Cryptanalysis of a data security protection scheme for VoIP

C. Li, S. Li, D. Zhang and G. Chen

Abstract: A voice-over-Internet protocol technique with a new hierarchical data security protection (HDSP) scheme using a secret chaotic bit sequence has been recently proposed. Some insecure properties of the HDSP scheme are pointed out and then used to develop known/chosen-plaintext attacks. The main findings are: given $n$ known plaintexts, about $(100 - (50/2^n))\%$ of secret chaotic bits can be uniquely determined; given only one specially-chosen plaintext, all secret chaotic bits can be uniquely derived; and the secret key can be derived with practically small computational complexity when only one plaintext is known (or chosen). These facts reveal that HDSP is very weak against known/chosen-plaintext attacks. Experiments are given to show the feasibility of the proposed attacks. It is also found that the security of HDSP against the brute-force attack is not practically strong. Some countermeasures are discussed for enhancing the security of HDSP and several basic principles are suggested for the design of a secure encryption scheme.

1 Introduction

With the rapid development of the Internet and digital communication technologies, it becomes possible to make a telephone call with a PC connected to the Internet, or to connect the link between two telephones partially via the Internet. This technique, called voice-over-Internet protocol (VoIP) [1], can be extended to provide many other services, such as fax-over-Internet protocol (FoIP) and video teleconferencing. Due to the obvious benefits and potential applications of VoIP technology, it attracts increasing interest from both vendors and consumers.

Differing from traditional telephony service, VoIP faces new risks in the networked world: since all data are transmitted over the Internet, any algorithm to attack digital computers can be used to break a VoIP system. Thus it is very important to provide sufficient security for VoIP services with a reasonable cost. In [2, 3] a new VoIP with a hierarchical data security protection (HDSP) scheme was proposed as a possible solution to the following two problems: reducing continuous packet loss to avoid large voice corruption, and encrypting the transmitted voice to provide high-level security. HDSP works under the control of a chaotic bit sequence, generated by iterating a discrete-time chaotic map whose initial condition and control parameter serve as the secret key. In [3] it was claimed that HDSP can resist the known-plaintext attack.

The present paper analyses the security of HDSP in detail. Due to some insecure properties of the HDSP scheme, the following problems are found: given $n$ known plaintexts, about $(100 - (50/2^n))\%$ of secret chaotic bits can be uniquely determined; given only one specially chosen plaintext, all secret chaotic bits can be uniquely derived; and the secret key can be derived with a practically small complexity when only one plaintext is known (or chosen). As a result, HDSP is very weak against known/chosen-plaintext attacks. In addition, it is found that the security of HDSP against the brute-force attack (i.e. an attack exhaustively searching the key) is not practically strong.

Some preliminaries on the modern cryptology are given to facilitate the following discussion and analyses. A brief introduction to HDSP is followed by a cryptanalysis of the scheme. Experimental results support theoretical results about proposed attacks. How to improve the security of HDSP is discussed and some basic principles for designing a good cipher is suggested.

2 Preliminaries of modern cryptology

To facilitate the following discussion on HDSP and the proposed attacks a brief introduction is given to the basic theory of modern cryptology [4]. Generally speaking, cryptology is the technology of encryption, comprising the following two parts: cryptography, which studies how to design good encryption algorithms; and cryptanalysis, which tries to find security weaknesses of proposed encryption algorithms and studies whether or not they are vulnerable to various attacks.

An encryption/decryption system is also called a ‘cipher’ or a ‘cryptosystem’. Accordingly the encryption machine is called an ‘encipher’ and the decryption machine is called a ‘decipher’. The message for encryption is called the ‘plaintext’ and the encrypted message is called the ‘ciphertext’. Assuming that the plaintext and ciphertext are denoted by $P$ and $C$, respectively, the encryption procedure in a cipher can be described as $C = E_K(P)$, where $K$ is the encryption key and $E(\cdot)$ is the encryption function. Similarly the
decryption procedure is \( P = D_K(C) \), where \( K_d \) is the decryption key and \( D(\cdot) \) is the decryption function. When \( K_s \neq K_d \) the cipher is called a ‘private-key cipher’ or a ‘symmetric cipher’. For private-key ciphers the encryption–decryption key must be transmitted from the sender to the receiver via a separate secret channel. When \( K_s \neq K_d \), the cipher is called a ‘public-key cipher’ or an ‘asymmetric cipher’. For public-key ciphers the encryption key \( K_e \) is published, and the decryption key \( K_d \) is kept private, for which no additional secret channel is needed for key transfer (Fig. 1).

Following the widely-acknowledged Kerckhoffs’ principle in cryptography [4], the security of a cipher should rely only on the decryption key \( K_d \), and it is assumed that all details of the encryption/decryption algorithms are known to the attackers. Thus the main task of cryptanalysis is to reconstruct the key or its equivalent form that can be used to successfully decrypt all or part of the plaintexts.

From a cryptographical point of view, a secure cipher should have a capability against all kinds of attacks. In most cases the designed cipher should be secure against a brute-force attack (i.e. the attack of exhaustively searching all possible keys) and the following four typical attacks:

- ciphertext-only attack: attackers can only observe some ciphertexts
- known-plaintext attack: attackers can get some plaintexts and the corresponding ciphertexts
- chosen-plaintext attack: attackers can choose some plaintexts and get the corresponding ciphertexts
- chosen-ciphertext attack: attackers can choose some ciphertexts and get the corresponding plaintexts

In the four kinds of attacks, a ciphertext-only attack is the easiest and the most common attack due to the fact that the communication channel is generally accessible to attackers. Known/chosen-plaintext attacks are possible when an attacker can temporarily access the encryption machine, or can successfully guess the plaintexts. A chosen-ciphertext attack is possible when an attacker can have temporary access to the decryption machine. The last three kinds of attacks, which seem seldom used in practice, are feasible in some real applications [4, Section 1.1] and have become increasingly common in the digital world today. For example, when VoIP software is configured to run with a previously-stored password, an attacker can successfully carry out a known/known-plaintext or chosen-ciphertext attack by hacking the computer, and can also do so easily when the authenticated user leaves the computer without logging out, e.g. when the attacker is a colleague in the same office. (Many programs support an autosaving function of the password for quick startup. Although the risk of such a function has been strongly criticised by many security experts, it is still widely used in today’s password-based software such as Microsoft IE, MSN, etc.) Generally speaking, if a cipher cannot resist a known/known-plaintext or chosen-ciphertext attack, it can hardly be accepted by the computer and network security community.

In the four kinds of attacks, a ciphertext-only attack is the easiest and the most common attack due to the fact that the communication channel is generally accessible to attackers. Known/known-plaintext attacks are possible when an attacker can temporarily access the encryption machine, or can successfully guess the plaintexts. A chosen-ciphertext attack is possible when an attacker can temporarily access the decryption machine, or can successfully guess the plaintexts. A chosen-plaintext attack is possible when an attacker can temporarily access the encryption machine, or can successfully guess the plaintexts. A known-plaintext attack is possible when an attacker can choose some plaintexts and get the corresponding ciphertexts. The chosen-plaintext or chosen-ciphertext attack is possible when an attacker can choose some plaintexts and get the corresponding ciphertexts. The chosen-ciphertext attack is possible when an attacker can temporarily access the decryption machine, or can successfully guess the plaintexts. A known-plaintext attack is possible when an attacker can choose some plaintexts and get the corresponding ciphertexts. The chosen-plaintext attack is possible when an attacker can temporarily access the encryption machine, or can successfully guess the plaintexts.

3 HDSP encryption scheme

In the HDSP-based VoIP system the encryption part is placed after the speech encoder, and the decryption part is placed before the speech decoder. So the plaintext of HDSP is the bit-stream encoded by the speech Codec, not the raw voice signal. The ciphertext is obtained by performing the following two steps on the plaintext [2, 3]:

**Step 1. Interframe interleaving (frame swapping):** divide the plaintext into \( S_f \)-byte frames and pseudorandomly permute the order of \( S_f \) continuous frames (i.e. every \( S_f \) frames compose an interleaving group);

**Step 2. Intraframe encryption (bit swapping and masking):** for each input byte pseudorandomly permute some bits and mask the four odd bits with XOR operations.

The frame/bit swapping and the bit-masking operations are all controlled by a secret chaotic bit sequence \( \{b(i)\} \), which is generated by iterating the chaotic logistic map [5]:

\[
\begin{align*}
    f(x) &= \mu x (1-x) \\
    x(0) &\text{ initial condition of the chaotic logistic map, which are both represented in the 16-bit binary form.}
\end{align*}
\]

**Initialisation procedure:** the logistic map from \( x(0) \) to generate a chaotic sequence

\[
\begin{align*}
    \{s(i)\}^{L_f-1}_{i=0} = x(0)
\end{align*}
\]

where \( L_f \) denotes the number of bits required during the following encryption procedure, and then extract the 16-bit representation of each chaotic state \( x(i) \) to obtain a pseudorandom bit sequence (PRBS) \( \{b(i)\}^{L_f-1}_{i=0} \). Such an initialisation procedure has been widely used in other chaos-based ciphers proposed by Yen (and Guo) et al. as a common method for generating pseudorandom bits; see Sec. 4.4.3 of [6] for a survey on other chaotic ciphers. The encryption procedure comprises the following stages:

(i) **Interframe interleaving**

divide \( g \) into \( S_f \)-byte frames: \( \{\text{frame}(i)\}^{N_f-1}_{i=0} \), where \( N_f = [N/S_f] \);

further divide \( g \) into \( S_g \)-frame groups: \( \{\text{group}(i)\}^{N_g-1}_{i=0} \), where \( N_g = [N/S_g] \);

set \( L = \log_2 S_g \) and \( \Delta_L = S_g - 2^L \);

all \( S_g \) frames in each group are permuted with \( S_g \) pseudorandom swapping operations: for \( i = 0 \ldots (N_g - 1) \) and for \( j = 0 \ldots (S_g - 1) \), swap \( \text{frame}(i \cdot S_g + j) \) and \( \text{frame}(i \cdot S_g + j') \), where

\[
    j' = \sum_{k=0}^{L-1} 2^k \times b(s + k) + \sum_{k=0}^{\Delta_L-1} b(s + L + k)
\]

and \( s = (i \cdot S_g + j) \cdot L \). In this stage, a total of \( (N_g \cdot S_f \cdot L + \Delta_L) \) chaotic bits are required: \( b(0), \ldots, b(N_g \cdot S_f \cdot L + \Delta_L - 1) \).

(ii) **Intraframe encryption**

assuming \( g^* = [g^*(i)]^{L_f-1}_{i=0} \) is the output plaintext of the

interframe interleaving stage, the ciphertext \( g' = [g'(i)]_{i=0}^{N-1} \) is determined in the following two steps:

- Pseudorandomly swap the four most-significant bits (MSBs) and the four least-significant bits (LSBs) of each byte \( g'(i) = \sum_{k=0}^{7} d_k(i) \cdot 2^k \) to obtain an intermediate byte

\[
g'^*(i) = \sum_{k=0}^{7} d_k'^*(i) \cdot 2^k : \quad \forall k = 0, 1, 2, 3,
\]

\[
(d_k'^*(i), d_{k+4}'(i)) = \text{Swap}_{w(k+4)}(d_k'(i), d_{k+4}'(i))
\]

where

\[
\text{Swap}_{w}(a, b) = \begin{cases} (a, b), & w = 0, \\ (b, a), & w = 1; \end{cases}
\]

- Mask the four odd bits of \( g'^*(i) \) to obtain the cipher-byte

\[
g'(i) = \sum_{k=0}^{7} d_k'(i) \cdot 2^k : \quad \forall k = 1, 3, 5, 7
\]

\[
d_k'(i) = d_k^*(i) \oplus b(4i + k)
\]

where \( \oplus \) denotes the XOR (exclusive OR) operation. In this stage a total of \((4N + 2)\) chaotic bits are required: \(b(0), \ldots, b(4N - 1), b(4N + 1), b(4N + 3)\).

In the whole encryption procedure a total of \(\max(N_g, S_g \cdot L + \Delta_g, 4N + 2)\) chaotic bits are required. Since the index of the last required chaotic bit is \(\max(N_g, S_g \cdot L + \Delta_g - 1), 4N + 3\), one has \(L_b = \max(N_g, S_g \cdot L + \Delta_g, 4N + 4)\).

- The decryption procedure is the reverse of the encryption procedure, briefly described as follows:

  (i) Inverse intraframe decryption (bit swapping and masking operations):

  mask the four odd bits of the cipher-byte \( g'(i) = \sum_{k=0}^{7} d_k(i) \cdot 2^k \) to restore the intermediate byte \( g'^*(i) \); pseudorandomly swap the four MSBs and the four LSBs of each byte \( g'^*(i) = \sum_{k=0}^{7} d_k'^*(i) \cdot 2^k \) to restore another intermediate byte \( g'(i) = \sum_{k=0}^{7} d_k(i) \cdot 2^k \).

  (ii) The same interframe interleaving is inversely exerted on \( g^* = [g^*(i)]_{i=0}^{N-1} \) to restore the plaintext \( g = [g(i)]_{i=0}^{N-1} \), where the term ‘inversely’ means that \( i = 0, \ldots, (N_g - 1) \) and \( j = (S_g - 1), \ldots, 0 \).

## 4 Cryptanalysis of HDSP

### 4.1 Brute-force attack

The brute-force attack is an attack of exhaustively searching the secret key from the set of all possible keys [4], which can be used in different attacking scenarios including ciphertext-only and known-plaintext attacks. Apparently the computational complexity of the brute-force attack is determined by the size of the key space and the complexity of verifying each key. The secret key in HDSP is \((\mu, x(0))\), which is represented by \(2 \times 16 = 32\) secret bits. Thus the size of the key space is \(2^{32}\). Since the complexity of verifying each key is equal to the one in the encryption procedure of HDSP \(- O(N) [3, \, Section \, 4.1] \), the total complexity of the brute-force attack is \(O(N \times 2^{32})\). However, because not all values of \(\mu\) can ensure the chaoticity of the logistic map [5], actually the size of the key space is smaller than \(2^{32}\) and the attack complexity is smaller than \(O(N \times 2^{32})\).

The following facts should be noted. In the known-plaintext attack scenario the key can be validated by simply comparing the decrypted signal with the known plaintext. In the ciphertext-only attack scenario the key has to be validated by some inherent features of the plaintexts, which may increase the attack complexity to some extent, but generally the complexity will not be larger than \(O(N^2 \times 2^{32})\).

To guarantee a high-level of security in today’s digital world a complexity of order \(O(2^{100})\) is required [4]. Apparently \(O(N \times 2^{12})\) is too small to reach such a goal. An attacker can easily find the secret key within a few hours (or even minutes) using a PC with a 1 GHz CPU, if \(N\) is not large enough (when \(N\) is too large it is easy for one to check only a small segment of the plaintext). To ensure high-level security, 64-bit binary representations of \(\mu\) and \(x(0)\) are suggested to provide a complexity of order \(O(2^{75})\) against a brute-force attack.

### 4.2 Known-plaintext attack

In [3, Section 4.2] it was claimed that HDSP can efficiently resist a known-plaintext attack. (A known-plaintext attack becomes possible when an attacker can temporarily access the encryption machine. Under such a condition, it is reasonable to assume that the attacker can get all plaintext outputs from the speech codec, since details of the codec are not kept secret in the HDSP-based VoIP system.) Here, this claim is re-evaluated, concluding that HDSP is not sufficiently secure against a known-plaintext attack. Basically, with \(n\) known plaintexts, from the probabilistic point of view, \((100 - (50/2^n))\%\) of chaotic bits can be correctly restored and only \(16/2^n\) bits need to be exhaustively guessed to break the secret key. (To break the secret key one needs to get 32 consecutive chaotic bits to recover two consecutive chaotic states, see Section 4.2.4. Since only \(50/2^n\)% of chaotic bits cannot be uniquely derived, one has to exhaustively search \(32 \cdot (50/2^n)\% = 16/2^n\) bits. This figure is meaningful only in a probabilistic sense. As discussed in Section 4.2.4, in real attacks it is possible to exhaustively search fewer bits.) Even with \(n = 1\), the complexity of breaking the secret key is sufficiently small: only \(O(2^7)\). Once the secret key is uncovered the HDSP scheme is completely broken. Furthermore, because HDSP works like a stream cipher, even without deriving the secret key, one can still use the partially reconstructed chaotic bit sequence to partially (almost completely when \(n\) is relatively large) recover the unknown plaintexts.

Similar to the encryption procedure of HDSP, the known-plaintext attack also works in two steps: break the interframe interleaving and break the intraframe encryption. The two steps provide a partial (or total) reconstruction of the secret chaotic bit sequence, which can then be used to restore the secret key. Generally, if \(N\) is sufficiently large only one known plaintext is enough for an attacker to restore the subkey \(\mu\) and an equivalent of another subkey \(x(0)\). In addition, if two or more plaintexts of size \(N\) are known, it is possible to correctly restore most chaotic bits, which can also be directly used as a replacement of the secret key to decrypt the ciphertexts encrypted with the same key (if their sizes are not larger than \(N\)).

Next, some important properties of the intraframe encryption step of HDSP are discussed, which are the
essential reasons for the insecurity of HDSP against known/ chosen-plaintext attacks. These insecure properties are structural defects of the encryption procedure used in HDSP, independent of any specifications of the speech data and the speech codec. The theoretical analysis of the proposed attacks will be based on the assumption that the plaintext (i.e. the output of the speech codec) is a uniformly-distributed random source. When the plaintext does not obey the uniform distribution, our experiments show that the performance of the proposed attacks may be somewhat better or worse (see Section 5).

4.2.1 Insecure properties of HDSP encryption scheme: In the intraframe encryption step of HDSP the
ith byte of the input signal \( g^i(t) = \sum_{k=0}^{2^n} d_k^i \cdot 2^k \) and
the ith byte of the output signal (i.e. the ciphertext) \( g^i(t) = \sum_{k=0}^{2^n} d_k^i \cdot 2^k \) satisfy the following properties.

Property 1: For \( k = 0, 2 \) (a) \( d_k^i + d_{k+4}^i = d_k^i + d_{k+4}^i \); (b) when \( d_k^i \neq d_{k+4}^i \), one has \( b(4i+k) = d_k^i \oplus d_k^i = d_k^i + d_{k+4}^i \).

Proof: From (2) and (3) one can see that the four even bits are swapped without being masked. Thus for \( k = 0, 2 \), \( d_k^i + d_{k+4}^i \) remains unchanged after the intraframe
encryption, i.e. \( d_k^i + d_{k+4}^i = d_k^i + d_{k+4}^i \). When \( d_k^i \neq d_{k+4}^i \), which means that \( d_k^i = d_{k+4}^i \) and
\( d_{k+4}^i = d_k^i \), the bit-swapping operation \( Swap_{a,b}(a, b) \) becomes (in this case, \( a \neq b \))

\[
\text{Swap}_{a,b}(a, b) = \begin{cases} 
(a, a) = (a \oplus 0, b \oplus 0) = (a \oplus w, b \oplus w), \\
(b, b) = (a \oplus 1, b \oplus 1) = (a \oplus w, b \oplus w). 
\end{cases}
\]

That is, \( \text{Swap}_{a,b}(a, b) = (a \oplus w, b \oplus w) \). Then one has
\( d_k^i = d_k^i \oplus b(4i+k) \) and \( d_{k+4}^i = d_{k+4}^i \oplus b(4i+4) \), which immediately leads to \( b(4i+k) = d_k^i \oplus d_k^i = d_k^i + d_{k+4}^i \). Thus the proof is complete. \( \square \)

Property 2: For \( k = 1, 3 \) (a) when \( d_k^i = d_{k+4}^i \), one has
\( b(4i+k) = d_k^i \oplus d_k^i \) and \( b(4i+1+k) = d_k^i \oplus d_k^i \). When \( d_k^i \neq d_{k+4}^i \), one has \( d_k^i = d_k^i \) and
\( b(4i+k) \oplus b(4i+1+k) = d_k^i + d_{k+4}^i \).

Proof: The two conditions are proved separately.

(a) When \( d_k^i = d_{k+4}^i \), the bit-swapping operation disappears and the cipher-bit is determined by the masking operation only: \( d_k^i = d_k^i \oplus b(4i+k) \) and \( d_{k+4}^i = d_{k+4}^i \oplus b(4i+k+4) \). That is, \( b(4i+k) = d_k^i \oplus d_k^i \) and \( b(4i+1+k) = d_k^i + d_{k+4}^i \).

(b) When \( d_k^i \neq d_{k+4}^i \), from (4), \( \text{Swap}_{a,b}(a, b) = (a \oplus w, b \oplus w) \), so one can get \( d_k^i = d_k^i \oplus b(4i+k) \) and
\( d_{k+4}^i = d_{k+4}^i \oplus b(4i+4) \). Substituting the two results into (3) one has the following results:

\[
\begin{align*}
&d_k^i = d_k^i \oplus b(4i+k) = (d_k^i \oplus b(4i+k)) \oplus b(4i+k) = d_k^i \\
&d_{k+4}^i = d_{k+4}^i \oplus b(4i+k+4) = (d_{k+4}^i \oplus b(4i+k)) \oplus b(4i+k+4) = d_k^i + d_{k+4}^i.
\end{align*}
\]

From these two conditions, the property is proved. \( \square \)

Property 3: For \( k = 1, 3 \), \( \forall i = 0, \ldots, N-2 \), if \( d_k^i = d_{k+4}^i \) and
\( d_k^i + 1 = d_{k+4}^i + 1 \), then \( b(4i+1+k) = d_k^i + d_{k+4}^i = d_k^i + d_{k+4}^i \).

Proof: This property is a direct corollary of property 2a.

In the following it is shown that the foregoing properties make the known-plaintext attack feasible in practice.

4.2.2 Breaking interface interleaving: The frame-swapping operations in the interface interleaving step actually correspond to a pseudorandom and secret frame-permutation in each group. One can represent the permutation of group(i) by a permutation vector \( \nu(i) = (\nu(i), \ldots, \nu(i, S_g-1)) \), where \( \nu_j \neq j \). With the permutation vector, the interface interleaving of group(i) can be described as follows: \( \forall j = 0, \ldots, (S_g-1) \), the jth frame in group(i) is permuted to be the \( \nu(j, j) \)th frame, i.e. \( \text{frame}(i, S_g+j) \) is permuted to be \( \text{frame}(i, S_g + \nu(i, j)) \). Basically, in this restoration stage the goal is to restore the permutation vectors of all groups: \( \nu(0), \ldots, \nu(N_g-1) \).

To restore the permutation vectors, at least an input signal (i.e. the plaintext \( g^* \)) and the corresponding output signal (i.e. \( g^* \)) should be known. However, in the known-plaintext attack, generally the intermediate signal \( g^* \) is not known. Fortunately, due to property 1a, some information of \( g^* \) can be obtained from the ciphertext \( g^* \). This generally is enough to restore the permutation vectors.

Next, define three sequences

\[
\hat{g} = [\hat{g}(i)]_{i=0}^{N-1}, \quad g^* = [g^*(i)]_{i=0}^{N-1}, \quad \text{and } \hat{g}^* = [\hat{g}^*(i)]_{i=0}^{N-1}
\]

where

\[
\begin{align*}
g^*(i) &= (d_0^i + d_4^i) + (d_2^i + d_6^i) \quad &\times 3 \in \{0, \ldots, 8\} \\
\hat{g}^*(i) &= (d_2^i + d_6^i) + (d_0^i + d_4^i) \quad &\times 3 \in \{0, \ldots, 8\} \\
\hat{g}(i) &= (d_0^i + d_4^i) + (d_2^i + d_6^i) \quad &\times 3 \in \{0, \ldots, 8\}
\end{align*}
\]

From property 1a, one can see that \( \hat{g}^* = \hat{g} \). Considering that the interface interleaving stage does not change the values of all frames but their positions, one can use \( \hat{g}^* \) and \( \hat{g} \) to restore the permutation vectors. To do so one has to divide both \( \hat{g} \) and \( \hat{g} \) into \( N_f \) frames \( \{\text{frame}(i)\}_{i=0}^{N_f-1} \), \( \{\text{frame}(i)\}_{i=0}^{N_f-1} \), and \( N_g \) groups \( \{\text{group}(i)\}_{i=0}^{N_g-1} \), \( \{\text{group}(i)\}_{i=0}^{N_g-1} \), respectively, in the same way as the encryption procedure of HDSP. Now, \( \forall i = 0, \ldots, (N_g-1) \), the permutation vector of group(i) can be estimated as follows (more general description of this algorithm for breaking secret permutations with a number of known/chosen plaintexts can be found in [7] where some common permutation-only image ciphers are analysed):

\* Step 1: For \( j = 0, \ldots, (S_g-1) \), calculate

\[
R_j(j, f) = \sum_{k=0}^{S_g-1} \hat{g}(i, j, k) \times 9^k
\]

and

\[
R_j(j, f) = \sum_{k=0}^{S_g-1} \hat{g}(i, j, k) \times 9^k
\]

where \( \hat{g}(i, j, k) \) and \( \hat{g}^*(i, j, k) \) denote the kth byte of the jth frame in group(i) and in group(i), respectively.
**Step 2:** Compare the values of $R_i(i, 0), \ldots, R_j(i, S_j - 1)$ and $R_m(i, 0), \ldots, R_m(i, S_m - 1)$ to get two partitions of the index-set $S_{ik} = \emptyset, \ldots, \emptyset, S_{ik}^1$: $[\Lambda(k)]_{k=0}^K$ and $[\Lambda(k)]_{k=0}^{K-1}$, where $K$ is the number of different values in the set $\{R_i(i, 0), \ldots, R_j(i, S_j - 1)\} = \{R_m(i, 0), \ldots, R_m(i, S_m - 1)\}$ and $\forall a, b \in \Lambda(k), \forall a', b' \in \Lambda(k), R_j(i, a) = R_j(i, b) = R_l(i, a') = R_l(i, b')$.

Note 1: $[\Lambda(k)]_{k=0}^K$ and $[\Lambda(k)]_{k=0}^{K-1}$ are partitions of $S_{ik}$ which means that $\bigcup_{k=0}^K \Lambda(k) = \bigcup_{k=0}^{K-1} \Lambda(k) = S_{ik}$ and $\forall k \neq k_2, \Lambda(k) \cap \Lambda(k_2) = \Lambda'(k_i) \cap \Lambda'(k_2) = \emptyset$.

Note 2: because the frame swapping operations do not change the value of $R_i(i, j)$, it is obvious that $\forall k = 0, \ldots, K-1$, the cardinality (size) of $\Lambda(k)$ is equal to that of $\Lambda' (k)$, i.e. $|\Lambda(k)| = |\Lambda'(k')|$.

**Step 3:** Derive an estimation of the permutation vector $\psi(i)$ of $\text{group}(i)$ from $[\Lambda(k)]_{k=0}^K$ and $[\Lambda(k)]_{k=0}^{K-1}$, under the following two conditions:

**Condition 1:** if $\forall k \in \{0, \ldots, K-1\}$, $\Lambda(k)$ contains only one element, i.e. $|\Lambda'(k)| = |\Lambda'(k)| = 1$ (and $K = S_j$), then the permutation vector $\psi(i)$ of group(i) can be uniquely derived: $\forall k = 0, \ldots, K-1$, $\Lambda(k)$ is set to the only element in $\Lambda(k)$.

**Condition 2:** if $\forall k \in \{0, \ldots, K-1\}$, $\Lambda(k)$ contains more than one element, i.e. $|\Lambda'(k)| = |\Lambda'(k)| = 2$ (and $K < S_j$), then the permutation vector $\psi(i)$ cannot be uniquely derived from $[\Lambda(k)]_{k=0}^{K-1}$ and $[\Lambda(k)]_{k=0}^{K-1}$. But one can get an estimated permutation vector $\tilde{\psi}(i) = [\psi(i, 0), \ldots, \psi(i, S_j - 1)]$ as follows: for $k = 0, \ldots, (K-1)$, determine a one-to-one mapping $f_{\psi(i)}: \Lambda(k) \rightarrow \Lambda(k)$, and then $\forall a \in \Lambda(k)$, set $\tilde{\psi}(i, a) = f_{\psi(i)}(a)$. Under condition 1 in step 3 the permutation vector can be correctly derived without any error. However, under condition 2, the estimated permutation vector $\tilde{\psi}$ may be wrong with a nonnegligible probability. This is due to the following fact: $\forall k = 0, \ldots, K-1$, there are $|\theta(\Lambda(k))|$ possible mappings of $f_{\psi(i)}$, so there are $|\prod_{k=0}^{K-1}\theta(\Lambda(k))|$ possible estimations of $\psi(i)$, among which only one is the correct permutation vector $\psi$.

Now study the occurrence probability of condition 2. Assuming the number of all possible values of $R_i(i, j)$ is $N_{R_{ij}}$, this probability can be easily calculated as follows:

$$\text{Prob}[\text{condition 2 occurs}] = 1 - \left(1 - \frac{N_{R_{ij}}}{N_{R_{ij}} - 1}\right) \cdots \left(1 - \frac{S_j - 1}{N_{R_{ij}} - 1}\right)$$

From the definition of $R_j$, one has $N_{R_j} = 9^S$. In most cases, $9^S$ is much larger than $S_j$, so the occurrence probability of condition 2 is so small that it can be simply ignored in practice. When condition 2 cannot be ignored (i.e. when $9^S$ is not much larger than $S_j$), the following constraints in the intraframe encryption stage can be used to detect wrong estimations:

- **Constraint 1:** $k = 1, 2, \forall i = 0, \ldots, (N-1)$, if $d(i) \neq d_{\psi(i)}(i), d_{\psi(i)}(i) = d(i)$.
- **Constraint 2:** $k = 1, 2, \forall i = 0, \ldots, (N-2)$, if $d(i) = d_{\psi(i)}(i)$ and $d_{\psi(i)}(i) = d_{\psi(i)}(i+1) + d_{\psi(i)}(i+1)$, then $d_{\psi(i)}(i) = d_{\psi(i)}(i+1) + d_{\psi(i)}(i+1) = b(4(i+1) + k)$.
- **Constraint 3:** As shown in the following subsubsection, the two chaotic bit sequences $\{b(4(i+1))\}_{i=0}^{N-1}$ and $\{b(4(i+3))\}_{i=0}^{N-1}$ are completely correlated, i.e. they satisfy (9). How to use this constraint to detect wrong permutation vectors is precisely explained there.

These three constraints can be deduced from properties 2 and 3. Once any of these constraints is violated, it can be immediately asserted that the current permutation vector is wrong.

Finally, the following fact should be noticed: in condition 2, the larger the number of the known plaintexts is, the less the probability of $\tilde{\psi}(i) \neq \psi(i)$ will be. Given $n$ known plaintexts $g_1, \ldots, g_n$, the number of all possible combinations of $R_j(i, j), \ldots, R_j(i, j)$ becomes $N_{R_j}^2 = 9^{2n}$, which means that the probability that condition 2 is satisfied exponentially decreases as $n$ increases. That is, the probability of getting a wrong permutation will exponentially decrease, so it is negligible when $n$ is sufficiently large (generally speaking, $n$ is sufficiently large if $n > (\log_9 S_j)/S_j$, which ensures that $9^{2n} < S_j$).

**4.2.2 Breaking intraframe encryption:** Once the interframe interleaving is correctly broken one can successfully get the intermediate signal $g^* = \{g^*(i)\}_{i=0}^{N-1}$ from the plaintext $g$. With $g^*$ and the ciphertext $g'$ the intraframe encryption can also be partially (or even totally) broken. During this breaking stage one can partially (or even totally) reconstruct the secret chaotic bit sequence, which can be used to further derive the secret key. The following properties are now available for an attacker to break the intraframe encryption:

- **Property 1b:** $k = 0, 2$, when $d(i) \neq d_{\psi(i)}(i), b(4(i + k)) = d_l(i) \oplus d_{\psi(i)}(i) = d_l(i) \oplus d_{\psi(i)}(i) = d_{\psi(i)}(i) \oplus d_{\psi(i)}(i)$
- **Property 2a:** $k = 1, 3$, when $d(i) = d_{\psi(i)}(i), b(4(i + k)) = d_l(i) \oplus d_{\psi(i)}(i) = d_{\psi(i)}(i) \oplus d_{\psi(i)}(i) = d_{\psi(i)}(i) \oplus d_{\psi(i)}(i)$
- **Property 2b:** $k = 1, 3$, when $d(i) \neq d_{\psi(i)}(i), b(4(i + 1) + k) = d_{\psi(i)}(i) \oplus d_{\psi(i)}(i)$

Based on these three properties one can estimate how many chaotic bits may be obtained in a known-plaintext attack. Without loss of generality, assume that each bit in $g^*(i)$ and $g'(i)$ distributes uniformly over $\{0, 1\}$ and any two bits are independent of each other. Under this assumption one can deduce the following results.

- **Even bits ($k = 0, 2$ from property 1b):**
  with only one known plaintext: the probability of $d(i) \neq d_{\psi(i)}(i)$ is $1/2$, so on average $50\%$ of all even chaotic bits can be correctly restored;
  with $n > 1$ known plaintexts: when $d(i) \neq d_{\psi(i)}(i)$ holds for at least one plaintext, the bit $b(4(i + k))$ can be correctly restored, and then it can be deduced that the probability of the event is $1 - (1/2)^n$.
- **Odd bits ($k = 1, 3$ from property 2a/b):**
  with only one known plaintext: $\forall i = 0, \ldots, (N-1)$, the value of $b(4(i + k)) \oplus b(4(i + 1) + k)$ can be correctly determined, i.e. one can get a new sequence $\{b^*(i) = b(4(i + k)) \oplus b(4(i + 1) + k)\}_{i=0}^{N-1}$. Apparently, if only one bit $b(4^*(i) + k)$ is known, the whole bit sequence $\{b(4(i + k))\}_{i=0}^{N-1}$ can be correctly restored with the deterministic sequence $\{b^*(i)\}_{i=0}^{N-1}$ as follows:

$$
\begin{align*}
\text{b(4^*(i) + k)} &\rightarrow \text{b(4(i + 1) + k)} \\
\text{b(4(i + 1) + k)} &\rightarrow \text{b(4(i + 2) + k)} \\
\text{b(4(i + 2) + k)} &\rightarrow \text{b(4(i + 3) + k)} \\
\vdots &\rightarrow \text{b(4(i + N) + k)}
\end{align*}
$$

From property 2a, two bits $b(4(i + k)$ and $b(4(i + 1) + k)$ can be correctly restored when $d(i) = d_{\psi(i)}(i)$. Thus it can be deduced that the probability that at least two
bits in $\{b(4i+k)\}_{i=0}^{N-1}$ are correctly restored is
\[
1 - (Prob[d_1^*(i) \neq d_2^*(i)])^N = 1 - \frac{1}{2^N}
\]
Since $N$ is generally sufficiently large, it is probabilistically true in almost all cases that all odd bits can be correctly restored.

Note 1: Even under an extreme condition where no intermediate byte $g^*(i)$ satisfying $d_1(i) = d_2(i)$, one can randomly guess the value of any bit $b(4i+k)$ and then get the whole bit sequence $\{b(4i+k)\}_{i=0}^{N-1}$. In this case, at most four guesses (two for $k=1$ and the other two for $k=3$) are needed to correctly restore all odd bits. In this sense, all odd bits can always be correctly reconstructed.

Note 2 (precise explanation of constraint 3): Assume that two bytes $g^*(i_1)$ and $g^*(i_2)$ satisfy $d_1(i_1) = d_2(i_1)$ and $d_1(i_2) = d_2(i_2)$ and all in-between bytes $g^*(i_1 + 1), \ldots, g^*(i_2 - 1)$ do not satisfy this condition, where $i_2 \geq i_1 + 2$. Then the subsequence $\{b(4i+k)\}_{i=i_1+1}^{i_2-1}$ can be uniquely derived by using (9) twice
\[
b(4i_1 + k) \rightarrow b(4(i_1 + 1) + k) \rightarrow \cdots \rightarrow b(4(i_2 - 1) + k) \rightarrow b(4i_2 + k)
\]
If the two derived subsequences are not identical it can be asserted that at least one permutation of the frames between $g(i_1)$ and $g(i_2)$ is wrong, i.e. at least one permutation vector of the groups between $g(i_1)$ and $g(i_2)$ is wrong, and when there is only one group between $g(i_1)$ and $g(i_2)$, the permutation vector of this group must be wrong.

with $n > 1$ known plaintexts: the probability that at least two bits in $\{b(4i+k)\}_{i=0}^{N-1}$ can be correctly restored is
\[
1 - (1 - (1/2^N))^n
\]
In summary, when only one plaintext is known, an average of 50% of even bits and all odd bits, i.e. 75% of all bits in $\{b(i)\}_{i=0}^{4N-1}$ and the last two bits $b(4N+1), b(4N+3)$, can be correctly restored. Furthermore, when $n \geq 1$ plaintexts are known, the percentage of correctly restored bits in $\{b(i)\}_{i=0}^{4N-1}$ becomes
\[
\left(\frac{50 + 25 + \frac{25}{2} + \cdots + \frac{25}{2^{N-1}}}{2^n}\right) \%
\]
which exponentially approaches 100% as $n$ increases. For the rest of the unrecovered chaotic bits, one can randomly guess their values, and on average half of the bits can be correctly matched to the true values. That is, the correctly restored bits in $\{b(i)\}_{i=0}^{4N-1}$ will reach $(100 - (50/2^{n+1}))\%$, which is 87.5% for $n = 1$.

If some permutation vectors have been uniquely determined in the stage of breaking interface interleaving, the corresponding guessed bits can be checked with (1) and the correctly restored bits may be even greater. In addition, since the wrong bits distribute randomly within the whole bit sequence and since human ears have a high capability to bear large audio noise, their negative influence on the quality of the voice data may not be so great, as verified by our experiments discussed later.

Finally, let us study the success probability of the decryption with the partially reconstructed chaotic bit sequence $b(0), \ldots, b(4N-1), b(4N+1), b(4N+3)$, when only one plaintext is known (i.e. $n = 1$). Since the two even bits $b(4i), b(4i+2)$ corresponding to a plain-byte $g(i)$ are correctly restored with probability $(1/2)^2 = 1/4$, it seems that the probability of correctly decrypting an unknown plain-byte $\tilde{g}(i) = \sum_{i=0}^{N-1} d(i) \cdot 2^i$ should also be 1/4. However, such an intuition is not true, because this probability is actually the addition of the following four probabilities:

- the probability that $b(4i), b(4i+2)$ are both correct, which is 1/4
- the probability that $b(4i)$ is correct, but $b(4i+2)$ is incorrect, and $d_2^*(i) = d_1^*(i)$, which is
  \[
  \frac{1}{2} \cdot \frac{1}{2} \cdot \frac{1}{2} = \frac{1}{8}
  \]
- the probability that $b(4i)$ is incorrect, but $b(4i+2)$ is correct, and $d_2^*(i) = d_2^*(i)$, which is also
  \[
  \frac{1}{2} \cdot \frac{1}{2} = \frac{1}{4}
  \]
- the probability that $b(4i), b(4i+2)$ are both incorrect, $d_2^*(i) = d_2^*(i)$ and $d_2^*(i) = d_2^*(i)$, which is
  \[
  \frac{1}{2} \cdot \frac{1}{2} = \frac{1}{4}
  \]
So the final probability of correctly decrypting a plain-byte is
\[
\frac{1}{4} + \frac{1}{8} + \frac{1}{8} = \frac{9}{16}
\]
In a similar way, this probability with $n \geq 1$ known plaintexts can be calculated as
\[
P(n) = \left(1 - \frac{1}{2^n}\right)^2 + \left(1 - \frac{1}{2^n}\right) \cdot \frac{1}{2^n} \cdot \frac{1}{2^n} \cdot \left(1 - \frac{1}{2^n}\right)
\]
\[
\quad \cdot \frac{1}{2^n} + \frac{1}{2^n} \cdot \frac{1}{2^n} \cdot \frac{1}{2^n} \cdot \frac{1}{2^n}
\]
\[
= 1 - \frac{1}{2^n} + \frac{1}{2^n + 2}
\]
\[
= 1 - \frac{1}{2^n + 1} + \left(\frac{1}{2^n + 1}\right)^2
\]
\[
= \left(1 - \frac{1}{2^n + 1}\right)^2
\]
Since each undetermined chaotic bit is identical with the original bit with probability 1/2 one has
\[
Prob[b(4i) \text{ is incorrect}] = Prob[b(4i+2) \text{ is incorrect}]
\]
\[
= \frac{1}{2} \cdot \frac{1}{2} = \frac{1}{2^n + 1}
\]
\[
Prob[b(4i) \text{ is correct}] = Prob[b(4i+2) \text{ is correct}]
\]
\[
= 1 - \frac{1}{2^n + 1}
\]
So the probability of correctly decrypting a plain-byte in real attacks will become
\[
P(n) = P(n+1) = \left(1 - \frac{1}{2^n + 2}\right)^2
\]
This result agrees well with our experiments (see Table 2 below).
4.2.4 Completely breaking the encryption scheme (breaking the secret key): With the reconstructed permutation vectors $\pi(0), \ldots, \pi(N_b - 1)$ and the partially restored chaotic bit sequence $b(0), \ldots, b(4N - 1), b(4N + 1), b(4N + 3)$, a ciphertext $g'$ can be decrypted to get an estimated plaintext $\hat{g} = [\hat{g}(i)]_{i=0}^{N_b - 1}$ as follows:

- use the partially reconstructed chaotic bits to cancel the intraframe encryption;
- derive the inverse permutation vector of each interleaving group and use it to cancel the interframe interleaving.

In the previous subsection it was shown that the breaking performance is not satisfactory because about 25% of plain-bytes cannot be correctly decrypted when only one plaintext is known. Although the performance can be exponentially enhanced with more known plaintexts, the procedure cannot decrypt any plain-byte beyond the position $N$. Therefore to thoroughly break the HDSP encryption scheme one has to break the secret key itself. In the following we show how to break the secret key from some consequent partially-reconstructed chaotic bits.

(a) Breaking initial condition $x(0)$ or an equivalent $x(i)$:

Recalling the generation of chaotic bits in the initialisation procedure of HDSP, one can see that each chaotic state $x(i)$ can be represented in the binary form as $0.b(16i + 0) \ldots b(16i + 15)$. Thus if the first 16 chaotic bits are all correctly restored, one can directly get the initial condition $x(0) = b(0)0 \ldots b(15)$. However, following the discussion in the previous subsection, generally not all even bits in $b(0), \ldots, b(15)$ can be uniquely determined. To restore these bits one has to exhaustively guess their values. Assuming the number of undetermined bits is $m \in [0, \ldots, 8]$, the guessing complexity is $O(2^m)$. On average, $m = (1/4) \times 16 = 4$, and the searching complexity is $O(2^4)$, which is practically small for PCs. Even under the worst condition, $m = 8$, the guessing complexity is only $O(2^8)$.

When $x(0)$ is enhanced to be represented with $B > 16$ bits and all the $B \leq B$ bits are extracted to generate the chaotic bit sequence $\{b(i)\}$, the guessing complexity under the worst condition will be $O\left(2^{B/2} \cdot 2^{B/2} = O\left(2^{B-B/2}\right)\right)$. If $(B - (B/2))$ is sufficiently large (for example, when $B = 80, B' = 16$, one has $B - (B/2) = 72$, which can be considered to be sufficiently large), the guessing complexity will be too large for the exhaustive guess of $x(0)$ on PCs. In this case, instead of guessing $x(0)$, one can try to find another chaotic state $x(i) = b(i \cdot B') \ldots b(i \cdot B' + (B' - 1))$ as an equivalent of the subkey $\{b(i)\}$, where the number of undetermined bits is less than $m < (B/4)$, to get a reduced guessing complexity not greater than $O\left(2^{B-B/4+m} < O\left(2^{B-(3B/4)}\right)\right)$. With $x(i)$ and the derived $\mu$ (see later), one can exactly reconstruct all chaotic bits beyond the position $(i \cdot B')$ and then restore all plain-bytes starting from the first group located after the plain-byte that is encrypted by the $(i \cdot B')$th chaotic bit. Apparently, for the original HDSP with $B = B' = 16$, such an idea of reducing the complexity is also feasible (but somewhat meaningless). Of course, using this method the guessing complexity has a lower bound $O\left(2^{B-B'}\right)$, which is the complexity of exhaustively guessing the $(B' - B')$ bits that do not occur in the chaotic bit sequence.

(b) Breaking control parameter $\mu$:

Given 32 consequent chaotic bits $b(16i + 0), \ldots, b(16i + 31)$, two consecutive chaotic states can be determined: $x(i) = 0.b(16i) \ldots b(16i + 15)$ and $x(i + 1) = 0.b(16i + 16) \ldots b(16i + 31)$, and then an estimated value of the secret subkey $\mu$ can be derived as

$$\hat{\mu} = \frac{x(i + 1)}{x(i) \cdot (1 - x(i))}$$

Due to the quantisation errors existing in the finite-precision environment, generally $\hat{\mu} \neq \mu$. Fortunately, following the error analysis of $\hat{\mu}$ given in Section 3.2 of [8], it has been shown that when $x(i + 1) \geq 2^{-n}$ ($n = 1, \ldots, 16$), one has $|\hat{\mu} - \mu| \leq 2^{n+1} \times 2^{-16}$. For example, when $x(i + 1) \geq 2^{-1} = 0.5$, one can exhaustively search $2^{16}$ = 65536 values in the neighbourhood of $\hat{\mu}$ to find the right value of $\mu$. All correctly-restored chaotic bits after $b(16i + 31)$ can be used to verify whether or not a searched value of $\mu$ is right.

As discussed, generally there are undetermined bits in $b(16i + 0), \ldots, b(16i + 31)$. The average number of such bits is $1/4 \times 32 = 8$, and the maximal number is 16. This means that one has to exhaustively search all values of the undetermined bits to calculate a number of different values of $\hat{\mu}$. One can see that the complexity will be about $O(2^8)$ on average and $O(2^{16})$ in the worst condition. To further reduce the searching complexity, one can try to find two consecutive chaotic states that contain less than $m < 8$ undetermined bits. (Similar to the condition of $x(0)$, when $B > 16$ bits are used to represent chaotic states, this becomes very useful to reduce the total complexity.)

The occurrence probability of such an event is

$$p(m) = \sum_{i=0}^{m} \binom{16}{i} \left(\frac{1}{2}\right)^{16}$$

and the average position of its first occurrence is

$$\frac{1}{p(m)} = \frac{2^{16}}{\sum_{i=0}^{m} \binom{16}{i}}$$

In Table 1 the values of $p(m)$ and $1/p(m)$ for $m = 0, \ldots, 8$ are shown. If the plaintext does not have a uniform distribution, the probability of $d_i(4) \neq d_{i+1}(4)$ may not be 1/2, and so the probability $p(m)$ may be different from the theoretical value (compare Tables 1 and 3).

<table>
<thead>
<tr>
<th>$m$</th>
<th>0</th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>5</th>
<th>6</th>
<th>7</th>
<th>8</th>
</tr>
</thead>
<tbody>
<tr>
<td>$p(m)$ ≈</td>
<td>0.0000153</td>
<td>0.000259</td>
<td>0.00209</td>
<td>0.0106</td>
<td>0.0384</td>
<td>0.105</td>
<td>0.227</td>
<td>0.402</td>
<td>0.598</td>
</tr>
<tr>
<td>$[1/p(m)]$</td>
<td>66536</td>
<td>3856</td>
<td>479</td>
<td>95</td>
<td>27</td>
<td>10</td>
<td>5</td>
<td>3</td>
<td>2</td>
</tr>
</tbody>
</table>

Table 1: Occurrence probability of 32 consequent bits with less than $m \leq 8$ undetermined bits, and average position of first occurrence of such bits in chaotic bit sequence.
attack, one can create a plaintext $g = \{g(i)\}_{i=0}^{N-1}$ as follows: $\forall i = 0, \ldots, (N-1)$, $d(i) \neq d(i)$ and $d(i) \neq d(i)$. With such a plaintext and its ciphertext $g'$, all chaotic bits can be uniquely determined. Thus the secret subkey $x(0) = 0.5(0) \ldots b(15)$ will be accurate, and another subkey $\mu$ can be exactly derived from any two consecutive chaotic states $x(i)$ and $x(i+1)$. The complexity of deriving $\mu$ is the complexity of searching over the neighbourhood of $\mu = (x(i+1))^{-1}/x(i)$ as the first chaotic state that is not less than 0.5, the search complexity will be minimised. As a result, one can see that HDSP is not secure at all against the chosen-plaintext attack.

5 Experiments

Some experiments are shown to support the theoretical analysis on the known-plaintext attack. (The chosen-plaintext attack is omitted here since it is just a special case of the known-plaintext attack.) The parameters used in the experiments are $N = 65536$, $S_r = 32$, $S_s = 16$, and the secret key is $x(0) = 16326/2^{16} \approx 0.249$, $\mu = 259752/2^{16} \approx 3.96$. Note that the values of $x(0)$ and $\mu$ are both randomly generated via the standard rand( ) function, not specially chosen to optimise the proposed attacks. The eight plaintexts involved are shown in Fig. 2, from top to bottom, denoted by $g_0$–$g_7$, respectively, where the first seven are candidates of known plaintexts and the last one is used to show the breaking performance. The corresponding ciphertexts of the eight plaintexts are denoted by $g_0$–$g_7$, which are not shown here since all of them are like meaningless noisy signals.

To simplify the experiments, we remove the speech codec in the whole HDSP-based VoIP system and directly use the uncompressed raw data as the plaintext. Such a simplification does not have any influence on the breaking performance of the secret key from the recovered chaotic bit sequence. For the known-plaintext attack, when the partially-recovered chaotic bit sequence is directly used to decrypt a ciphertext, the existence of speech codec may enlarge the recovery errors in the decoded voice signal. If such an enlargement is so serious that the intelligibility of the recovered voice signal is damaged, one can turn to derive the secret key, which always works as a perfect tool to break HDSP.

5.1 Partially reconstructing chaotic bit sequence

Following the breaking procedure discussed in the previous Section, when plaintexts $g_0, \ldots, g_{n-1}$ and their ciphertexts $g_0, \ldots, g_{n-1}$ are known (for $n = 1 \ldots 7$), we test the partial reconstruction of the chaotic bit sequence and the breaking performance when it is directly used to decrypt the ciphertext $g_7$. The percentage of undetermined chaotic bits for different values of $n$ is shown in Table 2, from which one can see that the percentage of the undetermined bits and the percentage of the undetermined bytes are both close to the theoretical predictions:

$$1 - P'(n) = 1 - \left(1 - \frac{1}{2^n+2}\right)^2$$

respectively.

Table 2: Percentage of undetermined bits in partially-reconstructed chaotic bit sequence $P_{\text{er}}$, and percentage of plaintexts of $g$ not correctly decrypted $P_{\text{er}}$ when $n = 1 \ldots 7$ plaintexts are known

<table>
<thead>
<tr>
<th>$n$</th>
<th>$1$</th>
<th>$2$</th>
<th>$3$</th>
<th>$4$</th>
<th>$5$</th>
<th>$6$</th>
<th>$7$</th>
</tr>
</thead>
<tbody>
<tr>
<td>$P_{\text{er}}$</td>
<td>26.4%</td>
<td>13.8%</td>
<td>7.23%</td>
<td>4.23%</td>
<td>2.28%</td>
<td>1.17%</td>
<td>0.640%</td>
</tr>
<tr>
<td>$1/2^n$</td>
<td>25.0%</td>
<td>12.5%</td>
<td>6.25%</td>
<td>3.13%</td>
<td>1.56%</td>
<td>0.781%</td>
<td>0.391%</td>
</tr>
<tr>
<td>$P_{\text{er}}$</td>
<td>24.5%</td>
<td>13.1%</td>
<td>6.90%</td>
<td>3.92%</td>
<td>2.05%</td>
<td>1.06%</td>
<td>0.591%</td>
</tr>
</tbody>
</table>

$$1 - P'(n) = 1 - \left(1 - \frac{1}{2^n+2}\right)^2 \approx 23.4% \quad 12.1% \quad 6.15% \quad 3.10% \quad 1.55% \quad 0.780% \quad 0.390%$$
texts are known

5.2 Breaking the secret key

As analysed in the foregoing, generally it is possible to derive the secret key \( x(0) \) (or its equivalent \( x(i) \)) and \( \mu \) from the partially-reconstructed chaotic bit sequence. To do so, one needs to find 32 consequent chaotic bits in which less than \( m \) bits are undetermined. When only the plaintext \( g_0 \) is known, for different values of \( m \), the numbers of all 32-bit groups satisfying the requirement are listed in Table 3. One can see that even for \( m = 0 \) there are enough positions to derive the secret key. By taking the first occurrence of the 32 consequent bits to successfully derive a chaotic state \( x(i) \) and the value of \( \mu \), one can decrypt any ciphertext from the first group after \( x(i) \), which begins at the position \( (4 \cdot i)/(S_g \cdot S_f) \cdot (S_g \cdot S_f) \). When \( g_0 \) is known, it was found that the first 32 consequent bits satisfying \( m = 0 \) occurs at \( x(163) \), which is used to derive \( \mu \) and then to decrypt \( g_7 \). The decrypted plaintext and the recovery error are shown in Fig. 6. It can be seen that all plain-bytes from \( g[(4 \cdot 163)/(16 \cdot 32)] \times (16 \times 32) = g(1024) \) are exactly recovered.

6 Improving HDSP

As shown in the previous Section, the insecurity of HDSP against known/chosen-plaintext attacks is attributed to the properties proved in subsection 4.2.1. In the first stage of breaking the interframe interleaving, properties 1a, 2a, 2b and 3 are involved; in the second stage of breaking the intra-frame encryption, properties 1b, 2a and 2b are involved. Also, the second breaking stage relies on the first one, since the intermediate signal \( g^* \) will not be available if the first stage does not work. This implies that the insecure properties play different roles in the known/chosen-plaintext attacks, which can be shown as follows:

\[
\begin{align*}
\text{g} & \xrightarrow{\text{property 1a}} g^* \\
& \xrightarrow{(\text{property 2a, 2b and 3})} g' \\
& \xrightarrow{\text{property 1b, 2a, and 2b}} \{b(i)\} \\
& \rightarrow \{x(0), x(i), x(i+1) \rightarrow \mu\}
\end{align*}
\]

One can see that property 1a is the basis of the whole attack, since properties 2a, 2b and 3 are just used to detect wrong permutation vectors. As a hint, if HDSP is modified to eliminate property 1a, the security against known/chosen-plaintext attack will be enhanced. However, if \( S_g \) is too small, it

### Table 3: Number of 32 continuous chaotic bits that have \( m \leq 8 \) undetermined bits, and occurrence frequency

<table>
<thead>
<tr>
<th>( m )</th>
<th>0</th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>5</th>
<th>6</th>
<th>7</th>
<th>8</th>
</tr>
</thead>
<tbody>
<tr>
<td>( N(m) )</td>
<td>72</td>
<td>185</td>
<td>464</td>
<td>888</td>
<td>1574</td>
<td>2537</td>
<td>4106</td>
<td>6113</td>
<td>8715</td>
</tr>
<tr>
<td>( \text{Freq}[N(m)] % )</td>
<td>0.0045</td>
<td>0.0113</td>
<td>0.0283</td>
<td>0.0542</td>
<td>0.0961</td>
<td>0.155</td>
<td>0.251</td>
<td>0.373</td>
<td>0.532</td>
</tr>
</tbody>
</table>
may be possible for an attacker to exhaustively search all ($S_k^!$) possible permutation vectors so as to pass the first breaking stage. Therefore, from the cryptographical point of view, all insecure properties of HDSP should be avoided to provide a high level of security. In addition, to resist other potential attacks, all known security defects should also be removed. In the following, we discuss how to amend the original HDSP scheme for better encryption.

- **Property 1a** is caused by the fact that no even bits are masked pseudorandomly by the secret chaotic bits. Also, property 1b is related to this defect. To fix this defect, we suggest masking all bits including odd bits and even bits.
- **Properties 1b, 2a and 3** are caused by the incapability of swapping operations for encryption purpose, and properties 2a and 3 are also partially caused by the invertibility of the XOR operation. These properties can be destroyed by changing the bit swapping and masking operations to other more complicated ones, for example, inserting an extra masking operation before the two bits are swapped, or changing bit swapping operation to a different bit function.
- **Property 2b** is caused by two flaws: (a) the equality of the masking operation and the swapping operation $\text{Swap}_1(a, b)$ when $a \neq b$ (see (4)); (b) the reuse of all odd bits: for $k = 1, 3, \forall i = 1 - N, \ldots, 1$, each bit $b(4i + k)$ are used thrice, once for the swapping operation of $d_l^k(i)$, and twice for the masking operations of $d_{l+1}^k(i) - 1$ and $d_l^k(i)$; similarly when $i = 0$ the bit $b(k)$ is used twice for the swapping and masking operations of $d_l^k(0)$. These two flaws disable the encryption for some bits and make the two chaotic bit sequences $\{b(4i + 1)\}_{i=0}^N$ and $\{b(4i + 3)\}_{i=0}^N$ totally correlated. The first flaw can be fixed with the same method for eliminating property 2a, and the second one can be removed by avoiding any reuse of chaotic bits, which means that at least $4 + 8 = 12$ chaotic bits are required for the encryption of each plain-byte (if swapping and masking operations are not replaced by other functions).

From this discussion one arrives at some principles for the design of a good encryption algorithm:

- Never repeatedly use any (secret) intermediate variables generated in the encryption procedure, such as the chaotic bits in the HDSP encryption scheme.
- Carefully avoid the occurrence of strong correlation between the plaintexts or some intermediate variables observable for attackers and the ciphertexts, and strong correlation between different secret parameters.
- Avoid the possibility of deriving the secret key under known/chosen-plaintext attacks. For HDSP, it is to avoid the possibility of deriving any chaotic states.

- Do not combine too-simple invertible functions to realise the encryption function. Instead, use good encryption functions defined in different groups to make their product irreversible, such as those used in IDEA [4].

7 Conclusions

The security of a recently-proposed encryption technique for VoIP, called HDSP [2, 3], has been analysed in detail. It has been found that HDSP cannot resist known/chosen-plaintext attacks, and that only one known/chosen plaintext is enough to break the secret key. It has also been found that the security of HDSP against the brute-force attack is very weak even for PCs. Both theoretical and experimental analyses have been given to support the feasibility of the proposed attacks. In conclusion, HDSP is not suggested for security-sensitive applications, particularly if the secret key may be reused to encrypt more than one plaintext (which is the scenario where known/chosen-plaintext attacks work very well [4]). Finally, some remedies have been suggested to improve the security of HDSP, along with some general guidelines for designing a secure encryption algorithm.

8 Acknowledgments

This research was partially supported by the National Natural Science Foundation of China under grant 60202002, by the Applied R&D Centres of the City University of Hong Kong under grants 9410011 and 9620004, and by the Hong Kong Polytechnic University’s Postdoctoral Fellowship Scheme under grant G-YX63.

9 References