Practical Cryptanalysis of the Open Smart Grid Protocol

Dumb Crypto in Smart Grids

Philipp Jovanovic[1] (@daeinar)

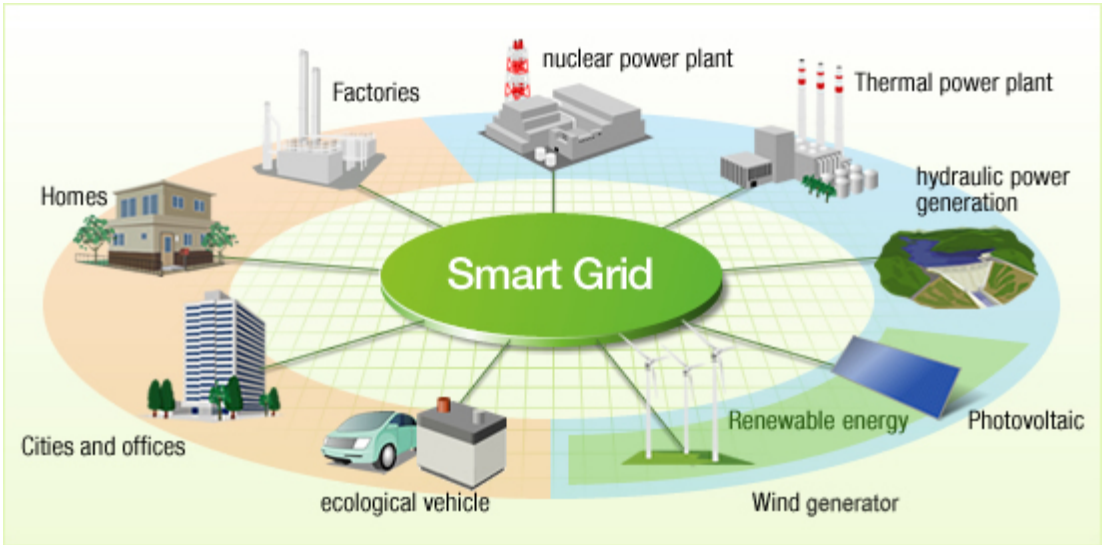
Samuel Neves[2] (@sevenps)

[1]University of Passau, Germany

[2]University of Coimbra, Portugal

Fast Software Encryption 2015
Istanbul, Turkey

Smart Grids



- Definition from Wikipedia:

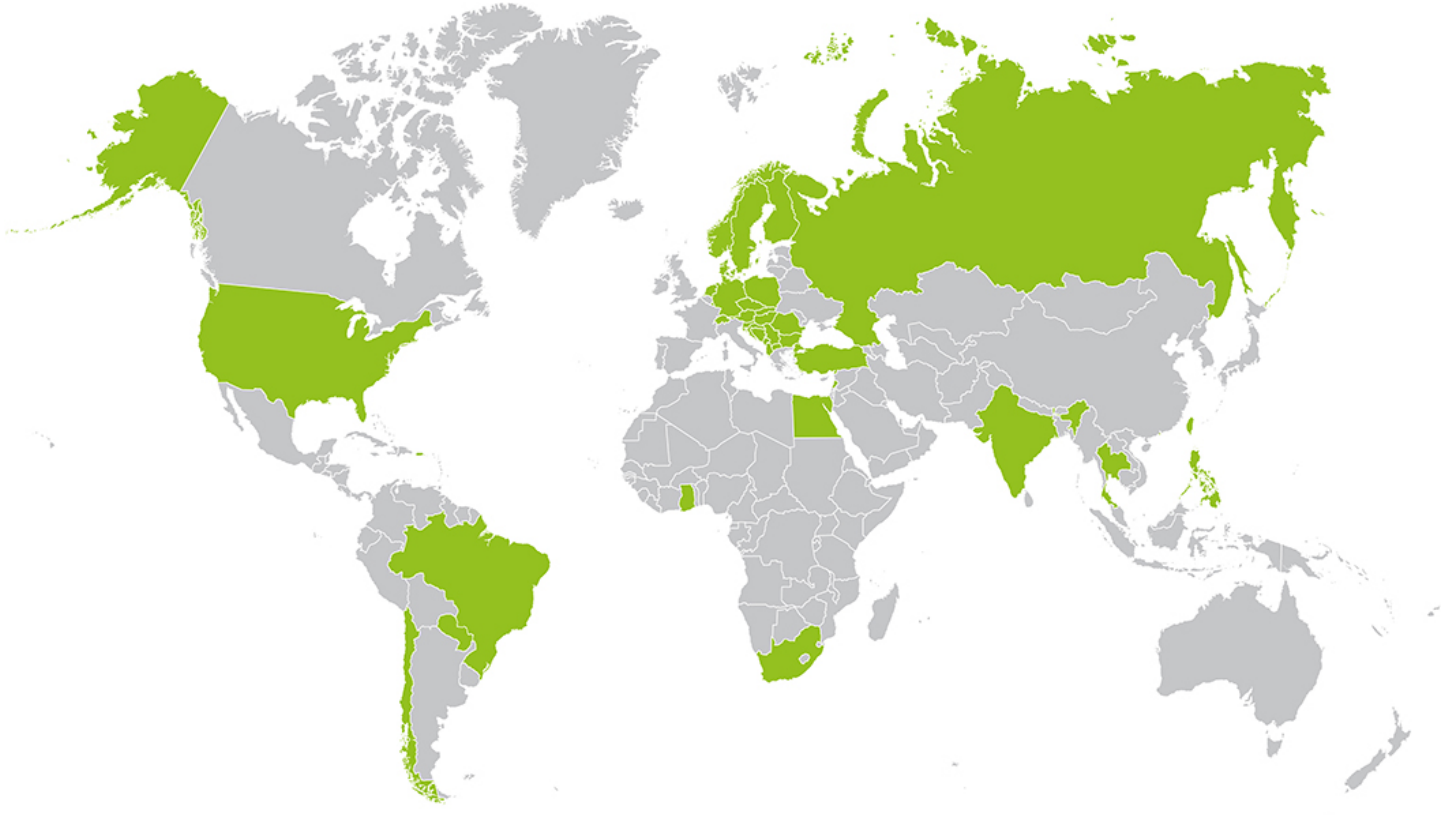
  "A smart grid is a modernized electrical grid that uses analog or digital information and communications technology to gather and act on information [...] in an automated fashion to improve the efficiency, reliability, economics, and sustainability of the production and distribution of electricity."

- Fast-growing technology.
- Critical infrastructure: communication needs protection.

Open Smart Grid Protocol (OSGP)

- **Application layer communication protocol** for smart grids.
- Developed by the **Energy Service Network Association** (ESNA) around 2010.
- Standardised by the **European Telecommunications Standards Institute** (ETSI) in 2012.
- Used in devices sold by **OSGP Alliance**/**Networked Energy Services** (NES).

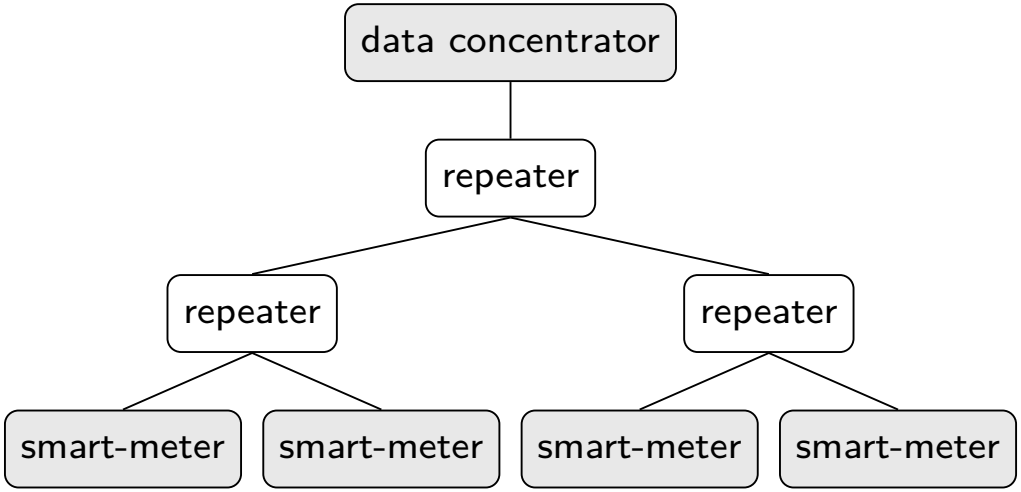
Open Smart Grid Protocol (OSGP)



Source: `http://www.networkedenergy.com/NESworldwide.php`

- Deployed in over **4 million** devices world-wide.
- Customers & Partners of OSGP Alliance/NES: E.ON, Vattenfall, Ericsson AB, Mitsubishi Electric, LG CNS, Oracle, ...

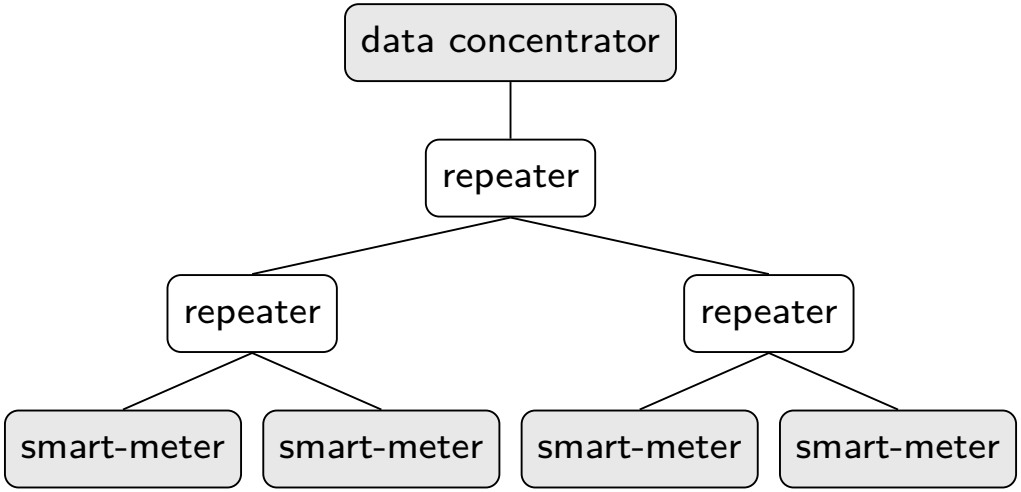
Open Smart Grid Protocol (OSGP)



OSGP's Network Topology

- Message sizes in bytes: 114 (max), 84 (read), 75 (write).
- Encrypted communication between smart-meters and data concentrators.
- **Authenticated encryption scheme:**
  - **RC4** (encryption)
  - **OMADigest** (authentication)
  - **EN14908** (key derivation)

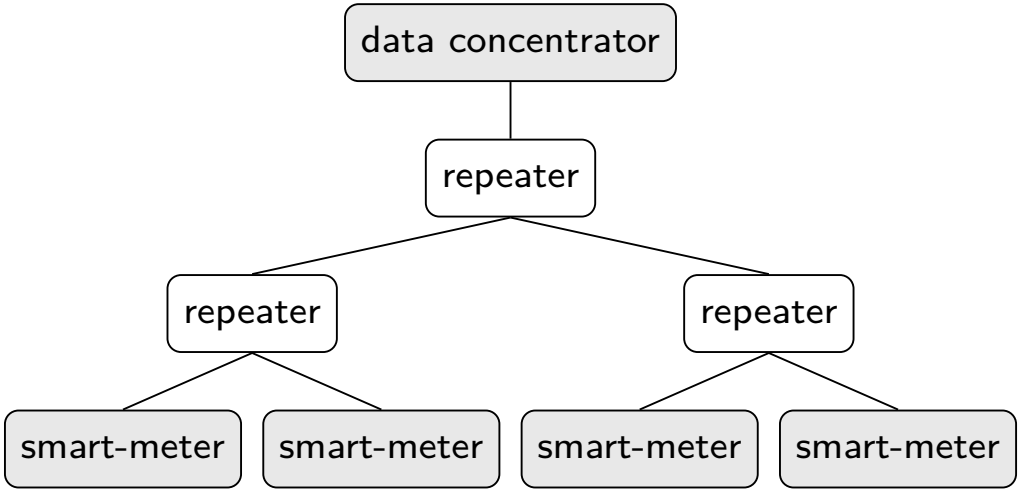
Open Smart Grid Protocol (OSGP)



OSGP's Network Topology

- Message sizes in bytes: 114 (max), 84 (read), 75 (write).
- Encrypted communication between smart-meters and data concentrators.
- **Authenticated encryption scheme:**
  - **RC4** (encryption)
  - **OMADigest** (authentication)
  - **EN14908** (key derivation)

Open Smart Grid Protocol (OSGP)



OSGP's Network Topology

- Message sizes in bytes: 114 (max), 84 (read), 75 (write).
- Encrypted communication between smart-meters and data concentrators.
- **Authenticated encryption scheme:**
  - **RC4** (encryption)
  - **OMADigest** (authentication)
  - **EN14908** (key derivation)

This Talk

Overview

- Cryptanalysis of the OMADigest. **Key recovery** attacks using:
  1. **Differentials**.
  2. **Bruteforce**.
  3. **Differential-based forgeries**.
- Based on publicly available documents.
- No experiments on actual (proprietary) OSGP hardware.
- **Disclosed** to OSGP Alliance/NES in **November 2014**.

Related Work

Structural Weaknesses in the Open Smart Grid Protocol

- By K. Kursawe and C. Peters (European Network for Cyber Security, the Netherlands).
- Overview article on security in OSGP.
- Presents basic attacks.
- Published on the IACR Cryptology ePrint Archive: Report 2015/088.
- **Disclosed** to OSGP Alliance/NES in **early 2014**.

Cryptanalysis of RC4 in OSGP

- By L. Feiten and M. Sauer (University of Freiburg, Germany).
- Transfers WEP attack on RC4 to the case of OSGP.
- Under submission.
- Draft shared privately.
- **Disclosed** to OSGP Alliance/NES in **November 2014**.

Related Work

Structural Weaknesses in the Open Smart Grid Protocol

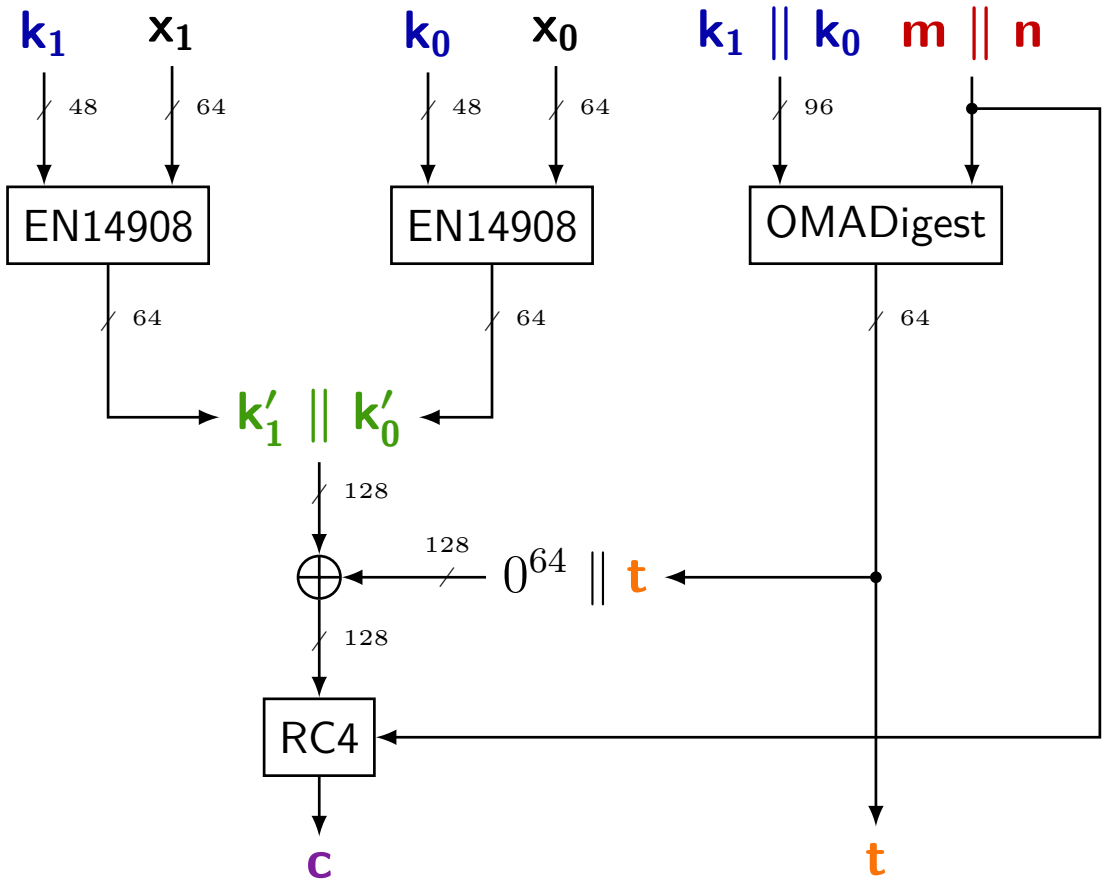
- By K. Kursawe and C. Peters (European Network for Cyber Security, the Netherlands).
- Overview article on security in OSGP.
- Presents basic attacks.
- Published on the IACR Cryptology ePrint Archive: Report 2015/088.
- **Disclosed** to OSGP Alliance/NES in **early 2014**.

Cryptanalysis of RC4 in OSGP

- By L. Feiten and M. Sauer (University of Freiburg, Germany).
- Transfers WEP attack on RC4 to the case of OSGP.
- Under submission.
- Draft shared privately.
- **Disclosed** to OSGP Alliance/NES in **November 2014**.

OSGP's Cryptographic Infrastructure

OSGP's Cryptographic Infrastructure



- $k_1 \parallel k_0$: Open Media Acces Key (OMAK).
- $k'_1 \parallel k'_0$: Base Encryption Key (BEK).
- x_0, x_1: constants.
- $m \parallel n$: message and counter.
- c, t: ciphertext and tag.

The EN14908 "Encryption Algorithm"

9.12 Encryption Algorithm

The LonTalk encryption algorithm facilitates one way encoding rather than real encryption. It uses a 48-bit encryption key K, a variable length APDU, A[len], and a 64-bit input string R to produce a 64-bit output string Y. Desirable properties of the random number R are defined in 9.14. Any 48-bit number is a valid encryption key.

The encryption function is not published in this version of the specification. Echelon has obtained expert advice on one way encryption functions. The advice is that it is impossible to prove beyond any doubt that a function has no inverse. Those who have seen the function as of June, 1994 believe it has no inverse, but Echelon has been advised that it is more secure if it is not published. Nevertheless, Echelon has, and shall continue to make the function available on a need to know basis provided that there is written agreement to keep the function confidential.

Source: `http://www.lonworks.org.cn/en/LonWorks/Lontalk%20Protocol%20Spec.pdf`

The OMADigest is an **"improved"** version of the EN14908 encryption algorithm.

The EN14908 "Encryption Algorithm"

9.12 Encryption Algorithm

The LonTalk encryption algorithm facilitates one way encoding rather than real encryption. It uses a 48-bit encryption key K, a variable length APDU, A[len], and a 64-bit input string R to produce a 64-bit output string Y. Desirable properties of the random number R are defined in 9.14. Any 48-bit number is a valid encryption key.

The encryption function is not published in this version of the specification. Echelon has obtained expert advice on one way encryption functions. The advice is that it is impossible to prove beyond any doubt that a function has no inverse. Those who have seen the function as of June, 1994 believe it has no inverse, but Echelon has been advised that it is more secure if it is not published. Nevertheless, Echelon has, and shall continue to make the function available on a need to know basis provided that there is written agreement to keep the function confidential.

Source: `http://www.lonworks.org.cn/en/LonWorks/Lontalk%20Protocol%20Spec.pdf`

The OMADigest is an **"improved"** version of the EN14908 encryption algorithm.

OMADigest

Function `OMADigest(`m`,`k`)`

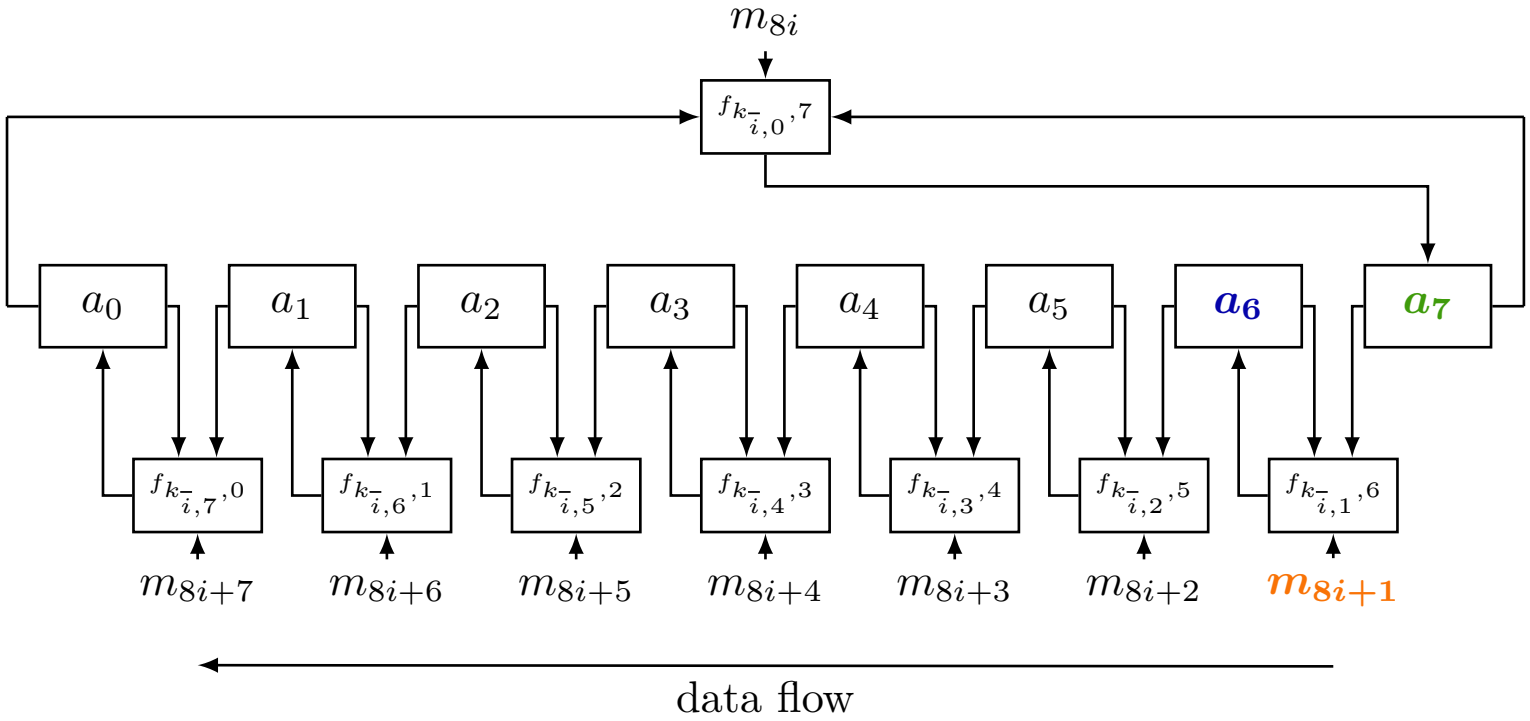
$a \leftarrow (0,0,0,0,0,0,0,0)$
$m \leftarrow m \parallel 0^{-|m| \bmod 144}$
foreach 144-*byte block* b *of* m **do**
  for $i \leftarrow 0$ **to** 17 **do**
    for $j \leftarrow 7$ **to** 0 **do**
      if $k_{i \bmod 12, 7-j} = 1$ **then**
      $a_j \leftarrow a_{(j+1) \bmod 8} + b_{8i+(7-j)} + (\neg(a_j + j)) \lll 1$
      else $a_j \leftarrow a_{(j+1) \bmod 8} + b_{8i+(7-j)} - (\neg(a_j + j)) \ggg 1$
    end
  end
end
return a

Observations

- 64-bit state a.
- Message is zero-padded: $m \mapsto m \parallel 0^{-|m| \bmod 144}$.
- Key extension: $k_0 \parallel \cdots \parallel k_{11} \mapsto k_0 \parallel \cdots \parallel k_{11} \parallel k_0 \parallel \cdots \parallel k_5$.
- Processing of a message byte depends exactly on one key bit.
- State update is almost linear.
- Algorithm is fully reversible.

OMADigest

- Data processing:



- The non-linear update function f:

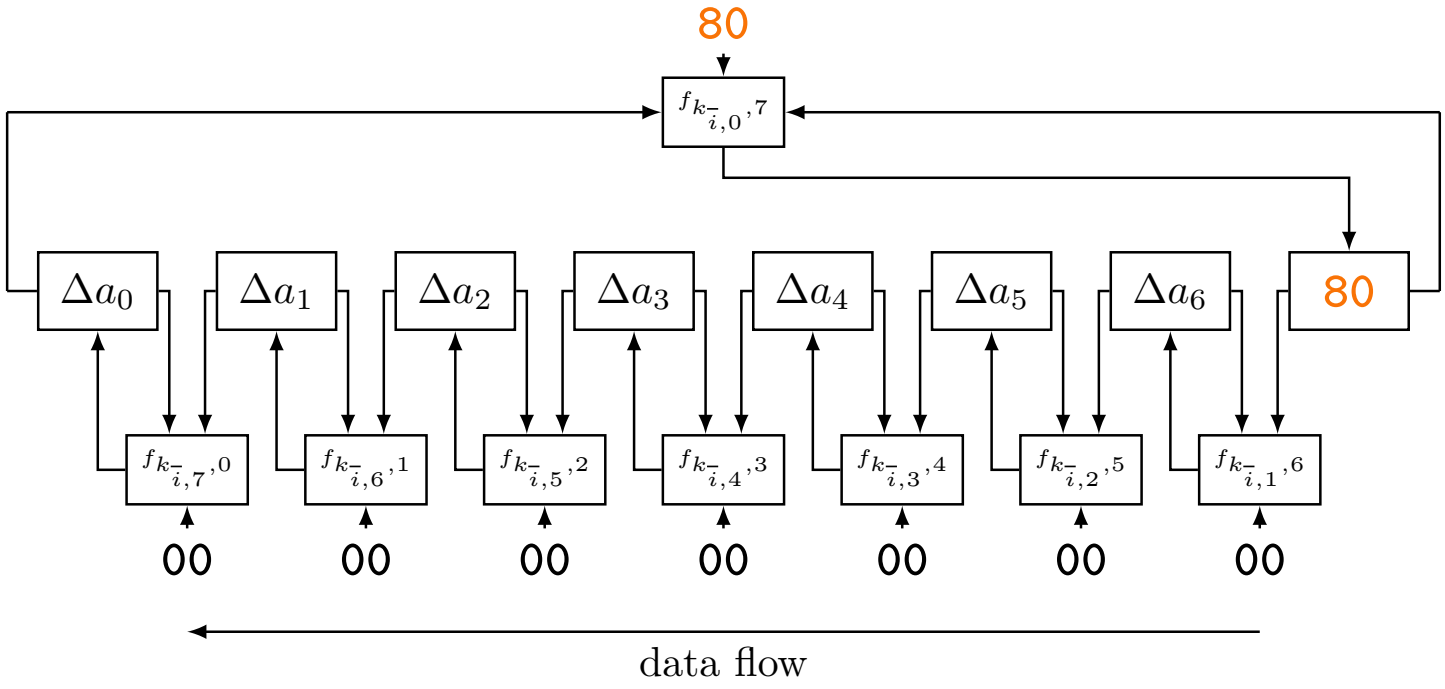
$$f_{k,c}(x, y, m) = \begin{cases} y + m + (\neg(x + c)) \lll 1 & \text{if } k = 1 \\ y + m - (\neg(x + c)) \lll 7 & \text{otherwise.} \end{cases}$$

- Note: $i = 0, \ldots, 17$ and $\bar{i} = i \bmod 12$.

Attack #1

Bitwise Key Recovery

- Injecting XOR-difference $\Delta m_{8i} = 80$:



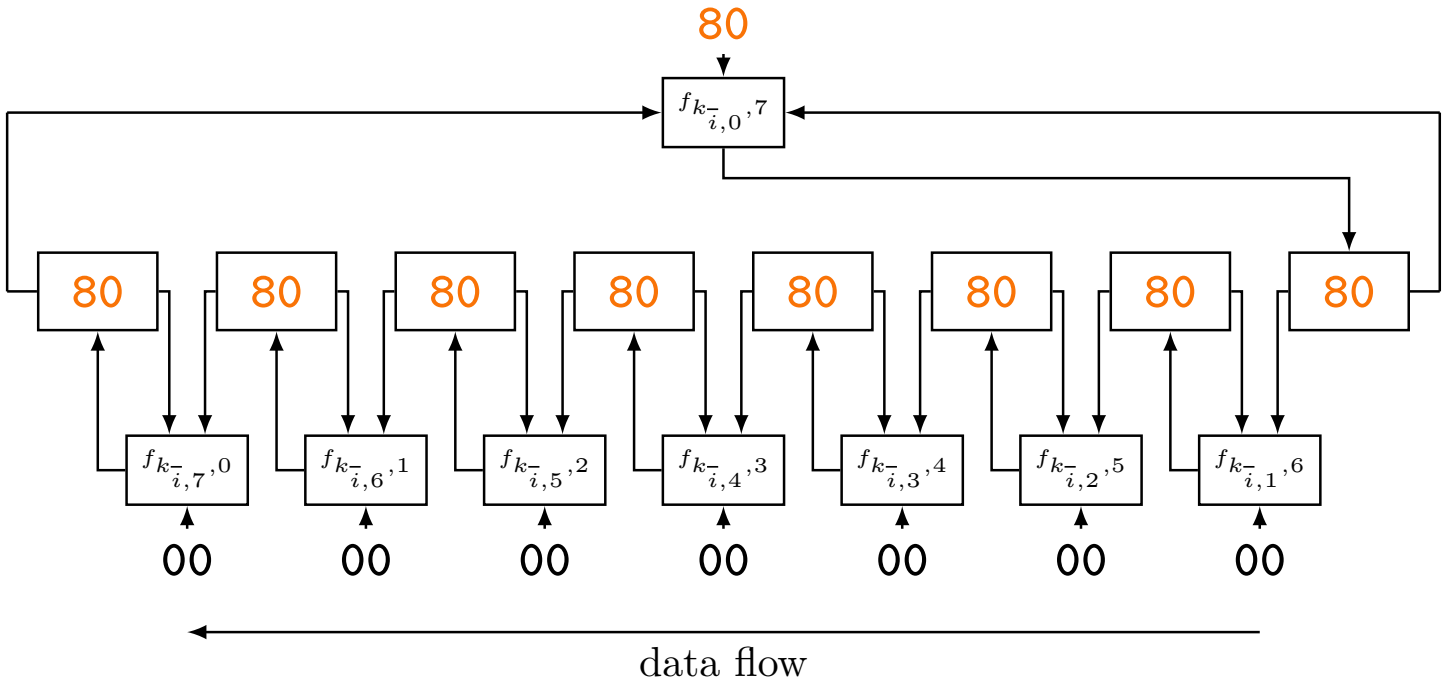
- The non-linear update function f:

$$f_{k,c}(x, y, \boldsymbol{m}) = \begin{cases} y + \boldsymbol{m} + (\neg(x + c)) \lll 1 & \text{if } k = 1 \\ y + \boldsymbol{m} - (\neg(x + c)) \lll 7 & \text{otherwise.} \end{cases}$$

- Note: $i = 0, \ldots, 17$ and $\bar{i} = i \bmod 12$.

Bitwise Key Recovery

- Difference propagation after processing $m_{8i}, \ldots, m_{8i+7}$:



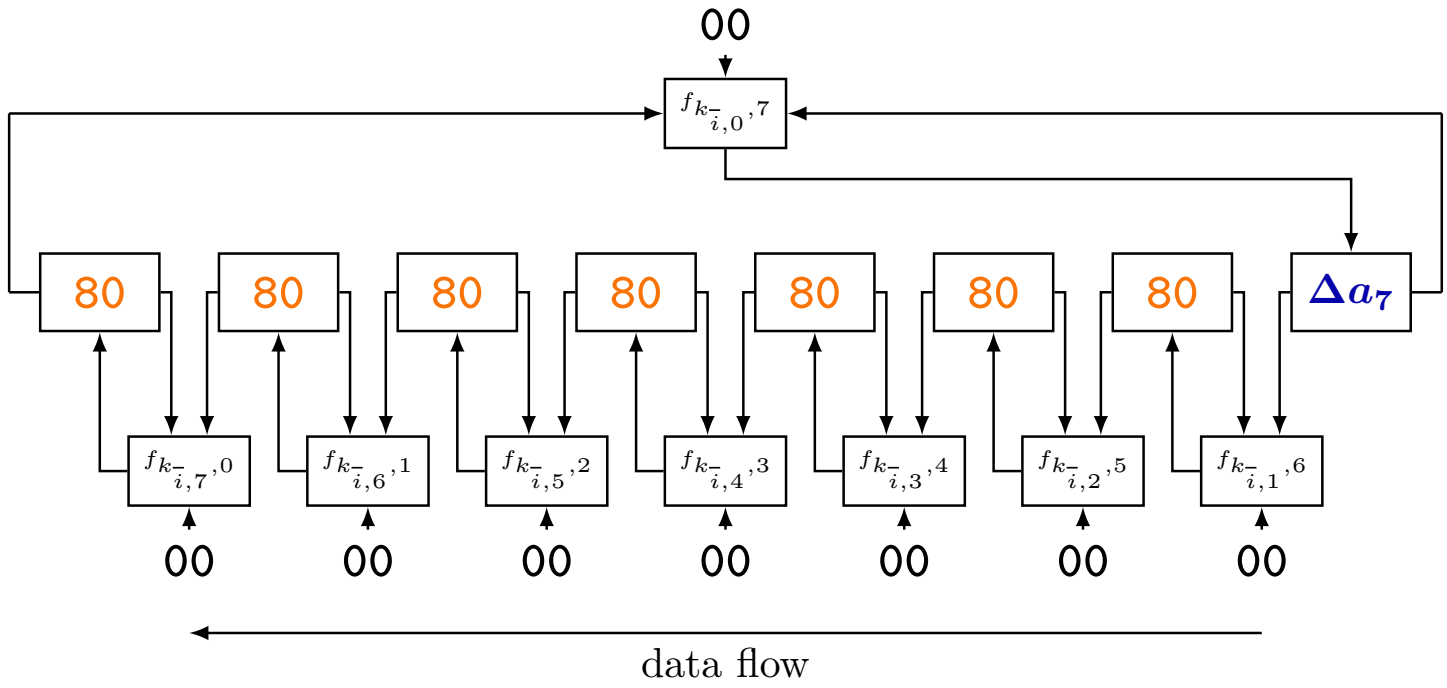
- The non-linear update function f:

$$f_{k,c}(x, y, m) = \begin{cases} y + m + (\neg(x + c)) \lll 1 & \text{if } k = 1 \\ y + m - (\neg(x + c)) \lll 7 & \text{otherwise.} \end{cases}$$

- Difference propagates with **probability 1** to the full state!

Bitwise Key Recovery

- Difference propagation after processing $m_{8i}, \ldots, m_{8i+7}, m_{8i+8}$:



- Possible output differences for the XOR-linearisation of f:

$$\Delta a_7 = \begin{cases} 81 = 80 \oplus 01 = 80 \oplus (80 \lll 1) & \text{if } k_{\bar{i},0} = 1 \\ C0 = 80 \oplus 40 = 80 \oplus (80 \lll 7) & \text{if } k_{\bar{i},0} = 0 \end{cases}$$

- Equal behaviour of $\texttt{lsb}$ for $\oplus$ and $+$: $\texttt{lsb}(k_{\bar{i}}) = k_{\bar{i},0} = \texttt{lsb}(\Delta a_7)$.

Bitwise Key Recovery

Full Key Recovery

In **96+1** queries with 144-byte **chosen-plaintexts**.

Can we do better?

Improving Bitwise Key Recovery

- Setting $\Delta m_{8i-8} = 80$ (eight steps earlier as bitwise attack) gives:

$i = 17, \ldots, 6$	a_0	a_1	a_2	a_3	a_4	a_5	a_6	a_7
$\ldots$	$\ldots$	$\ldots$	$\ldots$	$\ldots$	$\ldots$	$\ldots$	$\ldots$	$\ldots$
m_{8i-9}	00	00	00	00	00	00	00	00
m_{8i-8}	00	00	00	00	00	00	00	80
$\ldots$	$\ldots$	$\ldots$	$\ldots$	$\ldots$	$\ldots$	$\ldots$	$\ldots$	$\ldots$
m_{8i-1}	80	80	80	80	80	80	80	80
m_{8i}	80	80	80	80	80	80	80	Δa_7
m_{8i+1}	80	80	80	80	80	80	Δa_6	Δa_7
$\ldots$	$\ldots$	$\ldots$	$\ldots$	$\ldots$	$\ldots$	$\ldots$	$\ldots$	$\ldots$
m_{8i+7}	Δa_0	Δa_1	Δa_2	Δa_3	Δa_4	Δa_5	Δa_6	Δa_7

- Analysing the XOR-linearisation of f shows ...

Improving Bitwise Key Recovery

- Setting $\Delta m_{8i-8} = 80$ (eight steps earlier as bitwise attack) gives:

$i = 17, \ldots, 6$	a_0	a_1	a_2	a_3	a_4	a_5	a_6	a_7
$\ldots$	$\ldots$	$\ldots$	$\ldots$	$\ldots$	$\ldots$	$\ldots$	$\ldots$	$\ldots$
m_{8i-9}	00	00	00	00	00	00	00	00
m_{8i-8}	00	00	00	00	00	00	00	80
$\ldots$	$\ldots$	$\ldots$	$\ldots$	$\ldots$	$\ldots$	$\ldots$	$\ldots$	$\ldots$
m_{8i-1}	80	80	80	80	80	80	80	80
m_{8i}	80	80	80	80	80	80	80	Δa_7
m_{8i+1}	80	80	80	80	80	80	Δa_6	Δa_7
$\ldots$	$\ldots$	$\ldots$	$\ldots$	$\ldots$	$\ldots$	$\ldots$	$\ldots$	$\ldots$
m_{8i+7}	Δa_0	Δa_1	Δa_2	Δa_3	Δa_4	Δa_5	Δa_6	Δa_7

- Analysing the XOR-linearisation of f shows ...

Bytewise Key Recovery

- Key bits can be recovered iteratively

$$
\begin{aligned}
k_{\bar{i},0} &= \mathtt{lsb}(\Delta a_7) & k_{\bar{i},4} &= \mathtt{lsb}(\Delta a_3) \oplus \mathtt{lsb}(\Delta a_4) \\
k_{\bar{i},1} &= \mathtt{lsb}(\Delta a_6) \oplus \mathtt{lsb}(\Delta a_7) & k_{\bar{i},5} &= \mathtt{lsb}(\Delta a_2) \oplus \mathtt{lsb}(\Delta a_3) \\
k_{\bar{i},2} &= \mathtt{lsb}(\Delta a_5) \oplus \mathtt{lsb}(\Delta a_6) & k_{\bar{i},6} &= \mathtt{lsb}(\Delta a_1) \oplus \mathtt{lsb}(\Delta a_2) \\
k_{\bar{i},3} &= \mathtt{lsb}(\Delta a_4) \oplus \mathtt{lsb}(\Delta a_5) & k_{\bar{i},7} &= \mathtt{lsb}(\Delta a_0) \oplus \mathtt{lsb}(\Delta a_1)
\end{aligned}
$$

  for all $i = 17, \ldots, 6$ and $\bar{i} = i \bmod 12$.

- **Conclusion:**

  Setting $\Delta m_{8i-8} = 80$ leaks complete key byte $k_{\bar{i}}$.

Bytewise Key Recovery

- Key bits can be recovered iteratively

$$
\begin{aligned}
k_{\bar{i},0} &= \mathtt{lsb}(\Delta a_7) & k_{\bar{i},4} &= \mathtt{lsb}(\Delta a_3) \oplus \mathtt{lsb}(\Delta a_4) \\
k_{\bar{i},1} &= \mathtt{lsb}(\Delta a_6) \oplus \mathtt{lsb}(\Delta a_7) & k_{\bar{i},5} &= \mathtt{lsb}(\Delta a_2) \oplus \mathtt{lsb}(\Delta a_3) \\
k_{\bar{i},2} &= \mathtt{lsb}(\Delta a_5) \oplus \mathtt{lsb}(\Delta a_6) & k_{\bar{i},6} &= \mathtt{lsb}(\Delta a_1) \oplus \mathtt{lsb}(\Delta a_2) \\
k_{\bar{i},3} &= \mathtt{lsb}(\Delta a_4) \oplus \mathtt{lsb}(\Delta a_5) & k_{\bar{i},7} &= \mathtt{lsb}(\Delta a_0) \oplus \mathtt{lsb}(\Delta a_1)
\end{aligned}
$$

  for all $i = 17, \ldots, 6$ and $\bar{i} = i \bmod 12$.

- **Conclusion:**

  Setting $\Delta m_{8i-8} = 80$ leaks complete key byte $k_{\bar{i}}$.

Bytewise Key Recovery

Full Key Recovery

In **12+1** queries with 144-byte **chosen-plaintexts**.

Attack #2

Known-Plaintext Key Recovery

Prerequisites

- Two 144-byte messages

$$m = x \parallel y \text{ and } m' = x \parallel y'$$

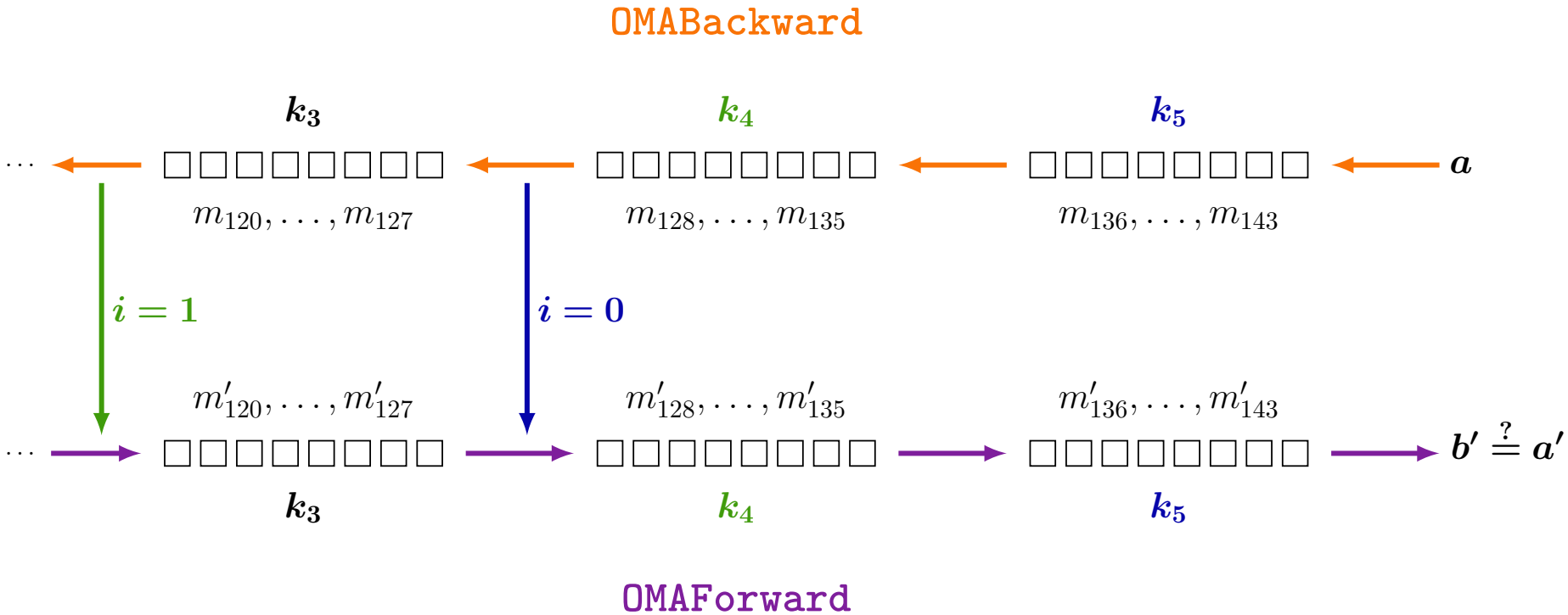
with $|y| = |y'| = r$ bytes and $y \neq y'$.

- Corresponding digests

$$a = \mathcal{O}(m) \text{ and } a' = \mathcal{O}(m')$$

with $\mathcal{O}$ being an oracle for the OMADigest using the key k.

Known-Plaintext Key Recovery



- For $i = 0, \ldots, 11$, set $r = 8i + 16$, guess $k_{17-i \bmod 12}$, and fix $k_{16-i \bmod 12} = 00$ (note: key byte has no effect on processing of m).
- Compute: $b' = \texttt{OMAForward}(\texttt{OMABackward}(a, m, k, r), m', k, r)$.
- Check: $b' = a'$.
- If so, guess for $k_{17-i \bmod 12}$ is saved as a candidate.

Known-Plaintext Key Recovery

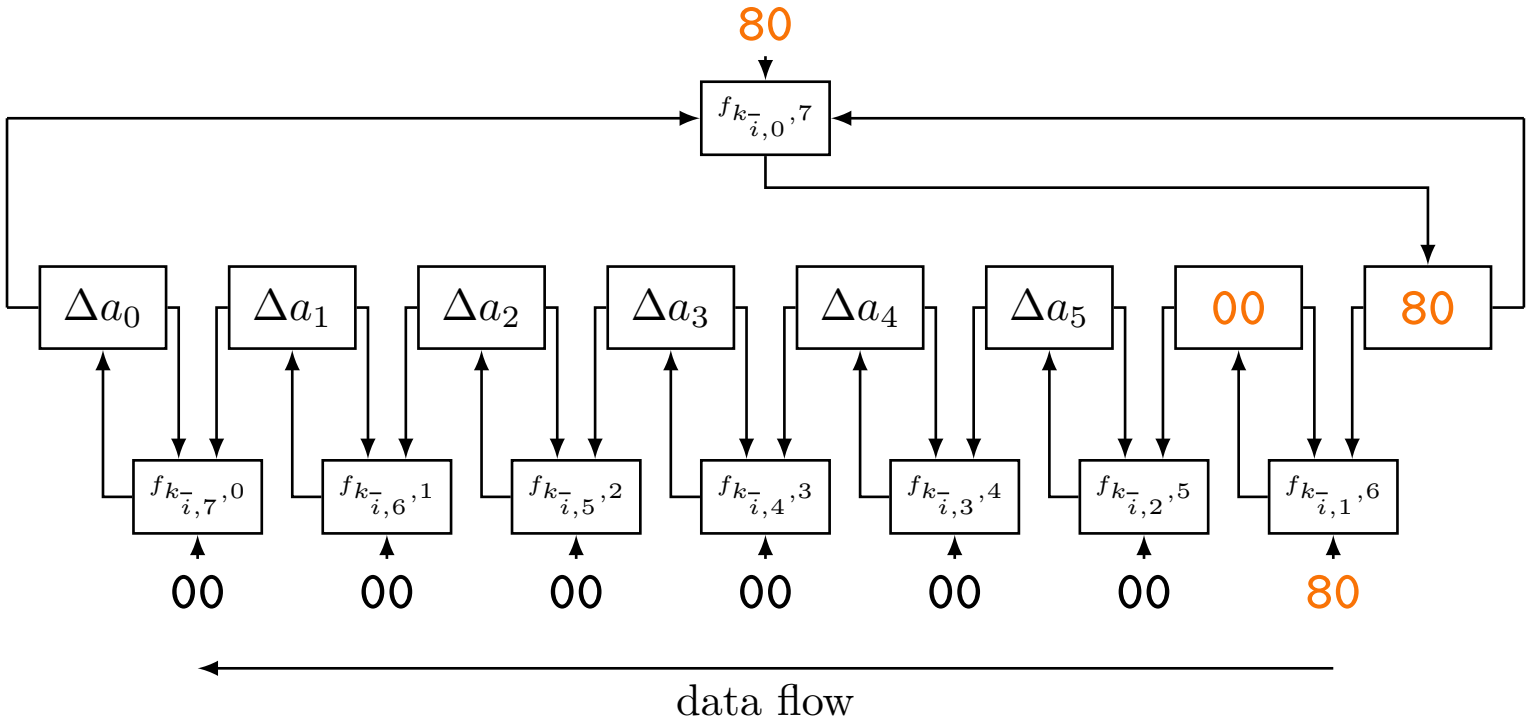
Full Key Recovery

- In **24** queries of 144-byte **known-plaintexts with common prefix**.
- In **12 + 1** queries of 144-byte **chosen plaintexts**.

Attack #3

Forgery Attacks

- Injecting XOR-differences $\Delta m_{8i+j} = 80$ and $\Delta m_{8i+j+1} = 80$



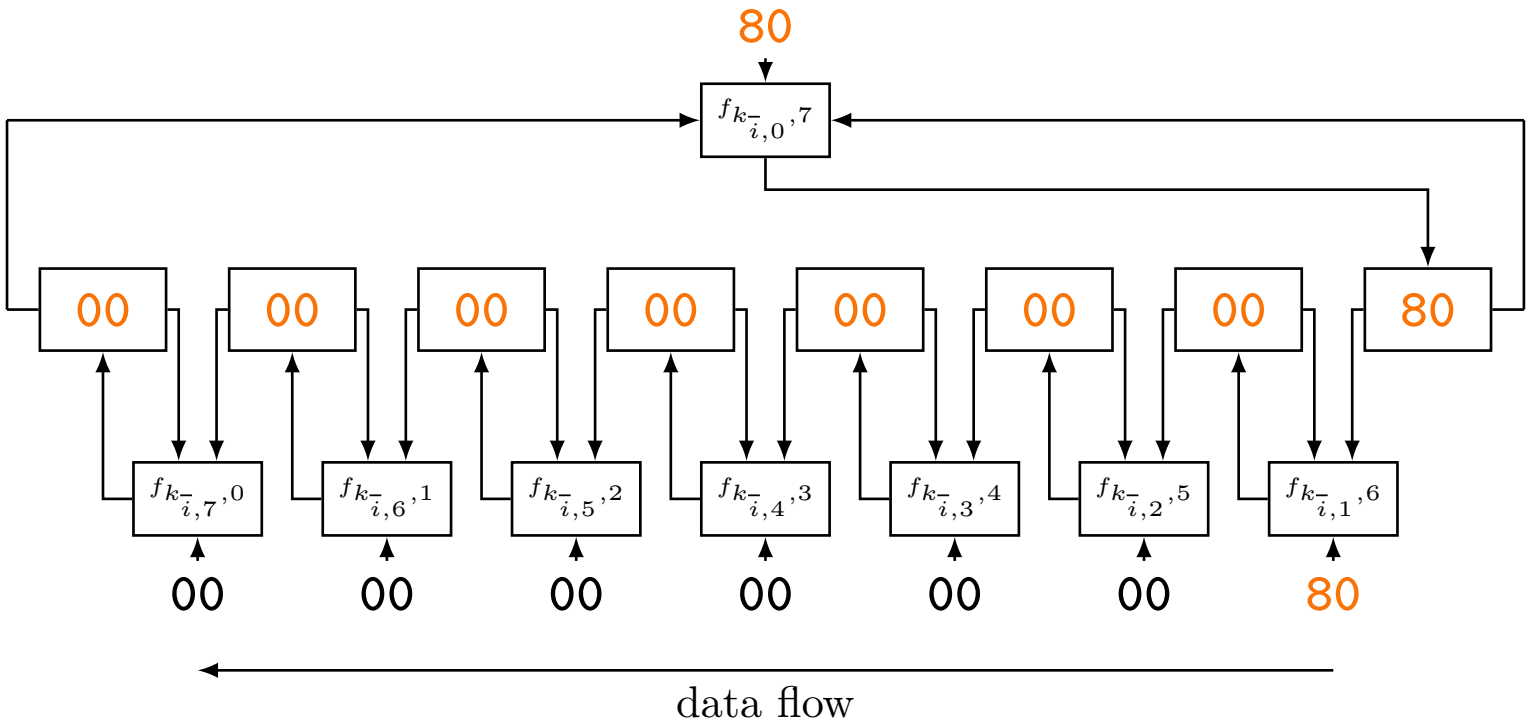
for $i = 0, \ldots, 17$, $\bar{i} = i \bmod 12$, and $j = 0, \ldots, 7$ (here: $j = 0$).

- The non-linear update function f:

$$f_{k,c}(x, y, m) = \begin{cases} y + m + (\neg(x + c)) \lll 1 & \text{if } k = 1 \\ y + m - (\neg(x + c)) \lll 7 & \text{otherwise.} \end{cases}$$

Forgery Attacks

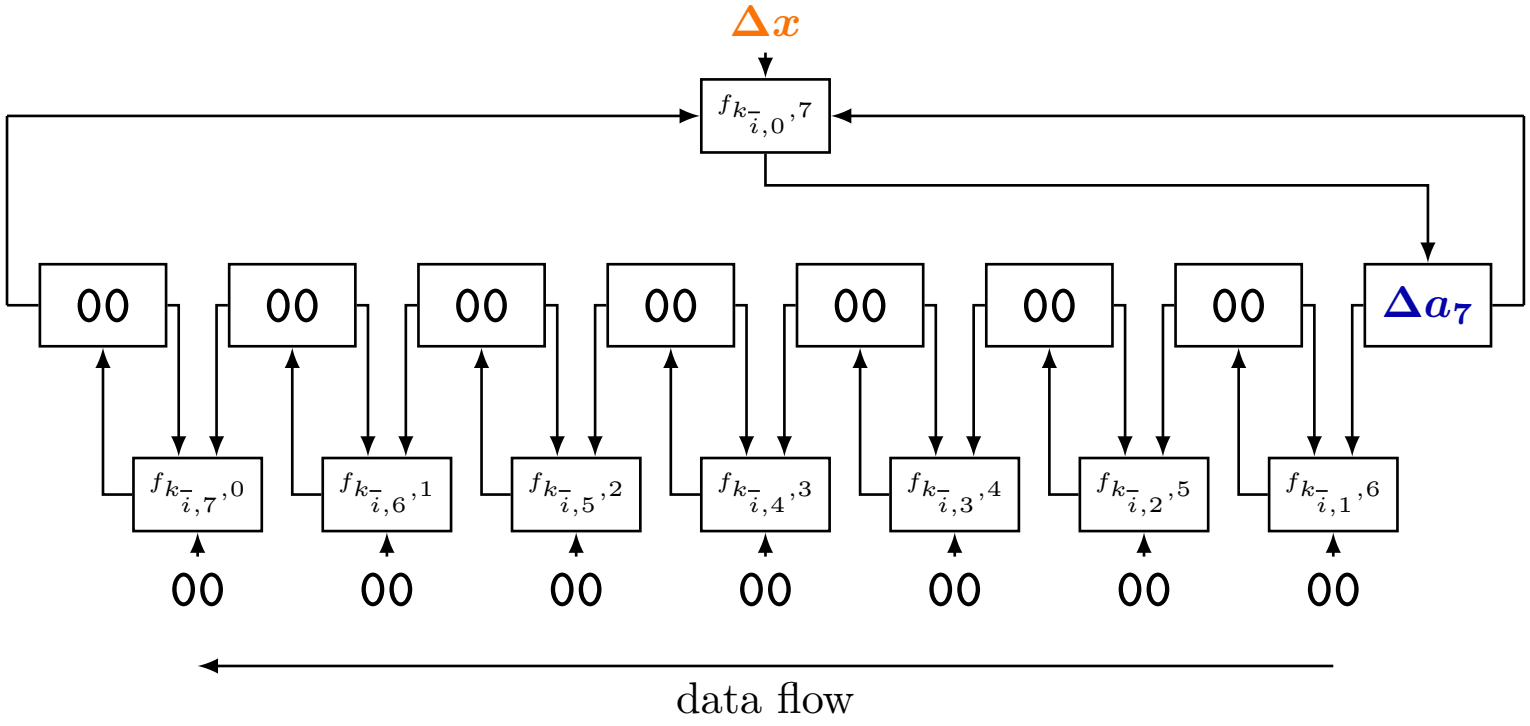
- Difference propagation after processing $m_{8i+j}, \ldots, m_{8i+j+7}$:



- No further propagation, stationary difference $\Delta a_7 = 80$.

Forgery Attacks

- Difference propagation after processing $m_{8i+j}, \ldots, m_{8i+j+7}, m_{8i+j+8}$:



- Inject XOR-difference $\Delta m_{8i+j+8} = \Delta x$ s.t. $\Delta a_7 = 00 \Rightarrow$ **forgery**!
- How do we choose Δx?

From Forgeries ...

- Options for Δx:

$k_{\overline{i+1},j} = 0$	Δx	C0	40						
	p	1/2	1/2						
$k_{\overline{i+1},j} = 1$	Δx	01	03	07	0F	1F	3F	7F	FF
	p	1/2	1/4	1/8	1/16	1/32	1/64	1/128	1/128

- Using $(\Delta m_{8i+j}, \Delta m_{8i+j+1}, \Delta m_{8i+j+8}) = (80, 80, \Delta x)$ with

$$\Delta x \in \{\texttt{C0}, 40, 01\}$$

has **probability $\approx 1/4$** to create a forgery.

From Forgeries ...

- Options for Δx:

$k_{\overline{i+1},j} = 0$	Δx	C0	40						
	p	1/2	1/2						
$k_{\overline{i+1},j} = 1$	Δx	01	03	07	0F	1F	3F	7F	FF
	p	1/2	1/4	1/8	1/16	1/32	1/64	1/128	1/128

- Using $(\Delta m_{8i+j}, \Delta m_{8i+j+1}, \Delta m_{8i+j+8}) = (80, 80, \Delta x)$ with

$$\Delta x \in \{\texttt{C0}, 40, 01\}$$

has **probability $\approx 1/4$** to create a forgery.

... to Key Recovery

1. Test $(\Delta m_{8i+j}, \Delta m_{8i+j+1}, \Delta m_{8i+j+8}) = (80, 80, \texttt{C0})$. Forgery?
   - Yes: $k_{i+1 \bmod 12,j} = 0$.
   - No: Continue.
2. Test $(\Delta m_{8i+j}, \Delta m_{8i+j+1}, \Delta m_{8i+j+8}) = (80, 80, 40)$. Forgery?
   - Yes: $k_{i+1 \bmod 12,j} = 0$.
   - No: $k_{i+1 \bmod 12,j} = 1$.

... to Key Recovery

1. Test $(\Delta m_{8i+j}, \Delta m_{8i+j+1}, \Delta m_{8i+j+8}) = (80, 80, \texttt{C0})$. Forgery?
   - Yes: $k_{i+1 \bmod 12,j} = 0$.
   - No: Continue.
2. Test $(\Delta m_{8i+j}, \Delta m_{8i+j+1}, \Delta m_{8i+j+8}) = (80, 80, \texttt{40})$. Forgery?
   - Yes: $k_{i+1 \bmod 12,j} = 0$.
   - No: $k_{i+1 \bmod 12,j} = 1$.

Forgery-based Key Recovery

Summary

- Full key recovery in **168 queries** (on average).
- Works with **chosen-plaintexts** and with **chosen-ciphertexts**.
  (due to stream cipher encryption)
- Key bits can be recovered in **arbitrary order**.
  (unlike as in attacks #1 and #2)
- **No restrictions** on the message size.

Forgery-based Key Recovery

Summary

- Full key recovery in **168 queries** (on average).
- Works with **chosen-plaintexts** and with **chosen-ciphertexts**. (due to stream cipher encryption)
- Key bits can be recovered in **arbitrary order**. (unlike as in attacks #1 and #2)
- **No restrictions** on the message size.

Forgery-based Key Recovery

Summary

- Full key recovery in **168 queries** (on average).
- Works with **chosen-plaintexts** and with **chosen-ciphertexts**. (due to stream cipher encryption)
- Key bits can be recovered in **arbitrary order**. (unlike as in attacks #1 and #2)
- **No restrictions** on the message size.

Forgery-based Key Recovery

Summary

- Full key recovery in **168 queries** (on average).
- Works with **chosen-plaintexts** and with **chosen-ciphertexts**. (due to stream cipher encryption)
- Key bits can be recovered in **arbitrary order**. (unlike as in attacks #1 and #2)
- **No restrictions** on the message size.

Conclusion

Overview on Digest Attacks

Attack	Type	B	Queries	Complexity	Oracle
#1	**CP**	**1**	**13**	$\mathbf{2^{3.58}}$	**Tag-generation**
	CP	2	7	$2^{10.58}$	Tag-generation
	CP	3	5	$2^{18.00}$	Tag-generation
	CP	4	4	$2^{25.58}$	Tag-generation
	CP	5	4	$2^{33.58}$	Tag-generation
	CP	6	3	$2^{41.00}$	Tag-generation
#2	**KP+ / CP**	**1**	**24/13**	$\mathbf{2^{10.58}}$	**Tag-generation**
	KP+ / CP	2	12 / 7	$2^{17.58}$	Tag-generation
	KP+ / CP	3	8 / 5	$2^{25.00}$	Tag-generation
	KP+ / CP	4	6 / 4	$2^{32.58}$	Tag-generation
	KP+ / CP	5	6 / 4	$2^{40.32}$	Tag-generation
	KP+ / CP	6	4 / 3	$2^{48.58}$	Tag-generation
#3	**Forgeries (CP / CC, XOR)**	—	**$\approx$ 168**	**$\approx$ 168**	**Tag-verification**
	Forgeries (CP, Additive)	—	$\approx$ 144	$\approx$ 144	Tag-verification

- B: time-query trade-off parameter.
- KP+: known-plaintext with common prefix.
- CP: chosen-plaintext.
- CC: chosen-cipertext.

Fin

We think:

- OSGP's cryptographic scheme offers **no protection** whatsoever. (assuming it is implemented as in the specification)
- Secure communication in OSGP highly doubtful as long as any of RC4, EN14908 or OMADigest is used.

Thank you!

Fin

We think:

- OSGP's cryptographic scheme offers **no protection** whatsoever. (assuming it is implemented as in the specification)
- Secure communication in OSGP highly doubtful as long as any of RC4, EN14908 or OMADigest is used.

Thank you!