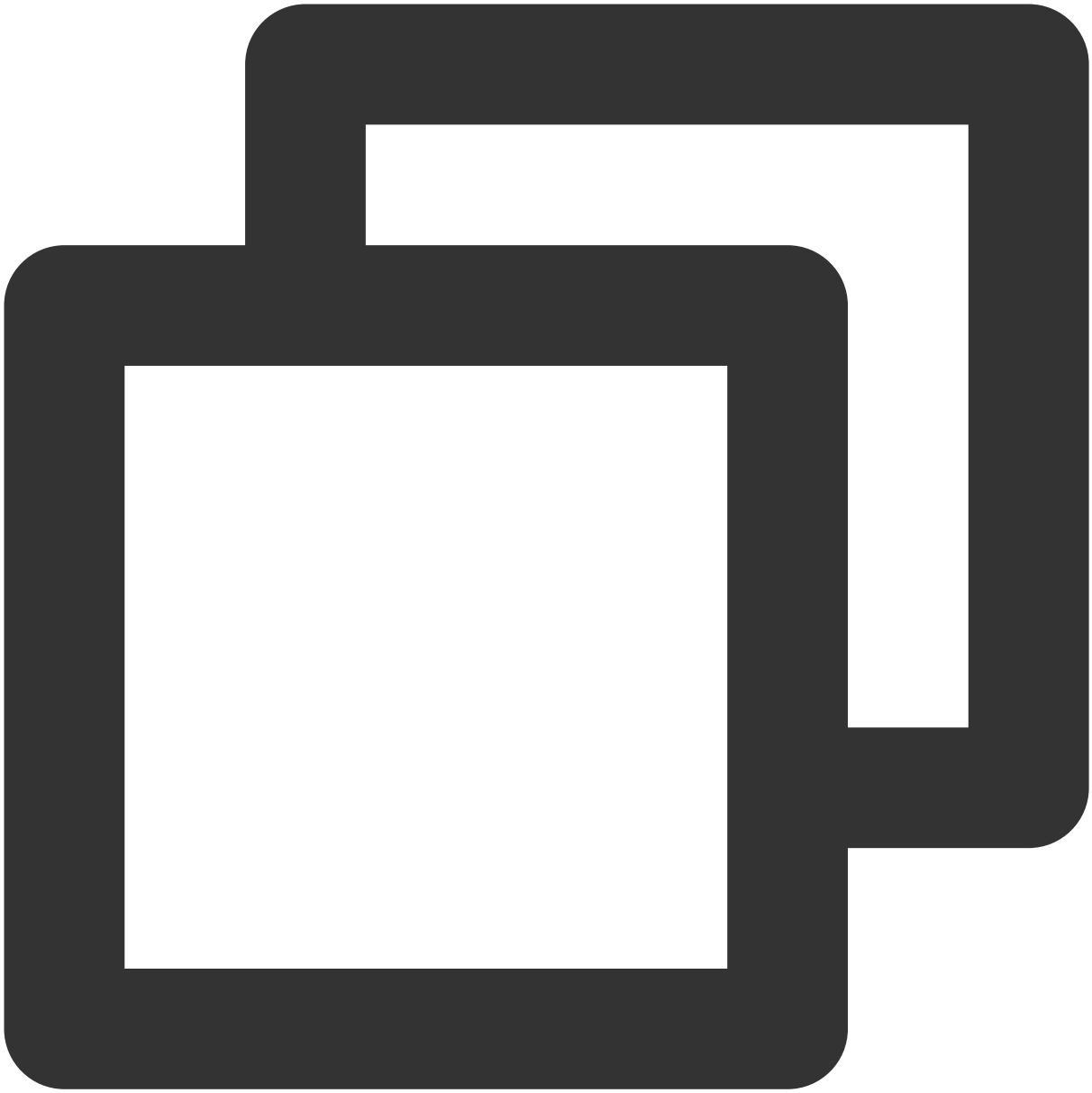
The network libraries of some programming languages automatically URL-encode all parameters. In this case, the signature string does not need to be URL-encoded again; otherwise, two rounds of URL-encoding will cause the signature to fail.

Other parameter values also need to be encoded with RFC 3986. Use %XY in percent-encoding for special characters such as Chinese characters, where "X" and "Y" are hexadecimal characters (0–9 and uppercase A–F). Using lowercase characters will cause an error.

7. Sample API 3.0 signature v1

Note:

In the Python 2 environment, the following `requests` dependency package must be installed first by running `pip install requests` .



```
# -*- coding: utf8 -*-
import base64
import hashlib
import hmac
import time

import requests

secret_id = "AKIDz8k*********LPx3EXAMPLE"
secret_key = "Gu5t9xGA*********YCN3EXAMPLE"
```

```
def get_string_to_sign(method, endpoint, params):
    s = method + endpoint + "/?"
    query_str = "&".join("%s=%s" % (k, params[k]) for k in sorted(params))
    return s + query_str

def sign_str(key, s, method):
    hmac_str = hmac.new(key.encode("utf8"), s.encode("utf8"), method).digest()
    return base64.b64encode(hmac_str)

if __name__ == '__main__':
    endpoint = "cvm.tencentcloudapi.com"
    data = {
        'Action' : 'DescribeInstances',
        'InstanceIds.0' : 'ins-09dx96dg',
        'Limit' : 20,
        'Nonce' : 11886,
        'Offset' : 0,
        'Region' : 'ap-guangzhou',
        'SecretId' : secret_id,
        'Timestamp' : 1465185768, # int(time.time())
        'Version': '2017-03-12'
    }
    s = get_string_to_sign("GET", endpoint, data)
    data["Signature"] = sign_str(secret_key, s, hashlib.sha1)
    print(data["Signature"])
    # The API will be called actually, and fees may be incurred if the call is succ
    # resp = requests.get("https://" + endpoint, params=data)
    # print(resp.url)
```

API 2.0 Signature

This signature version has been disused. We recommend you use **API 3.0 signature** with better performance. If you still need to use it, please go to API Explorer > **Signature Generation** and select **API 2.0 Signature** as the signature version.

Signature Failure

The following error codes may be returned for signature failure. Please resolve the errors accordingly.

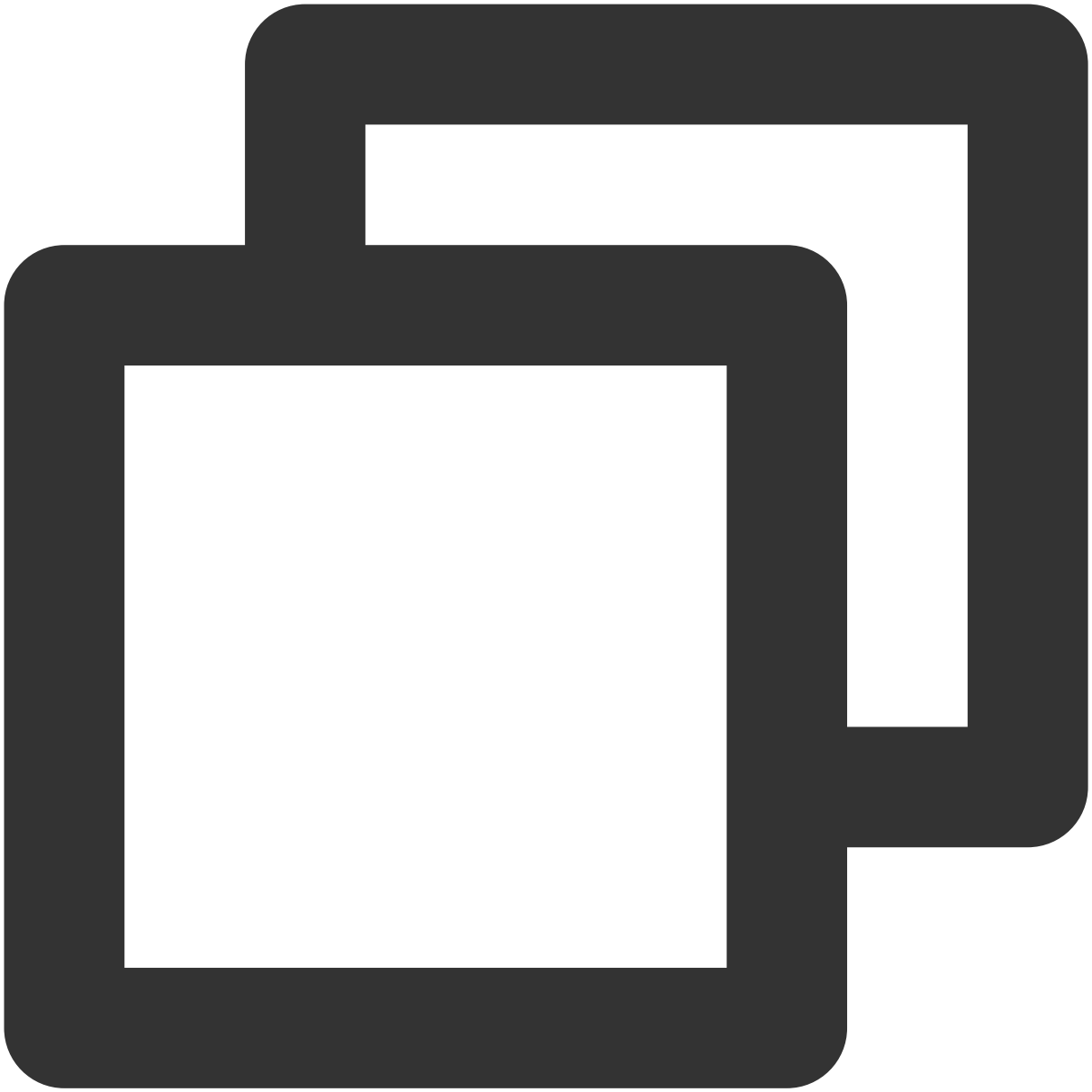
Error Code	Error Description
AuthFailure.SignatureExpire	The signature expired. The difference between the `Timestamp` and

	the server time cannot be greater than five minutes.
AuthFailure.SecretIdNotFound	The key does not exist. Log in to the console and check whether it is disabled or you copied fewer or more characters.
AuthFailure.SignatureFailure	Signature error. It is possible that the signature is calculated incorrectly, the signature does not match the content that is actually sent, or the `SecretKey` is incorrect.
AuthFailure.TokenFailure	Temporary credential token error.
AuthFailure.InvalidSecretId	Invalid key (not TencentCloud API key type).

Returned Result

Successful response

For example, when calling the CVM API `DescribeInstancesStatus` (version: 2017-03-12) to view the status of instances, if the request succeeds, you may see the following response:



```
{
    "Response": {``
        "TotalCount": 0,
        "InstanceStatusSet": [],
        "RequestId": "b5b41468-520d-4192-b42f-595cc34b6c1c"
    }
}
```

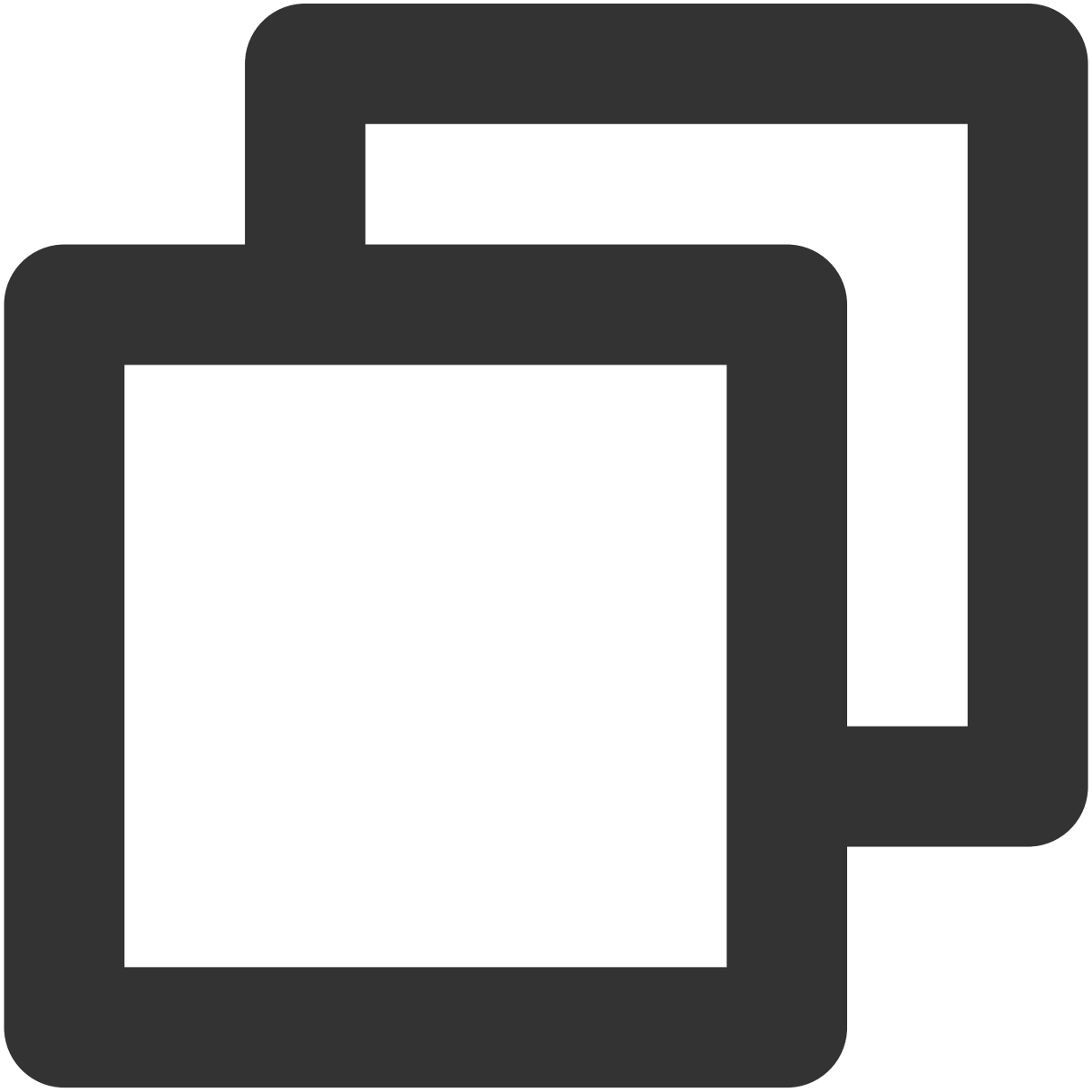
The API will return `Response` , which contains `RequestId` , as long as it processes the request, no matter whether the request is successful or not.

`RequestId` is the unique ID of an API request. It is required to troubleshoot issues.

Any fields other than the common fields are API-specific. For more information on such fields, please see the relevant API documentation. In this example, both `TotalCount` and `InstanceStatusSet` are specific to the `DescribeInstancesStatus` API. Since the user who initiated the request does not have a CVM instance yet, 0 is returned for `TotalCount` and `InstanceStatusSet` is empty.

Error response

If the call fails, you may see the following response:



```
{
```

```
    "Response": {
        "Error": {
            "Code": "AuthFailure.SignatureFailure",
            "Message": "The provided credentials could not be validated. Please che
        },
        "RequestId": "ed93f3cb-f35e-473f-b9f3-0d451b8b79c6"
    }
}
```

`Error` indicates that the request failed. A response for a failed request will always include the `Error`, `Code`, and `Message` fields.

`Code` indicates the specific error code, which is returned when an API request failed. You can use this code to locate the cause and solution of the error in the common or API-specific error code list.

`Message` explains the cause of the error. Note that the returned messages are subject to service updates. The information the messages provide may not be up-to-date and should not be the only source of reference.

`RequestId` is the unique ID of an API request. It is required to troubleshoot issues.

Common error codes

The `Error` field in a response indicates that the API call failed. The `Code` field in `Error` indicates the error code. The following table lists the common error codes that any services may return.

Error Code	Error Description
AuthFailure.InvalidSecretId	Invalid key (not TencentCloud API key type).
AuthFailure.MFAFailure	MFA failure.
AuthFailure.SecretIdNotFound	The key does not exist.
AuthFailure.SignatureExpire	The signature expired.
AuthFailure.SignatureFailure	Signature error.
AuthFailure.TokenFailure	Token error.
AuthFailure.UnauthorizedOperation	No CAM authorization.
DryRunOperation	DryRun Operation. It means that the request would have succeeded, but the DryRun parameter was used.
FailedOperation	The operation failed.
InternalError	Internal error.
InvalidAction	The API does not exist.

InvalidParameter	Incorrect parameter.
InvalidParameterValue	Invalid parameter value.
LimitExceeded	The quota limit is exceeded.
MissingParameter	A parameter is missing.
NoSuchVersion	The API version does not exist.
RequestLimitExceeded	The request rate limit is exceeded.
ResourceInUse	The resource is in use.
ResourceInsufficient	Insufficient resource.
ResourceNotFound	The resource does not exist.
ResourceUnavailable	The resource is unavailable.
UnauthorizedOperation	Unauthorized operation.
UnknownParameter	Unknown parameter error.
UnsupportedOperation	Unsupported operation.
UnsupportedProtocol	Unsupported HTTPS request method. Only GET and POST requests are supported.
UnsupportedRegion	Unsupported region.