Choosing new authentication and key generation algorithms for mobiles

Steve Babbage
Vodafone Group R&D

14-18 January 2013
Standards groups
First generation
GSM security architecture
GSM security limitations

> Key length
> One-way authentication
> Unprotected signalling
> A5/1, A5/2
UMTS security architecture (slightly simplified)

Authentication and key agreement algorithm f1–f5

RAND, SQN, MAC

RES

RES = XRES?

Encryption algorithm UEA, integrity algorithm UIA

SIM

Visited network

Home network

AKA

RAND

SQN

K

XRES

CK

MAC

IK

Check SQN

Check MAC

ENCRYPT USING CK

INTEGRITY PROTECT USING IK
Milenage

\[ \text{SQN} || \text{AMF} || \text{SQN} || \text{AMF} \]

\[ \text{Rijndael} \]

\[ \text{OP}_C \]

\[ \text{E}_K \]

\[ \text{OP}_C \rightarrow \]

\[ \text{rotate by r1} \]

\[ \text{c1} \]

\[ \text{OP}_C \rightarrow \]

\[ \text{E}_K \]

\[ \text{OP}_C \rightarrow \]

\[ \text{MAC(64)} \]

\[ \text{MAC}^*(64) \]

\[ \text{AK(48)} \]

\[ \text{RES(64)} \]

\[ \text{CK(128)} \]

\[ \text{IK(128)} \]

\[ \text{AK}^*(48) \]
Allow customising

> To make each operator's implementation different

> To prevent USIMs for operators being interchangeable, either through trivial modification of inputs and outputs or by reprogramming of a blank USIM

> To keep some algorithm details secret
  
  – Makes some attacks a little harder
Customising Milenage

SQN||AMF||SQN||AMF

RAND

OP\_C

E\_K

c1

rotate by r1

f1

f1*

f2

f5

OP\_C

E\_K

c2

rotate by r2

OP\_C

c3

rotate by r3

OP\_C

f3

OP\_C

E\_K

f4

OP\_C

E\_K

f5*

rotate by r4

rotate by r5
Why do we want a second algorithm?
Design considerations

> Trusted kernel
  – Don’t want to take several years
> Security proof, assuming the kernel is strong
> Can be a drop-in replacement for Milenage …
  – … or can accommodate a 256-bit key
> Efficiency
  – 8-bit out
  – 16-bit, 32-bit in
> Fundamentally different from Milenage
> Side channel attacks
### Candidates for a block cipher kernel

<table>
<thead>
<tr>
<th></th>
<th>ARIA</th>
<th>BLOWFISH</th>
<th>CAMELLIA</th>
<th>CAST</th>
<th>CLEFIA</th>
<th>IDEA</th>
<th>MARS</th>
<th>RC6</th>
<th>SEED</th>
<th>SERPENT</th>
<th>TEA, XTEA</th>
<th>TWOFISH</th>
</tr>
</thead>
<tbody>
<tr>
<td>May fall with AES (XSL-type attacks)</td>
<td>x</td>
<td>x</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>&lt; 256 bit key</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Key-dependent S-box</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>x</td>
<td></td>
<td></td>
</tr>
<tr>
<td>32-bit multiplications</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>x</td>
<td></td>
<td></td>
</tr>
<tr>
<td>IPR</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>x</td>
<td></td>
<td>x</td>
</tr>
</tbody>
</table>

- No need for kernel to be invertible
- Hash functions are a natural alternative to block ciphers
Candidates for a hash function kernel

- BLAKE
- Grøstl - Too close to AES
- SHA-2
- JH
- Keccak
- Skein
**Input and output sizes**

- **Authentication (MAC)**
  - \( \leq 256 \) K
  - 128 RAND
  - 48 SQN
  - 16 AMF

  \(--\) \( \leq 448 \)

- **Key generation**
  - \( \leq 256 \) K
  - 128 RAND

  \(--\) \( \leq 384 \)

- **Output Sizes**
  - 64 MAC
  - 64 MAC-A
  - 128 RES
  - 128 CK
  - 128 IK
  - 48 AK
  - 48 AK*

  \(--\) 480
Keyed MAC constructions

**SHA-2**

> With SHA-512, inputs fit into one block and outputs fit into one hash
> HMAC is the traditional, trusted choice
>   – Some precomputations possible, but still need two hash calls
> Simpler construction for fixed length input?
>   – Secret prefix, Hash (key || message)? No length extension attacks

**KECCAK**

> Documentation recommends Hash (key || IV-if-required || message) for MACs
Keccak considerations

> On the security of the keyed sponge construction, Bertoni / Daemen / Peeters / van Assche (SKEW 2011)

**Theorem 1.** The advantage of distinguishing $\text{KEYEDSPONGE}[f, K]$ from a random oracle, with $f$ a random permutation and $K$ uniformly distributed, is upper bounded by:

$$1 - \exp \left( -\frac{M^2}{2} + 2(M + 1)(N + 1) \right) + P_{\text{key}}(N),$$

where $M$ is the data complexity, $N$ the time complexity and $P_{\text{key}}(N)$ the probability of guessing the key after $N$ queries.

> Parameter choices

- State size 1600: Capacity 512, Rate 1088
- State size 1600: Capacity 1024, Rate 576
- State size 800: Capacity 512, Rate 288

> Wait for the SHA-3 standard?
SHA-2
> Better trusted?

Keccak
> More efficient?
  – 16-bit platforms
  – Large chaining value, though

> Simpler MAC construction?

> Easier to protect against side channel attacks?

SHA-2
> More implementations available
  – For SHA-256, anyway
Thank you