A precis of the new attacks on GSM encryption

Greg Rose, QUALCOMM Australia,

10 September, 2003.

There's very little information in the various press releases about these attacks, but at the same time probably too much information in the actual paper. So I'm writing this to attempt a readable explanation of the attacks.

First, the paper itself: by Elad Barkan, Eli Biham and Nathan Keller of Technion in Haifa, Israel, "Instant Ciphertext-Only Cryptanalysis of GSM Encrypted Communications" appeared at Crypto 2003 in Santa Barbara about three weeks ago. It was another week or two before the press noticed it. The paper finally became available on the Web yesterday (2003-09-09) at <http://cryptome.org/gsm-crack-bbk.pdf> . Barkan is the principal author.

Background about GSM encryption

The GSM voice calls are encrypted using a family of algorithms collectively called A5. A5/0 is no encryption. A5/1 is the "standard" encryption algorithm, while A5/2 is the "export" (weakened) algorithm. A5/3 is a new algorithm based on the UMTS/WCDMA algorithm Kasumi. While one of the attacks below manages to "walk around" A5/3, there is no attack against it directly.

GPRS encryption is done with a different family of algorithms: GEA0 (none), GEA1 (export), GEA2 (normal strength) and GEA3 (new, and effectively the same as A5/3). Note that the GEA1 and GEA2 algorithms do not have any relationship to the A5/1 and A5/2 algorithms, and they are not publicly known. There are no problems with any of these in the open literature.

All of these algorithms use a 64-bit key derived from a common mechanism: the mobile receives a random challenge, then the SIM card (a smartcard used to keep the subscriber's master key secret) calculates an authentication signature and an encryption key. The key calculated does *not* depend on what algorithm it is destined to be used with.

The encryption is done using a stream cipher, that is, the encryption algorithm takes the secret key and a frame number, and generates a pseudo-random stream of bits (keystream) that are XORed with the input to encrypt it, or are XORed with the received bits to decrypt them. Thus, the bits are effectively encrypted independently of one another.

The encryption is done *after* coding for error correction. The coding introduces known linear relationships between the bits to be encrypted; so even though the attacker might not know the values of particular input bits, they know that certain groups of them XOR to 0. So, taking the same groups of encrypted bits and XORing them reveals the corresponding XOR of the keystream bits. This is the fundamental problem that allows the attacks to work

without any knowledge at all of what is being encrypted, which is what they mean by "ciphertext only". This is a very new result.

The attacks:

There are effectively three different attacks discussed in the paper.

The fundamental attack is against A5/2. By doing a one-time precomputation and storing the results on a decent-sized disk, you can now intercept 4 frames of A5/2 encrypted voice, which yields enough known linear relationships in the keystream to look up the key. The attack is almost instantaneous, and requires only milliseconds of encrypted voice. This is a "passive attack", requiring only eavesdropping, and no-one can tell that it has been done. Once the key has been recovered, it can be used to decrypt the actual frames in both directions.

The second attack uses the fact that the first attack is so fast to interfere with the GSM protocol. This is an "active attack", meaning the attacker has to be able to interfere with the communication. (More detail about the active attacks below.) In practice, this means they would need something pretending to be a base station, but such attacks have happened in the past. Commercially available test equipment can do it, and really it's not much more than two cellphones back-to-back. Basically, even though the real base station and the mobile would both prefer to use a stronger encryption algorithm than A5/2, the false base station can convince the mobile to use A5/2 long enough to break it, recover the key, and respond to the real base station with correctly encrypted stuff using the stronger algorithm. This works because the same key is used for both the weak and the stronger algorithms.

The third attack is not nearly so elegant, but the simple fact is that if the other two attacks were not mentioned in the paper, it would have been very scary (and still publishable) all by itself. The same trick as above, identifying linear relationships in the encrypted frames, is used to classify output keystream in a way that makes it amenable to searching using a technique called a "time-memory tradeoff". The attacker takes four consecutive encrypted frames, runs a processing phase on them, and checks if the output is in their database. If they are lucky, it is, and they can then very quickly determine the input key. If they are not, they try again with a different set of four frames. Eventually, they will get lucky, and how long this takes depends on the size of the database, which in turn depends on how much time they've spent initially computing it.

Section 6 of the published paper shows that an A5/1 key can be recovered in real time (by which they mean less CPU time than it takes to intercept the data) with:
- 5 minutes of intercepted frames (doesn't have to be one call)
- 4.4 terabytes of disk (not that much these days)
- a lab full of PCs for a year's worth of one-time precomputation.

(This is just one point in the tradoff curve; I happen to think it's the sweet spot. But the NSA, for example, could compute a bigger database and do it with much less intercepted call duration.)

Note that the key recovery success is probabilistic. If you have 2.5 minutes of call, you have 50% chance of key recovery. If you listen to 10 calls at a time, you'll get a key every 30 seconds, roughly speaking.

More about the active attacks

The important thing about the active attacks is that the attacker can confuse a mobile into doing what it wants the mobile to do. At the limit, if the attacker has intercepted the random challenge sent to a particular mobile and has recorded all the traffic, whether it is GSM voice or GPRS data, they can later send the same random challenge to the mobile and tell it to use A5/2 to communicate. When the mobile responds, they recover the key, and it's the same key that will decrypt the recorded stuff, whatever it was encrypted with. What's more, the process with A5/2 is so fast that it can be done between the time the real network sends a challenge to the mobile and when it would time out waiting for a valid response.

And what if the target is using A5/1 but the attacker didn't get enough data to recover the key? They can simply call the target, ask for Bob, sound confused, verify that they called the right number, apologise, dither, and eventually the key will pop out. This currently works because the network usually keeps the same key in use for multiple calls. This also gets around any problems with identifying the correct target in the first place; the attacker just needs to know their phone number.

Lastly, once you have the key, you can do things with it that qualify as active attacks, such as originating calls or hijacking data sessions, that will be undetectable (at least by cryptographic means) by the network. The exposure will continue until the network issues a new challenge.

Proposals to address the problem

There are a number of proposals being bandied about to address the attacks. Some of these are still at a confidential stage, so I won't go into them. Others are obvious, though:

1. A5/2 has to come out of new handsets and should not be used by legitimate networks.
2. The mobile should be challenged (and a new key established) for every new service.

Conclusion

I continue to have job security. :-)

Greg Rose INTERNET: ggr@qualcomm.com
Qualcomm Australia VOICE: +61-2-9817 4188 FAX: +61-2-9817 5199
Level 3, 230 Victoria Road, http://people.qualcomm.com/ggr/
Gladesville NSW 2111 232B EC8F 44C6 C853 D68F E107 E6BF CD2F 1081 A37C