Summary

The Citizen Lab is an academic research group based at the Munk School of Global Affairs & Public Policy at the University of Toronto in Toronto, Canada.

We analyzed Baidu Input Method as part of our ongoing work analyzing popular mobile and desktop apps for security and privacy issues. We found privacy and security weaknesses in the encryption used by the Android and iOS versions of Baidu Input Method. To address these issues, we suggest using HTTPS or TLS rather than custom-designed network protocols.

Platform	**File/Package Name**	**Version analyzed**
Android	baiduinput_AndroidPhone_1000e.apk (com.baidu.input)	11.7.19.9
iOS	com.baidu.inputMethod	11.7.20

Table 1: The mobile versions of Baidu Input Method that we analyzed.

Findings

We found that the Android version transmitted keystrokes information via UDP packets to udpolimeok.baidu.com and that the iOS version transmitted keystrokes to udpolimenew.baidu.com. Both the Android and iOS versions transmitted these keystrokes according to a cryptographic protocol whose payload begins with the bytes 0x04 0x00. In the remainder of this section we detail multiple weaknesses in this protocol in the Android and iOS versions.

Android and iOS

To encrypt keystroke information, the Android and iOS versions of Baidu Input Method use elliptic-curve Diffie-Hellman and a pinned server public key (pk_s) to establish a shared secret key for use in a modified version of AES.

Upon opening the keyboard, before the first outgoing message is sent, the application randomly generates a client Curve25519 key pair, which we will call sk_c and pk_c. Then, a Diffie-Hellman shared secret k is generated using sk_c and the pinned pk_s key. The first 16 bytes of pk_c are reused as the IV for encryption, and k is used as the symmetric encryption key. The resulting encrypted ciphertext is then sent along with pk_c to the server. The server can then obtain the same Diffie-Hellman shared secret k from sk_s and pk_c to decrypt the ciphertext.

Baidu Input Method on Android and iOS encrypts data using a modified version of AES which mixes bytes differently and uses a modified counter (CTR) mode, illustrated in Figure 1.

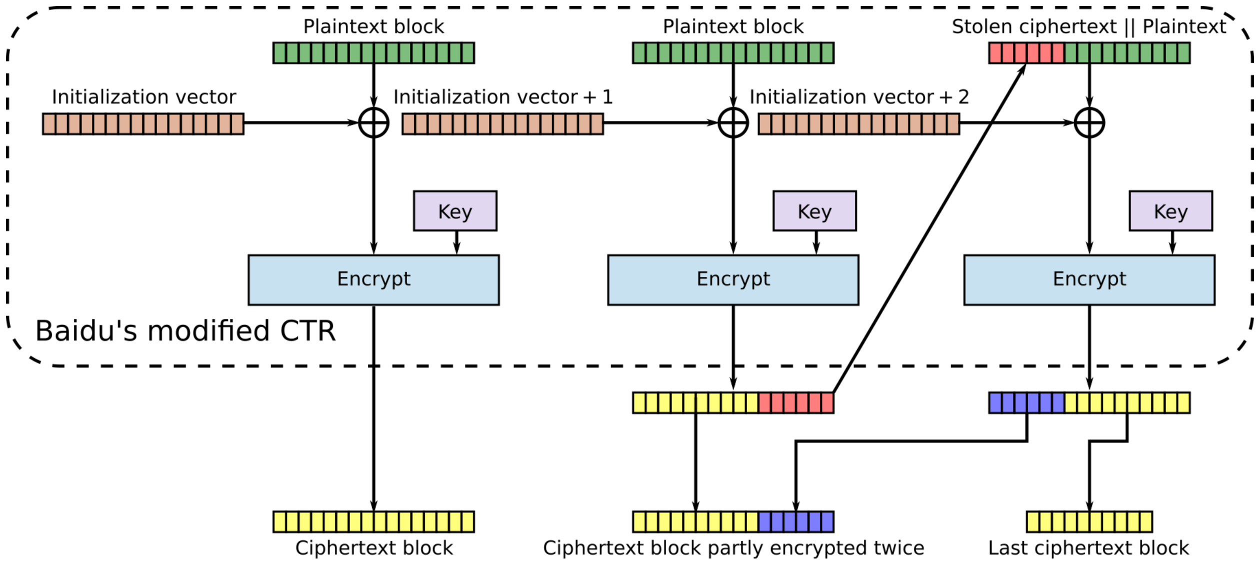

Figure 1: Illustration of modified CTR mode encryption scheme used by Baidu Input Method on Android and iOS. Adapted from this figure.

Generally speaking, any CTR cipher mode involves combining an IV with the value *i* of some counter, whose combination we shall notate in this document as IV + *i*. Most commonly, the counter value used for block *i* is simply *i*, i.e., it begins at zero and increments for each subsequent block, and Baidu's implementation follows this convention. There is no standard way to compute IV + *i* in CTR mode, but the way that Baidu Input Method the IV and *i* is by adding *i* to the left-most 32-bits of the IV, interpreting the IV and counter value in little-endian byte order. If the sum overflows, then no carrying is performed on bytes to the right of this 32-bit value. The implementation details we have thus far described do not deviate from a typical CTR implementation. However, where Baidu's modified CTR mode differs from ordinary CTR mode is in how the value IV + *i* is used during encryption. In ordinary CTR mode, to encrypt block *i* with key *k*, you would compute (plain$_i$ XOR encrypt(IV + *i*, *k*)). In Baidu's modified CTR mode, to encrypt block *i*, you would compute encrypt(plain$_i$ XOR (IV + *i*), *k*). As we will see later, this deviation will have implications to the security of the algorithm.

While ordinarily CTR mode does not require the final block length to be a multiple of the cipher's block size (in the case of AES, 16), Baidu's modified CTR mode does not automatically possess this property but rather achieves it by employing ciphertext stealing. If the final block length *n* is less than 16, Baidu's implementation encrypts the final 16 byte block by taking the last (16 - *n*) bytes of the penultimate ciphertext block and prepending them to the *n* bytes of the ultimate plaintext block. The encryption of the resultant block fills the last (16 - *n*) bytes of the penultimate ciphertext block and the *n* bytes of the final ciphertext block. Note, however, that this practice only works when the plaintext consists of at least two blocks. Therefore, if there exists only one plaintext block, then Baidu's implementation right-zero-pads that block to be 16 bytes.

Privacy issues with IV reuse

Since the IV and key are both directly derived from the client key pair, the IV and key are reused until the application generates a new key pair. This only happens when the user restarts the phone, or when the user switches to a different keyboard and back. From our testing, we have observed the same key and IV in use for over 24 hours. There are various issues that arise from key and IV reuse.

Reusing the same IV and key means that the same inputs will encrypt to the same encrypted ciphertext. Additionally, due to the way the block cipher is constructed, if block-sized portions of the plaintexts are the same, they will encrypt to the same ciphertext blocks. As an example, if the second block of two plaintexts are the same, the second block of the corresponding ciphertexts will be the same. Even within the same plaintext, if blocks are similar to each other, they could encrypt to the same ciphertext blocks.

Weakness in cipher mode

The electronic codebook (ECB) cipher mode is notorious for having the undesirable property that equivalent plaintext blocks encrypt to equivalent ciphertext blocks, allowing patterns in the plaintext to be revealed in the ciphertext (see Figure 2 for an illustration).

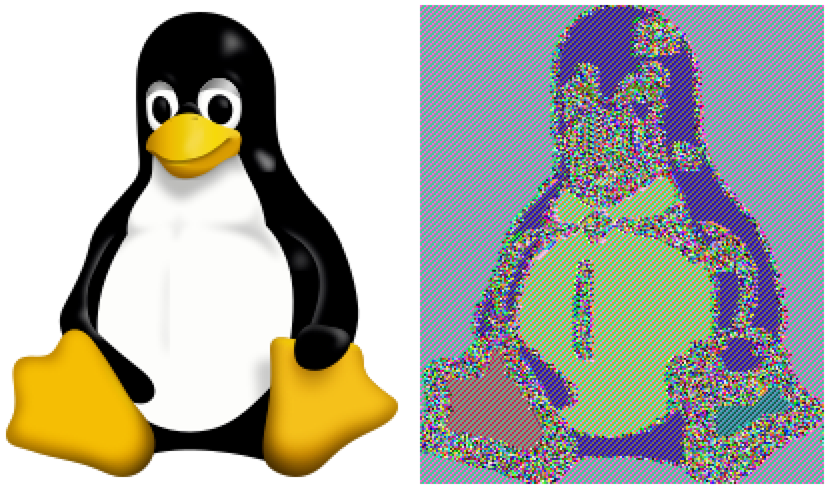

Figure 2: When a bitmap image (left) is encrypted in ECB mode, patterns in the image are still visible in the ciphertext (right). Adapted from these figures.

While the modified CTR mode used by Baidu does not as flagrantly reveal patterns as ECB mode, there do exist circumstances in which patterns in the plaintext can still be revealed in the ciphertext. Specifically, there exist circumstances in which there exists a counter-like pattern in the plaintext which can be revealed by the ciphertext (see Figure 3 for an example). These circumstances are possible due to the fact that (IV + *i*) is XORed with each plaintext block *i* before encryption. Thus, if the plaintext exhibits similar counting patterns as (IV + *i*), then for multiple blocks the value ((IV + *i*) XOR plaintext block *i*) may be equivalent and thus encrypt to an equivalent ciphertext.

Block	Plaintext																Ciphertext															
0	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	e2	d4	00	1c	c6	5d	80	33	0c	b9	48	7d	d5	27	72	7a
1	01	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	e2	d4	00	1c	c6	5d	80	33	0c	b9	48	7d	d5	27	72	7a

Figure 3: When encrypted with the randomly generated key “\x96f\x08\xd1o\x80\x82\x86\xa7\xb7\xdaC\x96\xee\xd1\xa2” and IV “H[T\x92\x0c\x80\xa6)o\x95\xe5\xc5j=\xe2” using Baidu’s modified CTR mode, the above plaintext blocks encrypt to the same ciphertext.

More generally, this cipher mode fails to provide the cryptographic property of diffusion. Specifically, if an algorithm provides diffusion, then, when we change a single bit of the plaintext, we expect half of the bits of the ciphertext to change. However, the example in Figure 3 illustrates a case where changing a single bit of the plaintext to form the following block caused zero bits of the ciphertext to change in the following block, a clear violation of the expectations of this property. The property of diffusion is vital in secure cryptographic algorithms so that patterns in the plaintext are not visible as patterns in the ciphertext.

Other privacy and security weaknesses

There are other weaknesses in the custom encryption protocol designed by Baidu Input Method that are not consistent with the standards for a modern encryption protocol used by hundreds of millions of devices.

Forward secrecy issues with static Diffie-Hellman

The use of a pinned static server key means that the cipher is not forward secret, a property of other modern network encryption ciphers like TLS. If the server key is ever revealed, any past message where the shared secret was generated with that key can be successfully decrypted.

Lack of message integrity

There are no cryptographically secure message integrity checks, which means that a network attacker may freely modify the ciphertext. There is a CRC32 checksum calculated and included with the plaintext data, but a CRC32 checksum does not provide cryptographic integrity, as it is easy to generate CRC32 checksum collisions. Therefore, modifying the ciphertext may be possible. In combination with the issue concerning key and IV reuse, this protocol may be vulnerable to a swapped block attack.

Mitigation

In order to address the reported issues, Baidu Input Method should secure all transmissions using a standard, well-tested network encryption protocol such as HTTPS, TLS, or QUIC instead of relying on custom-designed cryptography to secure the transmission of sensitive user data.