Symmetric-key Authenticated Key Exchange (SAKE) with Perfect Forward Secrecy

Gildas Avoine[1,2] Sébastien Canard[3] **Loïc Ferreira**[3,*]

[1] Institut Universitaire de France

[2] Univ Rennes, INSA Rennes, CNRS, IRISA, France

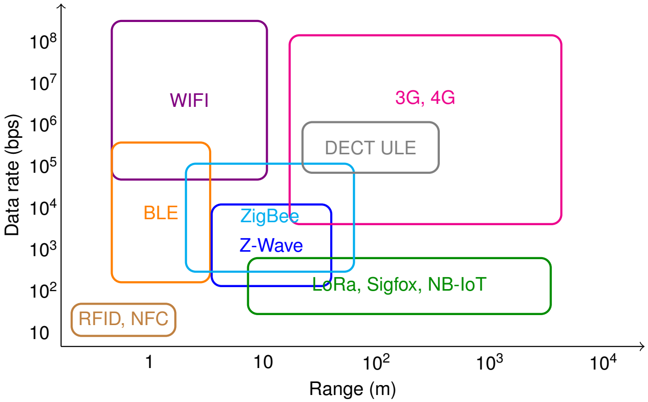
[3] Orange Labs, Applied Cryptography Group, Caen, France

CT-RSA 2020
February 24-28, 2020

* Work done while at IRISA

Context

- 125 billion connected IoT devices by 2030 [IHS17]



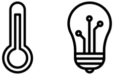
[IHS17] IHS Markit. *The Internet of Things: a movement, not a market*. 2017.

Context

- Many applications

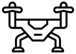
Smart city

Connected home



eHealth

FANET (drone)



Vehicular

Industrial

- Communication security ⇒ authenticated key exchange

Context

- Low-resource devices $\left\{\begin{array}{l}\text{memory}\\ \text{computation}\\ \text{energy}\end{array}\right.$
- Protocols based on asymmetric algorithms are too heavy for very constrained devices.
- Trade-off (very often): security vs. efficiency.

ANDY GREENBERG SECURITY 07.21.15 06:00 AM

Hackers Remotely Kill a Jeep on the Highway—With Me in It

CBS NEWS / *November 8, 2018, 7:28 AM*

How medical devices like pacemakers and insulin pumps can be hacked

Hackable implanted medical devices could cause deaths, researchers say

Thu 9 Aug 2018 23.36 BST

Watch a drone hack a room full of smart lightbulbs from outside the window

S.O.S.

Nov 3, 2016, 6:12am EDT

Goals and constraints

- 2-party protocol
- Key agreement
- Mutual authentication
- Forward secrecy

$\left.\begin{array}{l} \\ \\ \\ \\ \end{array}\right\}$ ~~(EC)DH + signature~~ Symmetric-key only

$\Rightarrow$ Symmetric-key Authenticated Key Exchange

Description of the protocol

A		B
$(K, K'_{j+1}, K'_j, K'_{j-1})$		(K, K')
	$\xrightarrow{A \| r_A}$	
		$\tau_B \leftarrow \mathrm{MAC}(K', B\|A\|r_B\|r_A)$
	$\xleftarrow{r_B \| \tau_B}$	
case $K' = K'_{j+1}$: upd enter case K'_j		
case $K' = K'_{j-1}$: $\varepsilon = 1$		
case $K' = K'_j$: kdf; upd $\varepsilon = 0$		
$\tau_A \leftarrow \mathrm{MAC}(K', \varepsilon\|A\|B\|r_A\|r_B)$		
	$\xrightarrow{\varepsilon \| \tau_A}$	
		case $\varepsilon = 1$: upd
		kdf; upd
	$\xleftarrow{\tau'_B}$	
case $\varepsilon = 1$: kdf; upd		
	$\xrightarrow{\tau'_A}$	

- kdf: $sk \leftarrow \mathrm{KDF}(K, r_A, r_B)$
- upd:
  - $K \leftarrow \mathrm{H}(K)$
  - $K' \leftarrow \mathrm{H}(K')$

Description of the protocol

✓ party authentication

A		B
$(K, K'_{j+1}, K'_j, K'_{j-1})$		(K, K')
	$\xrightarrow{A\|r_A}$	
		$\tau_B \leftarrow \mathrm{MAC}(K', B\|A\|r_B\|r_A)$
	$\xleftarrow{r_B\|\tau_B}$	
case $K' = K'_{j+1}$: upd; enter case K'_j		
case $K' = K'_{j-1}$: $\varepsilon = 1$		
case $K' = K'_j$: kdf; upd; $\varepsilon = 0$		
$\tau_A \leftarrow \mathrm{MAC}(K', \varepsilon\|A\|B\|r_A\|r_B)$		
	$\xrightarrow{\varepsilon\|\tau_A}$	
		case $\varepsilon = 1$: upd
		kdf; upd
	$\xleftarrow{\tau'_B}$	
case $\varepsilon = 1$: kdf; upd		
	$\xrightarrow{\tau'_A}$	

- kdf: $sk \leftarrow \mathrm{KDF}(K, r_A, r_B)$
- upd:
  - $K \leftarrow \mathrm{H}(K)$
  - $K' \leftarrow \mathrm{H}(K')$

Description of the protocol

✓ party authentication

✓ session key derivation

A		B
$(K, K'_{j+1}, K'_j, K'_{j-1})$		(K, K')
	$\xrightarrow{A \| r_A}$	
		$\tau_B \leftarrow \mathrm{MAC}(K', B\|A\|r_B\|r_A)$
	$\xleftarrow{r_B \| \tau_B}$	
case $K' = K'_{j+1}$: upd enter case K'_j		
case $K' = K'_{j-1}$: $\varepsilon = 1$		
case $K' = K'_j$: kdf; upd $\varepsilon = 0$		
$\tau_A \leftarrow \mathrm{MAC}(K', \varepsilon\|A\|B\|r_A\|r_B)$		
	$\xrightarrow{\varepsilon \| \tau_A}$	
		case $\varepsilon = 1$: upd
		kdf; upd
	$\xleftarrow{\tau'_B}$	
case $\varepsilon = 1$: kdf; upd		
	$\xrightarrow{\tau'_A}$	

- kdf: $sk \leftarrow \mathrm{KDF}(K, r_A, r_B)$
- upd:
  - $K \leftarrow \mathrm{H}(K)$
  - $K' \leftarrow \mathrm{H}(K')$

Description of the protocol

✓ party authentication

✓ session key derivation

master key update

A		B
$(K, K'_{j+1}, K'_j, K'_{j-1})$		(K, K')
	$\xrightarrow{A \| r_A}$	
		$\tau_B \leftarrow \mathrm{MAC}(K', B\|A\|r_B\|r_A)$
	$\xleftarrow{r_B \| \tau_B}$	
case $K' = K'_{j+1}$: upd; enter case K'_j		
case $K' = K'_{j-1}$: $\varepsilon = 1$		
case $K' = K'_j$: kdf; upd; $\varepsilon = 0$		
$\tau_A \leftarrow \mathrm{MAC}(K', \varepsilon\|A\|B\|r_A\|r_B)$		
	$\xrightarrow{\varepsilon \| \tau_A}$	
		case $\varepsilon = 1$: upd
		kdf; upd
	$\xleftarrow{\tau'_B}$	
case $\varepsilon = 1$: kdf; upd		
	$\xrightarrow{\tau'_A}$	

- kdf: $sk \leftarrow \mathrm{KDF}(K, r_A, r_B)$
- upd:
  - $K \leftarrow \mathrm{H}(K)$
  - $K' \leftarrow \mathrm{H}(K')$

Description of the protocol

✓ party authentication

✓ session key derivation

master key update

A $(K, K'_{j+1}, K'_j, K'_{j-1})$

B (K, K')

$\xrightarrow{A \| r_A}$

$\tau_B \leftarrow \mathrm{MAC}(K', B \| A \| r_B \| r_A)$

$\xleftarrow{r_B \| \tau_B}$

case $K' = K'_{j+1}$:
upd
enter case K'_j

case $K' = K'_{j-1}$:
$\varepsilon = 1$

case $K' = K'_j$:
kdf; upd
$\varepsilon = 0$

$\tau_A \leftarrow \mathrm{MAC}(K', \varepsilon \| A \| B \| r_A \| r_B)$

$\xrightarrow{\varepsilon \| \tau_A}$

case $\varepsilon = 1$:
upd

kdf; upd

$\xleftarrow{\tau'_B}$

case $\varepsilon = 1$:
kdf; upd

$\xrightarrow{\tau'_A}$

- kdf: $sk \leftarrow \mathrm{KDF}(K, r_A, r_B)$
- upd:
  - $K \leftarrow \mathrm{H}(K)$
  - $K' \leftarrow \mathrm{H}(K')$

Description of the protocol

- ✓ party authentication
- ✓ session key derivation
- master key update
- synchronisation

A: $(K, K'_{j+1}, K'_j, K'_{j-1})$

B: (K, K')

$A \xrightarrow{A \| r_A} B$

B: $\tau_B \leftarrow \mathrm{MAC}(K', B\|A\|r_B\|r_A)$

$A \xleftarrow{r_B \| \tau_B} B$

A:

case $K' = K'_{j+1}$:
  upd
  enter case K'_j

case $K' = K'_{j-1}$:
  $\varepsilon = 1$

case $K' = K'_j$:
  kdf; upd
  $\varepsilon = 0$

$\tau_A \leftarrow \mathrm{MAC}(K', \varepsilon\|A\|B\|r_A\|r_B)$

$A \xrightarrow{\varepsilon \| \tau_A} B$

B:

case $\varepsilon = 1$:
  upd

kdf; upd

$A \xleftarrow{\tau'_B} B$

A:

case $\varepsilon = 1$:
  kdf; upd

$A \xrightarrow{\tau'_A} B$

- kdf: $sk \leftarrow \mathrm{KDF}(K, r_A, r_B)$
- upd:
  - $K \leftarrow \mathrm{H}(K)$
  - $K' \leftarrow \mathrm{H}(K')$

Description of the protocol: key evolution

$$
\begin{tikzcd}
\text{authentication} & K'_0 \arrow[r, "\mathrm{H}"] & K'_1 \arrow[r] & K'_2 \arrow[r] & K'_3 \arrow[r] & \cdots \\
\text{derivation} & K_0 \arrow[r, "\mathrm{H}"] \arrow[d, "\mathrm{KDF}"] & K_1 \arrow[r] \arrow[d] & K_2 \arrow[r] \arrow[d] & K_3 \arrow[r] \arrow[d] & \cdots \\
 & sk_0 & sk_1 & sk_2 & sk_3 &
\end{tikzcd}
$$

Description of the protocol

- ✓ party authentication
- ✓ session key derivation
- master key update
- synchronisation

A		B
$(K, K'_{j+1}, K'_j, K'_{j-1})$		(K, K')
	$\xrightarrow{A \| r_A}$	
		$\tau_B \leftarrow \mathrm{MAC}(K', B\|A\|r_B\|r_A)$
	$\xleftarrow{r_B \| \tau_B}$	
case $K' = K'_{j+1}$: upd enter case K'_j		
case $K' = K'_{j-1}$: $\varepsilon = 1$		
case $K' = K'_j$: kdf; upd $\varepsilon = 0$		
$\tau_A \leftarrow \mathrm{MAC}(K', \varepsilon\|A\|B\|r_A\|r_B)$		
	$\xrightarrow{\varepsilon \| \tau_A}$	
		case $\varepsilon = 1$: upd
		kdf; upd
	$\xleftarrow{\tau'_B}$	
case $\varepsilon = 1$: kdf; upd		
	$\xrightarrow{\tau'_A}$	

- kdf: $sk \leftarrow \mathrm{KDF}(K, r_A, r_B)$
- upd:
  - $K \leftarrow \mathrm{H}(K)$
  - $K' \leftarrow \mathrm{H}(K')$

Description of the protocol

- ✓ party authentication
- ✓ session key derivation
- master key update
- synchronisation

A		B
$(K, K'_{j+1}, K'_j, K'_{j-1})$		(K, K')
	$\xrightarrow{A \| r_A}$	
		$\tau_B \leftarrow \mathrm{MAC}(K', B\|A\|r_B\|r_A)$
	$\xleftarrow{r_B \| \tau_B}$	
case $K' = K'_{j+1}$: upd; enter case K'_j		
case $K' = K'_{j-1}$: $\varepsilon = 1$		
case $K' = K'_j$: kdf; upd; $\varepsilon = 0$		
$\tau_A \leftarrow \mathrm{MAC}(K', \varepsilon\|A\|B\|r_A\|r_B)$		
	$\xrightarrow{\varepsilon \| \tau_A}$	
		case $\varepsilon = 1$: upd
		kdf; upd
	$\xleftarrow{\tau'_B}$	
case $\varepsilon = 1$: kdf; upd		
	$\xrightarrow{\tau'_A}$	

- kdf: $sk \leftarrow \mathrm{KDF}(K, r_A, r_B)$
- upd:
  - $K \leftarrow \mathrm{H}(K)$
  - $K' \leftarrow \mathrm{H}(K')$

Description of the protocol

- ✓ party authentication
- ✓ session key derivation
- master key update
- synchronisation

A		B
$(K, K'_{j+1}, K'_j, K'_{j-1})$		(K, K')
	$\xrightarrow{A\Vert r_A}$	
		$\tau_B \leftarrow \mathrm{MAC}(K', B\Vert A\Vert r_B\Vert r_A)$
	$\xleftarrow{r_B\Vert \tau_B}$	
case $K' = K'_{j+1}$: upd enter case K'_j		
case $K' = K'_{j-1}$: $\varepsilon = 1$		
case $K' = K'_j$: kdf; upd $\varepsilon = 0$		
$\tau_A \leftarrow \mathrm{MAC}(K', \varepsilon\Vert A\Vert B\Vert r_A\Vert r_B)$		
	$\xrightarrow{\varepsilon\Vert \tau_A}$	
		case $\varepsilon = 1$: upd
		kdf; upd
	$\xleftarrow{\tau'_B}$	
case $\varepsilon = 1$: kdf; upd		
	$\xrightarrow{\tau'_A}$	

- kdf: $sk \leftarrow \mathrm{KDF}(K, r_A, r_B)$
- upd:
  - $K \leftarrow \mathrm{H}(K)$
  - $K' \leftarrow \mathrm{H}(K')$

Description of the protocol

✓ party authentication
✓ session key derivation
master key update
synchronisation
} ✓ forward secrecy

A		B
$(K, K'_{j+1}, K'_j, K'_{j-1})$		(K, K')
	$\xrightarrow{A\|r_A}$	
		$\tau_B \leftarrow \mathrm{MAC}(K', B\|A\|r_B\|r_A)$
	$\xleftarrow{r_B\|\tau_B}$	
case $K' = K'_{j+1}$: upd; enter case K'_j		
case $K' = K'_{j-1}$: $\varepsilon = 1$		
case $K' = K'_j$: kdf; upd; $\varepsilon = 0$		
$\tau_A \leftarrow \mathrm{MAC}(K', \varepsilon\|A\|B\|r_A\|r_B)$		
	$\xrightarrow{\varepsilon\|\tau_A}$	
		case $\varepsilon = 1$: upd
		kdf; upd
	$\xleftarrow{\tau'_B}$	
case $\varepsilon = 1$: kdf; upd		
	$\xrightarrow{\tau'_A}$	

- kdf: $sk \leftarrow \mathrm{KDF}(K, r_A, r_B)$
- upd:
  - $K \leftarrow \mathrm{H}(K)$
  - $K' \leftarrow \mathrm{H}(K')$

Description of the protocol

✓ party authentication

✓ session key derivation

master key update
synchronisation

} ✓ forward secrecy

A $(K, K'_{j+1}, K'_j, K'_{j-1})$

B (K, K')

$\xrightarrow{A \| r_A}$

$\tau_B \leftarrow \mathrm{MAC}(K', B\|A\|r_B\|r_A)$

$\xleftarrow{r_B \| \tau_B}$

case $K' = K'_{j+1}$:
upd
enter case K'_j

case $K' = K'_{j-1}$:
$\varepsilon = 1$

case $K' = K'_j$:
kdf; upd
$\varepsilon = 0$

$\tau_A \leftarrow \mathrm{MAC}(K', \varepsilon\|A\|B\|r_A\|r_B)$

$\xrightarrow{\varepsilon \| \tau_A}$

case $\varepsilon = 1$:
upd

kdf; upd

$\xleftarrow{\tau'_B}$

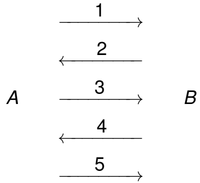
case $\varepsilon = 1$:
kdf; upd

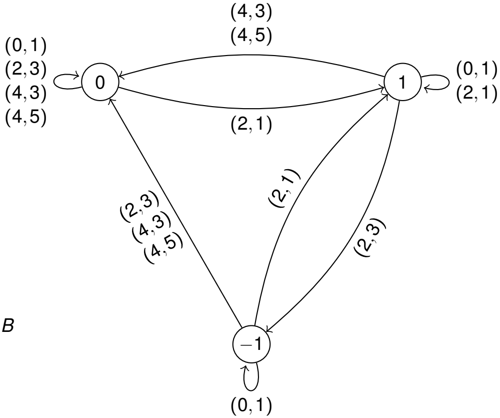
$\xrightarrow{\tau'_A}$

- kdf: $sk \leftarrow \mathrm{KDF}(K, r_A, r_B)$
- upd:
  - $K \leftarrow \mathrm{H}(K)$
  - $K' \leftarrow \mathrm{H}(K')$

Synchronisation issue

- Party A (or B) can only be one step ahead, one step behind, or synchronised ($\delta \in \{-1, 0, 1\}$).
- Whatever the initial synchronisation gap δ between A and B, after a complete and correct session A and B
  - share a fresh session key,
  - have updated their master keys,
  - are synchronised.





Security arguments

- Security model: Brzuska, Jacobsen, Stebila [BJS16].
  - The adversary can forward, alter, drop any message exchanged by honest parties, or insert new messages.
  - It can interact with several oracles (NewSession, Send, Corrupt, Reveal, Test).
- Two main security properties
  - Entity authentication.
  - Key indistinguishability.
- The adversarial model also captures forward secrecy.

$$\begin{aligned}
\mathrm{adv}^{\text{ent-auth}}_{SAKE}(\mathscr{A}) &\leq nq\Big((nq-1)2^{-\lambda} + (q+1)\mathrm{adv}^{\mathrm{prf}}_{\mathrm{H}}(\mathscr{B}) + 2\mathrm{adv}^{\text{suf-cma}}_{\mathrm{MAC}}(\mathscr{C})\Big) \\
\mathrm{adv}^{\text{key-ind}}_{SAKE}(\mathscr{A}) &\leq nq\Big((q-1)\mathrm{adv}^{\mathrm{prf}}_{\mathrm{H}}(\mathscr{B}) + \mathrm{adv}^{\mathrm{prf}}_{\mathrm{KDF}}(\mathscr{D})\Big) + \mathrm{adv}^{\text{ent-auth}}_{SAKE}(\mathscr{A})
\end{aligned}$$

where n is the number of parties, q the maximum number of instances (sessions) per party, λ the size of the pseudo-random values (r_A, r_B), and $\mathscr{B}$ is an adversary against the PRF-security of H, $\mathscr{C}$ an adversary against the SUF-CMA-security of MAC, and $\mathscr{D}$ an adversary against the PRF-security of KDF.

[BJS16] C. Brzuska, H. Jacobsen, and D. Stebila. "Safely Exporting Keys from Secure Channels: On the Security of EAP-TLS and TLS Key Exporters". In: *EUROCRYPT*. 2016.

SAKE-AM: “agressive mode” of SAKE

$$
\begin{array}{lcl}
\textbf{A} & & \textbf{B} \\
(K, K') & & (K, K'_{j+1}, K'_j, K'_{j-1}) \\
\tau_A \leftarrow \mathrm{MAC}(K', A \| B \| r_A) & & \\
& \xrightarrow{\quad A \| r_A \| \tau_A \quad} & \\
& & \text{case } K' = K'_{j+1}: \\
& & \quad \text{upd} \\
& & \quad \text{enter case } K'_j \\
& & \text{case } K' = K'_{j-1}: \\
& & \quad \varepsilon = 1 \\
& & \text{case } K' = K'_j: \\
& & \quad \text{kdf; upd} \\
& & \quad \varepsilon = 0 \\
& & \tau_B \leftarrow \mathrm{MAC}(K', \varepsilon \| B \| A \| r_B \| r_A) \\
& \xleftarrow{\quad \varepsilon \| r_B \| \tau_B \quad} & \\
\text{case } \varepsilon = 1: & & \\
\quad \text{upd} & & \\
\text{kdf; upd} & & \\
\tau'_A \leftarrow \mathrm{MAC}(K', A \| B \| r_A \| r_B) & & \\
& \xrightarrow{\quad \tau'_A \quad} & \\
& & \text{case } \varepsilon = 1: \\
& & \quad \text{kdf; upd} \\
& \xleftarrow{\quad \tau'_B \quad} &
\end{array}
$$

IoT setting

- SAKE and SAKE-AM together.
- Any party can initiate the protocol.
- The smallest amount of computations is done by the same party (end-device).

Advantageous for low-resource end-devices

End-device is responder (SAKE)

End-device [B]		**Back end** [A]
(K, K')		(K, K'_{j-1})
	$\xleftarrow{\quad A \Vert r_A \quad}$	
	$\xrightarrow{\quad r_B \Vert \tau_B \quad}$	
		compute ε
	$\xleftarrow{\quad \varepsilon \Vert \tau_A \quad}$	
	$\xrightarrow{\quad \tau'_B \quad}$	
	$\left[\xleftarrow{\quad \tau'_A \quad}\right]$	

End-device is initiator (SAKE-AM)

End-device [A]		**Back end** [B]
(K, K')		(K, K'_{j-1})
	$\xrightarrow{\quad A \Vert r_A \Vert \tau_A \quad}$	
		compute ε
	$\xleftarrow{\quad \varepsilon \Vert r_B \Vert \tau_B \quad}$	
	$\xrightarrow{\quad \tau'_A \quad}$	
	$\left[\xleftarrow{\quad \tau'_B \quad}\right]$	

IoT setting

- SAKE and SAKE-AM together.
- Any party can initiate the protocol.
- The smallest amount of computations is done by the same party (end-device).

Advantageous for low-resource end-devices

End-device is responder (SAKE)

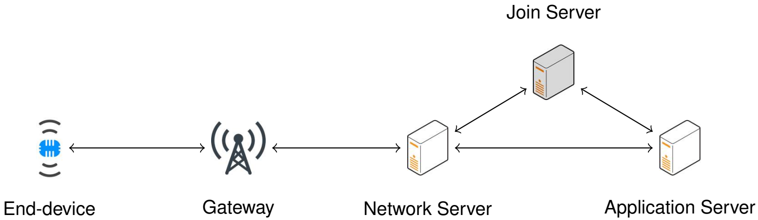
End-device [B]		**Back end** [A]
(K, K')		(K, K'_{j-1})
	$\xleftarrow{A \| r_A}$	
	$\xrightarrow{r_B \| \tau_B}$	
		compute ε
	$\xleftarrow{\varepsilon \| \tau_A}$	
	$\xrightarrow{\tau'_B}$	
	$\left[\xleftarrow{\tau'_A}\right]$	

End-device is initiator (SAKE-AM)

End-device [A]		**Back end** [B]
(K, K')		(K, K'_{j-1})
	$\xrightarrow{A \| r_A \| \tau_A}$	
		compute ε
	$\xleftarrow{\varepsilon \| r_B \| \tau_B}$	
	$\xrightarrow{\tau'_A}$	
	$\left[\xleftarrow{\tau'_B}\right]$	

Practical application

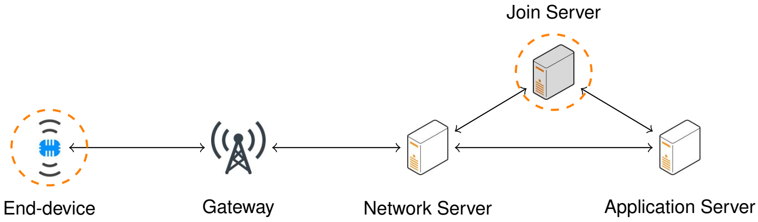
- LoRaWAN: security protocol for Low-Power Wide Area Networks (LPWAN). LoRaWAN $\simeq$ 3G/4G but optimised for IoT/M2M.
- Currently deployed in more than 100 countries worldwide (America, Europe, Africa, Asia).
- Promoted by LoRa Alliance (+500 members).
- Version 1.0: weak against likely practical attacks [AF18].
- Version 1.1: to be deployed.



[AF18] G. Avoine and L. Ferreira. "Rescuing LoRaWAN 1.0". In: *Financial Cryptography and Data Security*. 2018.

Practical application

- LoRaWAN: security protocol for Low-Power Wide Area Networks (LPWAN).
  LoRaWAN $\simeq$ 3G/4G but optimised for IoT/M2M.
- Currently deployed in more than 100 countries worldwide (America, Europe, Africa, Asia).
- Promoted by LoRa Alliance (+500 members).
- Version 1.0: weak against likely practical attacks [AF18].
- Version 1.1: to be deployed.



[AF18] G. Avoine and L. Ferreira. "Rescuing LoRaWAN 1.0". In: *Financial Cryptography and Data Security*. 2018.

Practical application

- SAKE-AM adapted to LoRaWAN 1.1
  - Counters (instead of pseudo-random values).
  - “Confirmation” with the MAC session keys (instead of the updated authentication master key).
- Only change in LoRaWAN 1.1: Join Request message computed with *JSIntKey* (instead of *NwkKey*) master key.
- Additional cost (in most cases)
  - End-device: $2 \times$ H (computation).
  - Server: 1 key (storage) $+ 2 \times$ H (computation).

End-device [A] ($AppKey, NwkKey$)		**Join Server [B]** ($AppKey, NwkKey, JSIntKey_{j-1}$)
	$\xrightarrow{\text{Join Request}}$	
		compute ε
	$\xleftarrow{\text{Join Accept}}$	
	$\xrightarrow{\text{RekeyInd}}$	
	$\xleftarrow{\text{RekeyConf}}$	

Conclusion

- To the best of our knowledge: SAKE is the first protocol
  - in the symmetric-key setting,
  - that provides forward secrecy,
  - with no additional functionality (e.g., synchronised clock, extra procedure),
  - provably secure in a strong security model.
- Limitation: sequential executions. Not an issue depending on the context.
- Advantageous for low-resource devices.
- Suitable for actual use cases (e.g., LoRaWAN).

Symmetric-key Authenticated Key Exchange (SAKE) with Perfect Forward Secrecy

Gildas Avoine[1,2] Sébastien Canard[3] Loïc Ferreira[3,*]

[1] Institut Universitaire de France

[2] Univ Rennes, INSA Rennes, CNRS, IRISA, France

[3] Orange Labs, Applied Cryptography Group, Caen, France

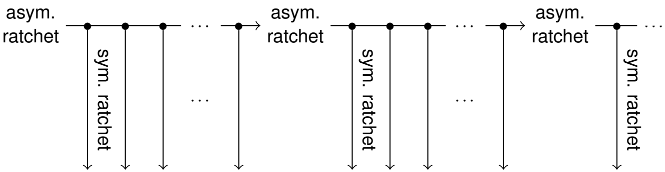
CT-RSA 2020
February 24-28, 2020

[*] Work done while at IRISA

Backup slides

Related work

- Signal's symmetric-key ratchet [PM16], ACD [ACD19], liteARCAD [CDV19], etc.
- Authenticated key exchange protocol vs. asynchronous secure messaging protocols:
  - inspiring yet not strictly comparable,
  - but rather complementary (key exchange phase/application phase).



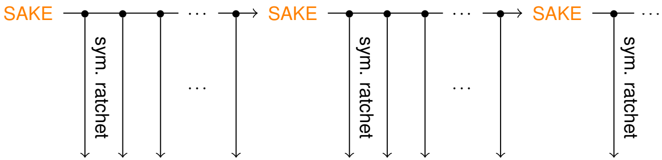
[ACD19] J. Alwen, S. Coretti, and Y. Dodis. "The Double Ratchet: Security Notions, Proofs, and Modularization for the Signal Protocol". In: *EUROCRYPT*. 2019.

[CDV19] A. Caforio, F. B. Durak, and S. Vaudenay. *Beyond Security and Efficiency: On-Demand Ratcheting with Security Awareness*. Cryptology ePrint Archive. 2019.

[PM16] T. Perrin and M. Marlinspike. *The Double Ratchet Algorithm*. 2016.

Related work

- Signal's symmetric-key ratchet [PM16], ACD [ACD19], liteARCAD [CDV19], etc.
- Authenticated key exchange protocol vs. asynchronous secure messaging protocols:
  - inspiring yet not strictly comparable,
  - but rather complementary (key exchange phase/application phase).



[ACD19] J. Alwen, S. Coretti, and Y. Dodis. "The Double Ratchet: Security Notions, Proofs, and Modularization for the Signal Protocol". In: *EUROCRYPT*. 2019.

[CDV19] A. Caforio, F. B. Durak, and S. Vaudenay. *Beyond Security and Efficiency: On-Demand Ratcheting with Security Awareness*. Cryptology ePrint Archive. 2019.

[PM16] T. Perrin and M. Marlinspike. *The Double Ratchet Algorithm*. 2016.