TSG-SA WG3 (Security) meeting #18 **_S3-010XXX_**

Phoenix, AZ 21th May – 24th May 2001

Agenda Item: 9.5

Source: Lucent

Title: Hybrid sync-frame/sync-free E2E Encryption

Document for: Discussion

1 Scope

2 Introduction

At the February SA3 meeting #17, Lucent introduced the concept of non-synchronized voice encryption [1] for which we will use the shorthand term “sync-free”. The sync-free architecture needed a block encryptor with a short and variable block length. Unfortunately, well-known and extensively cryptanalyzed ciphers cannot meet this need. As noted at the meeting, a Feistel network (e.g. Luby-Rackoff construction) could meet this need. Candidates for the round functions of this construct might be Rijndael or Kasumi. However, there would be a significant processing overhead due to the number of round functions needed to secure a 24-hour call due to the short block sizes.

The number of rounds and thus processing overhead can be reduced sharply by using a hybrid architecture to emulate, cryptographically speaking, a call of a few seconds duration. Specifically, the hybrid would comprise the sync-free architecture and a modification of the Tetra synchronization method [2][1]. For the higher AMR modes, the processing overhead due to the sync-free component would be small. As will be described, a two-way synergy would occur: sharply reduced block cipher processing, and an amelioration of speech degradation from what might occur in a Tetra-like approach alone.

Like its components, this hybrid approach is compatible with an option to put E2E encryption and key management in the application layer. The final part of this document will describe this option and mention how it can potentially both speed development and allow E2E encryption between different entities. This option is just that; it is not meant to ultimately supplant a more system-dependent method with faster (intra-system) call set-up time.

3 Sync Frames

The Tetra system (Terrestrial Trunked Radio) achieves synchronization, in-band, by stealing frames at rates between 1 and 4 Hz and inserting synchronization information into these stolen frames. The AMR codec can use a similar in-band method with a somewhat less frequent cryptosync. Frame sync counters adjacent to the transmitting and receiving codecs increment every frame and are kept in identical states by periodically replacing frames with a special sync frame. The sync frames contain the transmitting frame sync counter’s LSBs. When these frames are received, the receiving frame sync counter’s LSBs are set equal to those of the sync frames. To roughly gauge how frequently frames can be stolen for AMR sync information, we first postulate the method defined in the next section. What follows is an example implementation. A final design

[1] The study group proposed the Tetra synchronization method as a standalone cryptosync alternative.

could use different values or be architected differently. (For the remainder of this document, *GERAN-specific* data are in *italics*.)

3.1 CRC protection of sync bits

The sync bits need to be heavily protected because an error in sync yields loud noise for the duration of the asynchronous condition. The effect of such an asynchronous condition can be heard by listening to the embedded file

for a PC, or syncLossPCM.au for a Sun workstation. This file depicts a condition where the ciphers at the two ends drift in and out of synchronicity every couple of seconds.

The best protected bits are called the Class A bits (*Class 1a bits*) in UMTS (*GERAN*) since these are undergo strong error correction and additionally are error-detected by an 8-bit (*6-bit*) CRC. Also, SID (Silence Descriptor) frames in UMTS's Source Controlled Rate (SCR) operation are placed in the Class A slot for those conversations that are SCR-enabled.

Sync frames need to replace both speech frames and silence frames (if present). A given conversation may not be SCR-enabled so that only speech frames are available. Alternatively, in a SCR-enabled call, there may be prolonged periods of silence during which it would be important to maintain synchronization.

There are two types of silence frames: SID_FIRST and SID_UPDATE. The SID_FIRST frame is meant to indicate the start of a silence period and can carry no further information. Thus it cannot be used for sync. The SID_UPDATE frame carries 35 "comfort noise" bits which can be substituted with sync. These frames occur every 8th frame during silence periods. The remaining 7 frames are not transmitted. This saves mobile power and reduces the net interference in the channel.

The 35 bits in SID_UPDATE frames happen to comprise the minimum size Class A (*Class 1a*) block. To simplify the design, we will assume that all sync frames contain 35 bits of sync information, regardless of whether they are SID_UPDATE frames or speech frames. In UMTS (*GERAN*), the sync information will fill the lesser significant bits of the Class A bits (*Class 1a bits*).

These sync frames will be sent periodically, or nearly periodically in the case of SID_UPDATE frames. Nominally, the period might be about 10 seconds. Periodicity can assist the receiving detector in locating a sync frame.

3.2 Sync frame structure

Next, partition the 35 bits into a 27-bit fixed pattern "header" and an 8-bit cryptosync field[2]. The pattern should be random looking. Moreover, a random-looking pattern should be XORed with the 8-bit cryptosync field. This randomization will minimize the possibility of a potentially probable frame of all zeroes or ones mimicking a sync frame.

Assume that the frame sync counters are synchronized at the start of a call, and that they flywheel after that. During the call, every frame is examined by the receiving sync detector which looks for the 24-bit pattern. It additionally checks to make sure that the cryptosync field is within ±64 frames (7 bits range) of where it should be. (The channel should never experience a delay hit of more than ±1 second and typically will experience hits within a range of ±3 frames.) When this pattern is detected, the receiving frame sync counter is corrected accordingly.

A false detection will occur every 2^{28} frames (28 = 35 – 7). A caller will experience this once in 2 months of continuous calling time, or practically speaking, once every few years. However, the two ends would again be synchronized when the next (real) sync frame was detected within 10 seconds. So false detection does not appear to be an issue.

[2] A final design may partition the 35 bits differently. In particular, it may apportion more bits for the cryptosync field if the frame sync counter initialisation is felt to be non-robust.

3.3 Error Concealment

Using the Class A (*Class 1a*) block or SID block yields another advantage. When the receiving detector indicates a sync frame, it sets the BFI flag to "1" which then causes error concealment. This reduces the amount of perceptual degradation due to the sync frame replacement.

3.4 Frequency of Sync Frames

Earlier, we said that sync frames might be sent about every 10 seconds. Motivation for this number is as follows. First, given a sync hit, it is desirable to synchronize as frequently as possible to minimize the out-of-sync duration. So the limit on how frequently depends on when the sync frames begin to perceptually degrade the speech. There are three sources of data to give a rough upper bound on the period: PESQ (Perceptual Evaluation of Speech Quality) testing that was performed with channel noise, and the effect of FACCH frames on North American TDMA-136 systems. Both show that degradation is perceived (0.15 MOS point) in the 12.2 kbps mode when the average period drops to about 4 seconds. So as not to significantly add to the degradation already induced by FACCH-like messages and channel noise, it might be appropriate to relax the sync period to 10 seconds.

In contrast to channel noise and FACCH messages, sync frames as defined here will not err the Class B (*Class 1b*) and Class C (*Class 2*) bits. Thus it may be that sync frames can be sent more frequently than every 10 seconds. Whether or not they can be sent as frequently as several times a second, as in Tetra, needs to be answered by further testing. Sending them less frequently than the Tetra system may prove objectionable to the caller due to noise bursts when synchronization errs. Upping the period to more than 5 seconds would tend to cause the customer to terminate the call.

4 Hybrid Architecture

If the frame sync period is large enough to cause objectionable out-of-sync noise, a hybrid design can sharply reduce the net speech degradation. As an example, assume that the sync period is 10 seconds. With a sync frame-only method of cryptosync, the customer can experience up to 10 seconds of loud noise when sync is lost. However, a hybrid design would augment the sync frame architecture with the architecture described in [1]. The frame sync counters would still increment every frame as described above. However, the cryptosync presented to the cipher would remain constant for the 10-second period between sync frames. To achieve a 10 second period, this would be implemented by discarding the 9 LSBs of the frame sync counter before being input to the cipher. If a new sync frame were not received within this period, the frame sync counter would continue to flywheel and the cryptosync input to the cipher would change state.

Typically, sync hits are expected to be in the range of ±3 frames. Thus the only out-of-sync induced noise would occur for up to 3 frames duration around the sync frame time.

5 Summary

The ciphering and synchronization part of E2E encryption for either the sync frame-only or the hybrid method would reside as an application between the codec and error protection/detection layer. On the both transmitting and receiving sides, this application would contain a sync frame counter. On the transmitting side, the application would insert sync frames into the CRC-protected area of the frame, based on the state of the sync frame counter. On the receiving side, the application will detect sync frames and resynchronize its sync frame counter accordingly. It will also signal to the error concealment function, via the BFI bit, when a sync frame was received so that masking could occur. (The Class B (*Class 1b*) and Class C (*Class 2*) bits are unaffected by the sync frames, making error concealment an attractive process.)

6 Application Layer Option

As described in the summary, the ciphering and synchronization part of E2E encryption would reside in an application between the codec and error handling layer. As such, it can be viewed as being part of the application layer. To allow a fully application-layer option, key management could also be brought to this layer. At the start of a call, the E2E applications would use signalling and public keying (e.g. Diffie-Hellman) to establish a session key. This key would be established in a system-independent manner: It doesn't matter whether the call is CS, VoIP, IP-phone, UMTS, GERAN, or any mixture of these. The type of session key generation would be defined by a parameter delivered in an initial message.

For Lawful Intercept purposes, each application sends its key to the UMTS message handler which in turn transmits the key, air interface encrypted, to its respective base station, gateway, etc. in a system dependent manner.

The intent here is to make the system-dependent portion an add-on rather than having it be an integral part of the design. Doing so may bypass much system design. In effect, the encrypted voice and its attendant key management would be tunneled through the 3GPP system. Such an approach may speed standardization and development. The downside here would be the public keying overhead. However, this method of session key generation would be specified as an option, with another system-dependent method ultimately specified as the default.

The above advantages and further advantages are summarized below:

1. The integral part of the design is system independent, and thus applies to CS, VoIP, UMTS, GERAN, and to any mixture of these. Also, it would be compatible with non-3GPP termini such as IP phones, were such to employ the same codec plus E2E application.
2. The design potentially allows an earlier release of E2E encryption due to the lessened need for involvement from other 3GPP groups.
3. It provides a model for E2E encryption other codec types such as the Wideband AMR codec, and other real-time, transparent multimedia (if such are identified).

Later releases could define a faster, 3GPP-style of session key generation. In fact, the study group felt that session keys could be derived from the CKs at both ends.

7 References

[1] 3GPP SA3 contribution, S3-010093, "End-to-End Encryption of Wireless VoIP (E3 VoIP)", TSG-SA WG3 (Security) meeting #17, Göteborg, 27th February – 2nd March 2001.

[2] "Terrestrial Trunked Radio (Tetra); Voice plus Data (V+D); Part 7: Security", ETS 300-392-7, section 7.4.2, Edition 1 – 1996-12.