Single-query Quantum Hidden Shift Attacks

Xavier Bonnetain[1]
André Schrottenloher[2]

[1]Université de Lorraine, CNRS, Inria, LORIA
[2]Inria, Univ Rennes, CNRS, IRISA

Outline

1. Quantum Attacks in Symmetric Crypto
2. Interlude: Post-processing
3. Single-query Hidden Shift
4. Application to Tiaoxin

Quantum Attacks in Symmetric Crypto

Context: nonce-based authenticated encryption

Authenticated encryption (AE):

$$x \text{ (plaintext)} \longrightarrow \boxed{E_{K,N}} \longrightarrow y, t = E_{K,N}(x)$$

ciphertext **and** authentication tag

- ***N* shall not be reused**

= Adversary has a **black-box** (oracle) that encrypts:

$$x \rightarrow \boxed{E_{K,N}} \rightarrow y, t$$

(One of) the goal(s) is to find **K**.

Quantum computing in a single slide

Quantum state (**n** qubits):

- $|\psi\rangle = \sum_{x\in\{0,1\}^{\mathbf{n}}} \alpha_x |x\rangle$
- α_x are complex numbers (**amplitudes**)
- Measurement outputs x with prob. $|\alpha_x|^2$

We transform the state using **unitary operations**, then measure.

(Typical) operations:

- Classical **reversible** operations "in superposition": transform each bit-string $|x\rangle \mapsto |\mathcal{A}(x)\rangle$
- Hadamard transform: turn $\sum_x f(x)|x\rangle$ into $\frac{1}{2^{\mathbf{n}/2}}\sum_x \widehat{f}(x)|x\rangle$ (up to normalization)

$$\widehat{f}(x) := \sum_y (-1)^{x\cdot y} f(y)$$

The two quantum adversaries

The "standard" (Q1)

$$x \longrightarrow \boxed{E_{K,N}} \longrightarrow y, t$$

- Adversary is quantum
- Black-box is classical

The "superposition" (Q2)

$$|x\rangle\,|0\rangle \longrightarrow \boxed{E_{K,N}} \longrightarrow |x\rangle\,|y,t\rangle$$

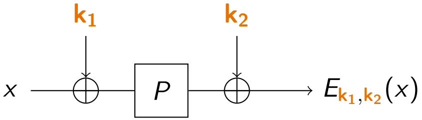
- Adversary is quantum
- Black-box **is quantum**

Making sense of the Q2 model

- Q1 adversary can be "store now, decrypt later"
- Q2 adversary needs to be **active**

- The model is **non-trivial**
- Attacks give valuable information for **provable security**
- There are relations between the Q1 and Q2 models

Example: Even-Mansour cipher



$$E_{k_1,k_2}(x) = k_2 \oplus P(x \oplus k_1)$$

- **Q1 secure [ABKM22]**
- **Q2 insecure [KM12]**

Consider the function:

$$f(x) = E_{k_1,k_2}(x) \oplus P(x) \implies f(x \oplus k_1) = f(x) \ .$$

In Q2, finding k_1 is an **easy** quantum problem (hard in Q1).

Kuwakado, Morii, "Security on the quantum-type even-mansour cipher", ISITA 2012

Alagic, Bai, Katz, Majenz, "Post-Quantum Security of the Even-Mansour Cipher", EUROCRYPT 2022

Simon's algorithm

- Replace f by g: $\{0,1\}^{\mathbf{n}} \to \{-1,1\}$ with $g(x \oplus \mathbf{k_1}) = g(x)$.
- **Phase oracle** for g: $|x\rangle \mapsto g(x)|x\rangle$ **can be implemented** (if you can compute g)

1. Start from $|0\rangle$
2. Apply H: $\frac{1}{\sqrt{2^{\mathbf{n}}}} \sum_x |x\rangle$
3. Apply phase oracle: $\frac{1}{\sqrt{2^{\mathbf{n}}}} \sum_x g(x)|x\rangle$
4. Apply H again: $\frac{1}{2^{\mathbf{n}}} \sum_y \widehat{g}(y)|y\rangle$

Simon's algorithm (ctd.)

Lemma

- If $y \cdot \mathbf{s} = 1$, then

$$\widehat{g}(y) = \sum_x (-1)^{x \cdot y} g(x) = \sum_{\text{Half space}} \left((-1)^{x \cdot y} + (-1)^{x \cdot (y \oplus \mathbf{s})} \right) g(x) = 0$$

- One can only measure y such that $y \cdot \mathbf{s} = 0$

- With g "random enough", a single query $\implies$ 1 bit of information on $\mathbf{s}$

$\implies$ $\mathcal{O}(\mathbf{n})$ queries to succeed

Q2 attacks in symmetric crypto

If you authorize Q2 queries, all of these can be broken with low effort:

- Even-Mansour cipher
- 3-round Feistel PRP
- CBC-MAC, PMAC, GMAC, GCM, OCB
- ΘCB, LightMAC, LightMAC+, Deoxys, ZMAC, PMAC, PolyMAC, GCW-SIV(2)...

Making sense of Q2 with nonce-based AE

- The adversary still has quantum access
- **But** nonce is **classical** and changes at each query

$$|x\rangle\,|0\rangle \longrightarrow \boxed{E_{K,N_1}} \rightarrow |x\rangle\,|y,t\rangle \qquad |x\rangle\,|0\rangle \longrightarrow \boxed{E_{K,N_2}} \rightarrow |x\rangle\,|y,t\rangle$$

$$|x\rangle\,|0\rangle \longrightarrow \boxed{E_{K,N_3}} \rightarrow |x\rangle\,|y,t\rangle \qquad \dots$$

Limitation of Simon's period-finding

- Simon's algorithm only finds **1 bit of information per query**
- The function can be different but the period **s** has to **be the same**

But what if **s** depends on N?

Interlude: Post-processing

Oracle post-processing

To build the periodic function, we often do not need all of the cipher's output.

- If we can do multiple queries, post-processing is easy.
- What about a single query?

Linear post-processing

In general:

Given access to oracles O_f and O_g, one can emulate $O_{g\circ f}$ using one query to O_g and two queries to O_f.

$$|x\rangle\,|0\rangle\,|0\rangle \rightarrow |x\rangle\,|f(x)\rangle\,|0\rangle \rightarrow |x\rangle\,|f(x)\rangle\,|g(x)\rangle \rightarrow |x\rangle\,|0\rangle\,|g(x)\rangle$$

We need more:

Given access to oracles O_f and O_g **where g is a linear function**, *one can emulate $O_{g\circ f}$ using* **two queries to O_g** *and* **one query to O_f.**

$\Longrightarrow$ we can linearly "post-process" oracles!

Hosoyamada, Sasaki, "Quantum Demiric-Selçuk meet-in-the-middle attacks: Applications to 6-round generic Feistel constructions", SCN 2018

Bhaumik, Bonnetain, Chailloux, Leurent, Naya-Plasencia, S. Seurin, "QCB: efficient quantum-secure authenticated encryption", ASIACRYPT 2021

Linear post-processing (ctd.)

$$
\begin{aligned}
\text{Start:}\quad & |x\rangle\,|y\rangle \\
\text{Uniform superposition:}\quad & |x\rangle\,|y\rangle \sum_z |z\rangle \\
\text{Query } O_g\text{:}\quad & \sum_z |x\rangle\,|y \oplus g(z)\rangle\,|z\rangle \\
\text{Query } O_f\text{:}\quad & \sum_z |x\rangle\,|y \oplus g(z)\rangle\,|z \oplus f(x)\rangle \\
\text{Query } O_g\text{:}\quad & \sum_z |x\rangle\,|y \oplus g(z) \oplus g(z \oplus f(x))\rangle\,|z \oplus f(x)\rangle \\
\text{Use linearity of } g\text{:}\quad & = \sum_z |x\rangle\,|y \oplus g(f(x))\rangle\,|z \oplus f(x)\rangle \\
& = |x\rangle\,|y \oplus g(f(x))\rangle \sum_z |z \oplus f(x)\rangle \\
\text{Uncompute last register:}\quad & |x\rangle\,|y \oplus g(f(x))\rangle
\end{aligned}
$$

Single-query Hidden Shift

Single-query hidden shift

Let $g\ :\{0,1\}^{\mathbf{n}} \to \{-1,1\}$. With a single query to $|x\rangle \mapsto g(x \oplus \mathbf{s})\,|x\rangle$, can we recover $\mathbf{s}$?

Ozols, Roetteler, Roland, "Quantum rejection sampling". ACM Trans. Comput. Theory 2013

Single-query hidden shift (ctd.)

Uniform superposition: $\sum_x |x\rangle$

Call oracle: $\sum_x g(x \oplus \mathbf{s}) |x\rangle$

Do H transform: $\sum_y \left(\sum_x (-1)^{x \cdot y} g(x \oplus \mathbf{s}) \right) |y\rangle$

$$= \sum_y (-1)^{y \cdot \mathbf{s}} \underbrace{\left(\sum_x (-1)^{x \cdot y} g(x) \right)}_{= \widehat{g}(y)} |y\rangle$$

multiply by $1/\widehat{g}(y)$: $\sum_y (-1)^{y \cdot \mathbf{s}} |y\rangle$

H again: $|\mathbf{s}\rangle$

Single-query hidden shift (ctd.)

Uniform superposition: $\sum_x |x\rangle$

Call oracle: $\sum_x g(x \oplus \mathbf{s}) |x\rangle$

Do H transform: $\sum_y \left(\sum_x (-1)^{x \cdot y} g(x \oplus \mathbf{s}) \right) |y\rangle$

$$= \sum_y (-1)^{y \cdot \mathbf{s}} \underbrace{\left(\sum_x (-1)^{x \cdot y} g(x) \right)}_{= \widehat{g}(y)} |y\rangle$$

multiply by $1/\widehat{g}(y)$: $\sum_y (-1)^{y \cdot \mathbf{s}} |y\rangle$ $\Leftarrow$ WHAT???

H again: $|\mathbf{s}\rangle$

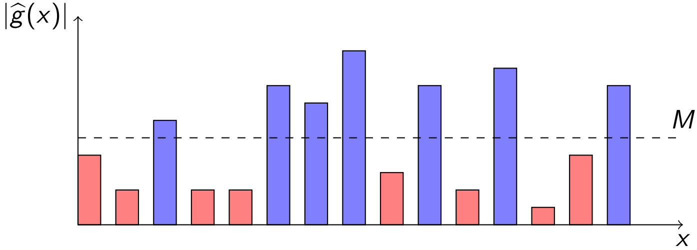
Rejection sampling

The operation:

$$\sum_y (-1)^{y\cdot s} \widehat{g}(y) |y\rangle \mapsto \sum_y (-1)^{y\cdot s} |y\rangle$$

is not unitary and sometimes $\widehat{g}(y)$ is zero ...

- We will succeed with some probability
- To maximize the success, we **cut off the small values of $\widehat{g}(y)$:** choose a bound M and $G := \#\{x, |\widehat{g}(x)| \geq M\}$.



Rejection sampling (ctd.)

Let $\alpha_y := \frac{M}{\widehat{g}(y)}$ if $|\widehat{g}(y)| \geq M$ and 0 otherwise.

$$\sum_y (-1)^{y \cdot s} \widehat{g}(y) |y\rangle$$

Compute $\widehat{g}(y)$: $\sum_y (-1)^{y \cdot s} \widehat{g}(y) |y\rangle |\widehat{g}(y)\rangle$

Append ancilla qubit: $\sum_y (-1)^{y \cdot s} \widehat{g}(y) |y\rangle |\widehat{g}(y)\rangle |0\rangle$

Controlled-rotate the ancilla: $\sum_y (-1)^{y \cdot s} \widehat{g}(y) |\widehat{g}(y)\rangle |y\rangle \left(\alpha_y |0\rangle + \sqrt{1-\alpha_y^2} |1\rangle\right)$

Uncompute $\widehat{g}(y)$: $\sum_y (-1)^{y \cdot s} \widehat{g}(y) |y\rangle \left(\alpha_y |0\rangle + \sqrt{1-\alpha_y^2} |1\rangle\right)$

$$\sqrt{\text{Prob. of success}} |\text{State we want}\rangle |0\rangle + (\dots) |\text{Garbage}\rangle |1\rangle$$

Rejection sampling (ctd.)

Step 1: measuring 0 in the ancilla

We obtain:

$$\frac{1}{\sqrt{G}} \sum_{y, |\widehat{g}(y)| \geq M} (-1)^{y \cdot \mathbf{s}} |y\rangle \quad .$$

Succeeds w.p. $p := \frac{1}{2^{2\mathbf{n}}} \sum_y |\alpha_y \widehat{g}(y)|^2 = \frac{M^2}{2^{2\mathbf{n}}} G$. **We know if we failed.**

Step 2: measuring s

The final state is not exactly $\sum_y (-1)^{y \cdot \mathbf{s}} |y\rangle$. After H we measure $\mathbf{s}$ w.p.: $p' := \frac{G}{2^{\mathbf{n}}}$. **We don't know if we failed.**

- $pp' = \frac{M^2 G^2}{2^{3\mathbf{n}}}$
- A single query to $g(\cdot \oplus \mathbf{s})$
- Need to compute $\widehat{g}(y) \implies g$ has to be "simple"

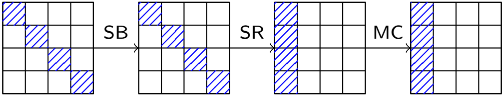
Application to Tiaoxin-346

Specification of Tiaoxin-346

Tiaoxin is an AES-based AE.

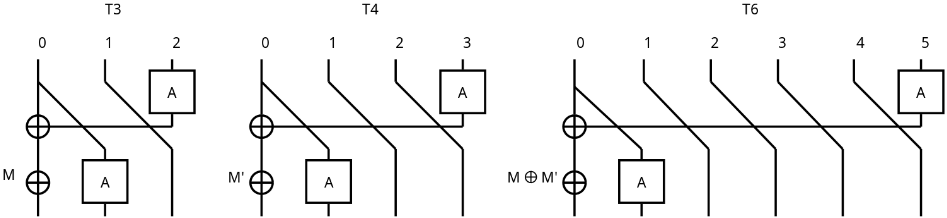
- AES state: 4×4 matrix of bytes
- AES round: $A = MC \circ SR \circ SB$

$\Longrightarrow$ SB the only non-linear operation



- The internal state $T = T_3 \| T_4 \| T_6 = 13$ registers (1664 bits)

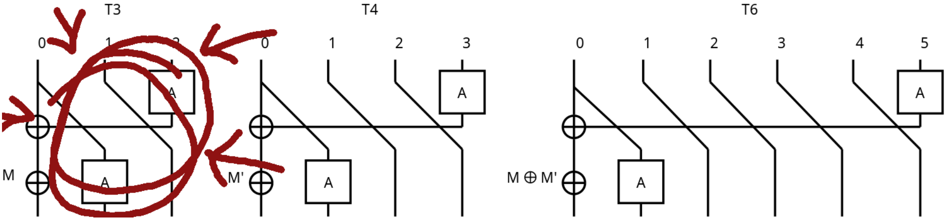
Specification of Tiaoxin-346 (ctd.)



- Initialization: **load K**, N and apply 15 unkeyed rounds
- Finalization: apply 20 unkeyed rounds, then tag $=$ XOR or all registers
- Encryption:

$$\begin{cases} T \leftarrow R(T, M_i, M_i', M_i \oplus M_i') \\ C_i = T_3[0] \oplus T_3[2] \oplus T_4[1] \oplus (T_6[3] \& T_4[3]) \\ C_i' = T_6[0] \oplus T_4[2] \oplus T_3[1] \oplus (T_6[5] \& T_3[2]) \end{cases}$$

Specification of Tiaoxin-346 (ctd.)



- Initialization: **load** **K**, N and apply 15 unkeyed rounds
- Finalization: apply 20 unkeyed rounds, then tag = XOR or all registers
- Encryption:

$$\begin{cases} T \leftarrow R(T, M_i, M_i', M_i \oplus M_i') \\ C_i = T_3[0] \oplus T_3[2] \oplus T_4[1] \oplus (T_6[3] \& T_4[3]) \\ C_i' = T_6[0] \oplus T_4[2] \oplus T_3[1] \oplus (T_6[5] \& T_3[2]) \end{cases}$$

Tiaoxin-346 attack

- We will recover the state $T_3[0,1,2]$ at the beginning of the encryption phase
- By computing backwards the empty rounds, we find **K**

After a few rounds:

$$C'_1 = M_0 \oplus M_1 \oplus M'_0 \oplus M'_1 \oplus T_6[0] \oplus A(\mathbf{M_0} \oplus \mathbf{T_3[0]} \oplus \mathbf{A(T_3[2])}) \oplus$$
$$A(T_6[4]) \oplus A(T_6[5]) \oplus A(T_4[0]) \oplus (T_6[3] \& A(T_3[0]))$$
$$C'_3 = M_0 \oplus M_1 \oplus M_2 \oplus M_3 \oplus M'_0 \oplus M'_1 \oplus M'_2 \oplus M'_3 \oplus$$
$$T_6[0] \oplus A(T_6[3]) \oplus A(T_6[4]) \oplus A(T_6[5]) \oplus A(T_6[2])$$
$$A\left[\mathbf{M_0} \oplus \mathbf{M_1} \oplus \mathbf{M_2} \oplus \mathbf{T_3[0]} \oplus \mathbf{A(T_3[1])} \oplus \mathbf{A(T_3[2])} \oplus \mathbf{A(A(T_3[0]))}\right] \oplus$$
$$A\left[M'_0 \oplus M'_1 \oplus T_4[0] \oplus A(T_4[2]) \oplus A(T_4[3])\right] \oplus$$
$$(T_6[1] \& (A(\mathbf{M_0} \oplus \mathbf{M_1} \oplus \mathbf{T_3[0]} \oplus \mathbf{A(T_3[1])} \oplus \mathbf{A(T_3[2])})))$$

$\Longrightarrow$ **3 variables**, **3 hidden shifts**, the rest is constant

Tiaoxin-346 attack (ctd.)

- XOR C'_1 and C'_3
- Select one bit of each column (assume that $T_6[1] = 1$ for these bits)
- XOR these bits

Recall that $A = MC \circ SR \circ SB$. We obtain an oracle:

$$\underbrace{(x_0, \ldots, x_{47})}_{\text{bytes that make up the 3 variables}} \mapsto \bigoplus_i \underbrace{\alpha_i}_{\text{Linear masks}} \cdot \underbrace{SB}_{\text{S-Box of the AES}} (x_i \oplus \mathbf{s_i})$$

- convert to a phase $(\{0,1\} \to \{-1,1\})$

$$(x_0, \ldots, x_{47}) \mapsto g(x_0, \ldots, x_{47}) := \prod_i \underbrace{(-1)^{\alpha_i \cdot SB(x_i \oplus \mathbf{s_i})}}_{:= g_i(x_i \oplus \mathbf{s_i})}$$

Tiaoxin-346 attack (ctd.)

At this point we can implement: $|x\rangle \mapsto g(x \oplus \mathbf{s})\,|x\rangle$. It remains to:

1. Implement $|x\rangle\,|0\rangle \mapsto |x\rangle\,|\widehat{g}(x)\rangle$
2. Estimate the success probability
3. Find $T_3[0,1,2]$ from the shift

1.

We have: $\widehat{g} = \prod_i \widehat{g_i}$. The g_i are 8-bit functions, computing $\widehat{g_i}$ is easy.

2.

- We know the distribution of $\widehat{g_i}$
- We can compute **exactly** the distribution of $\widehat{g}$

$$M = 2^{195.40}, \quad G = 2^{365.62}, \quad p = 2^{-11.58}, \quad p' = 2^{-18.37}, \quad pp' = 2^{-29.95}$$

Tiaoxin-346 attack (ctd.)

3. We recover:

$$\mathbf{T_3[0] \oplus A(T_3[2])}$$
$$\mathbf{T_3[0] \oplus A(T_3[1]) \oplus A(T_3[2]) \oplus A(A(T_3[0]))}$$
$$\mathbf{T_3[0] \oplus A(T_3[1]) \oplus A(T_3[2])}$$

$\Longrightarrow$ deduce $T_3[0]$, $T_3[1]$, $T_3[2]$

- 2^4 repetitions to have $T_6[1] = 1$ at 4 bits
- $pp' \simeq 2^{30}$
- $\simeq 2^{22}$ nonlinear gates per run of the algorithm

$\Longrightarrow$ 2^{34} Q2 queries and 2^{56} nonlinear gates

What happened

- Observing only C yields an intractable system of equations in T_3, T_4, T_6
- With hidden shifts, we get new equations
- The system may become easier to solve

Conclusion

- This paper: Rocca, Rocca-S, AEGIS-128L(*), Tiaoxin-346 are unsafe against Q2 attacks
- Their common point: broken in the nonce-misuse setting

eprint.iacr.org/2023/1306
gitlab.inria.fr/capsule/single-query-hidden-shift

Thank you!

* Under a heuristic on the distribution of Fourier coefficients of a biased Boolean function.