

# Smoke

An Android Echo Software Application

About.....	3
Android.....	4
Database Containers.....	5
Discovery via Cryptography.....	6
Fire.....	7
Inflate.....	8
Ozone Address.....	9
Private Public Key Server.....	10
SipHash Identities.....	11
TCP, UDP Protocols.....	12

## About

Smoke is an Android communications research project. The software is composed of a single multi-threaded application. A companion application, SmokeStack, provides mobile server services.

Software sources are available at <https://github.com/textbrowser/smoke> and <https://github.com/textbrowser/smokestack>.

## Android

Smoke has been successfully tested on Android versions 4.4, 5.0, 5.1, 6.0, 7.0, and 7.1. Android versions 4.4, 5.0, and 5.1 are not officially supported.

According to <https://developer.android.com/about/dashboards/index.html>, Smoke supports 92.7% of all Android versions.

## Database Containers

Most of the database fields contain authentically-encrypted values. Some fields contain keyed digests, including keyed digests of binary (false / true) values. Values are stored as  $E(\text{Data}, K_e) \parallel \text{HMAC}(E(\text{Data}, K_e), K_a)$  and  $\text{HMAC}(\text{Data}, K_a)$ .

## Discovery via Cryptography

Cryptographic discovery is a mechanism which allows servers to lighten the computational and data responsibilities of mobile devices.

Shortly after a Smoke instance connects to a SmokeStack service, the Smoke instance shares some non-private material. The material allows a SmokeStack server to transfer messages to their correct destinations. SmokeStack instances routinely distribute gathered, non-expired material to other SmokeStack services, thus creating a network of cooperative SmokeStacks.

Cryptographic Discovery assumes a trustworthy network.

To mitigate replay attacks, Smoke offers SmokeStack instances random identity streams during message-retrieval requests. The identity streams self-expire.

## Fire

Fire creates communication networks between Smoke and Spot-On. Authentication and encryption key-generation is described below. The Digest Key should contain at least 48 random bytes.

```
authentication_key = Digest Key
encryption_key := pbkdf2(Salt,
                          Channel || "aes256" || "sha384",
                          10000,      // Iteration Count
                          2304)       // Bits (288 Bytes)
encryption_key := encryption_key[0 ... 32]
```

## Inflate

Smoke expands text-messaging data to 8192 bytes. If the provided data exceeds 8192 bytes, Smoke expands the provided data by  $1024 + \text{mod}(\text{data length}, 2)$  bytes.



## Ozone Address

An Ozone address is a pseudo-private string which identifies a virtual entity. Smoke and SmokeStack utilize Ozones as a means of retrieving and storing offline messages and public key pairs. Smoke supports one Ozone while SmokeStack supports infinitely many. Ozone addresses must be shared separately. Retrieved messages are only meaningful within the context of a session. It is possible for multiple Smoke parties to house distinct Ozones if common SmokeStack instances are aware of the distinct Ozone addresses.

Please note that public Ozone addresses will introduce denial of service vulnerabilities.

## **Private Public Key Server**

In addition to housing messages, SmokeStack also serves as a private public key server. A SmokeStack administrator is responsible for coordinating the storage of public key pairs of participants. Participants may request public key pairs of specific participants via Ozone addresses.

## SipHash Identities

Exchanging public key pairs is often an involved process. Smoke implements the pseudo-random function SipHash so as to simplify the process. The SipHash function generates outputs of 8 bytes (16 characters hexadecimal). These short strings are easily memorized and/or distributed via other communications applications. SipHash identities are generated as follows:

```
id := siphash(public-encryption-key || public-signature-key,
              pbkdf2(sha512(public-encryption-key || public-signature-key), // Salt
                    public-encryption-key || public-signature-key,
                    4096, // Iteration Count
                    128)) // Bits (16 Bytes)
```

Non-confidential authentication and encryption key streams from SipHash identities are generated as follows:

```
keystream1 := pbkdf2(sha512(id), // Salt
                    id,
                    4096,          // Iteration Count
                    160)          // Bits (20 Bytes)

keystream2 := pbkdf2(sha512(id), // Salt
                    keystream1,
                    1,             // Iteration Count
                    768)          // Bits (96 Bytes)
```

### Public Data

Chat Encryption Key

Algorithm: RSA

Fingerprint: 27:1ef6:31:d7:67:4b:b3:82:00:4b:59:3c:d9:16:41:93:47:a5:a3:c5:17:1e:5f:

70:56:06:cc:a8:de:f2:1d:ea:b4:ca:d7:99:34:a0:a6:8f:27:2e:df:9a:78:7c:43:a1:a1:bc:63:3f:51:e2:9a:83:1f:73:66:22:63:01:f6

Format: X.509

Size: 3072

Chat Signature Key

Algorithm: RSA

Fingerprint: a3:42:81:f1:34:d1:dd:c8:2f:2c:1d:a3:c4:95:31:17:79:73:d7:b2:6c:df:9c:91:8e:1c:07:6f:42:af:16:31:4a:9c:

69:7d:d8:b6:de:5f:ab:8a:b1:58:38:ae:96:ec:37:fd:ef:fc:21:3d:a4:c2:db:36:a3:80:92:fb:ee:5e

Format: X.509

Size: 3072

**SipHash Chat ID**

**@39B8-3DE5-A567-9C6F**

RESET SMOKE

The transport keys which are generated from SipHash identities may be used for exchanging public-key data via the Echo Public Key Share (EPKS) protocol.

It is impossible to avoid SipHash collisions as there are infinitely-many inputs and a limited number of outputs.

## TCP, UDP Protocols

Smoke supports both the TCP and UDP network protocols. Multicast and unicast UDP varieties are provided. Multiple clients may be defined via Settings. A limit on the number of clients is not imposed. When defining neighbors, one may define SmokeStack and/or Spot-On neighbors. SmokeStack, the companion application of Smoke, offers mobile server services as well as message and public-key storage.

### Neighbor Servers

Control	Remote
Action ▾	Control: Connect Status: Connected 192.168.178.18:4710:TCP 192.168.178.165:59430 Proxy: Remote Certificate's Fingerprint: 35:88:56:47:8a:b5:7f:0d:9d:4b:6f:48:16:16:4d:0f:a8:d3:a6:6f:67:10:fd:33:c0:d1:06:64:64:ee:db:3b:a0:11:30:d5:55:7f:cc:b0:a6:89:33:08:ea:2d:7f:c3:eb:90:89:6d:bd:1e:5a:47:f8:fc:f1:7b:d9:7f:34:fc Session Cipher: TLS_ECDHE_RSA_WITH_AES_128_GCM_SHA256 Temp. Queued: 0 / 256 In: 48.23 KiB Out: 25.27 KiB Outbound Queued: 0 Uptime: 1:52 Min.
Action ▾	Control: Connect Status: Connected 192.168.178.100:4710:TCP 192.168.178.165:46678 Proxy: Remote Certificate's Fingerprint: ac:e0:fc:62:a3:90:d1:89:20:3e:f7:b5:ee:31:ea:7f:87:f9:67:86:38:d1:e5:44:37:ae:30:b3:1e:ea:e0:c8:70:23:8f:9f:31:70:7f:1b:04:02:5a:15:71:81:f0:ce:d2:e2:7c:66:7e:86:61:80:85:78:02:d7:74:b3:cc:3a Session Cipher: TLS_ECDHE_RSA_WITH_AES_256_GCM_SHA384 Temp. Queued: 0 / 256 In: 10.42 KiB Out: 25.27 KiB Outbound Queued: 0 Uptime: 1:52 Min.

☒ Automatic Refresh ☒ Details

☐ Echo

REFRESH

IP Address

4710

Scope ID

☒ IPv4 ☐ IPv6

TCP ▾

Proxy IP Address

Proxy Port

HTTP ▾

ADD

RESET FIELDS