An Implementation of the Hash-Chain Signature Scheme for Wireless Sensor Nodes

Nadia Mourier, Reinhard Stampp, **Falko Strenzke**

FlexSecure GmbH, Germany,
nadia.mourier@gmail.com
reinhard.stampp@cased.de
strenzke@flexsecure.de

May 4, 2013

Introduction

- lightweight public key signature schemes
  - more efficient handling of keys
  - usually not lightweight
- CANS 2009: Dahmen-Krauß Hash-Chain Signature scheme
- Implementation of the scheme on MSP430 CPU
- Correction of an error in the PRNG specification in the original paper
- A new time-memory tradeoff
- Evaluation of two block ciphers

Introduction

- lightweight public key signature schemes
  - more efficient handling of keys
  - usually not lightweight
- CANS 2009: Dahmen-Krauß Hash-Chain Signature scheme
- Implementation of the scheme on MSP430 CPU
- Correction of an error in the PRNG specification in the original paper
- A new time-memory tradeoff
- Evaluation of two block ciphers

- lightweight public key signature schemes
  - more efficient handling of keys
  - usually not lightweight
- CANS 2009: Dahmen-Krauß Hash-Chain Signature scheme
- Implementation of the scheme on MSP430 CPU
- Correction of an error in the PRNG specification in the original paper
- A new time-memory tradeoff
- Evaluation of two block ciphers

Introduction

- lightweight public key signature schemes
  - more efficient handling of keys
  - usually not lightweight
- CANS 2009: Dahmen-Krauß Hash-Chain Signature scheme
- Implementation of the scheme on MSP430 CPU
- Correction of an error in the PRNG specification in the original paper
- A new time-memory tradeoff
- Evaluation of two block ciphers

Introduction

- lightweight public key signature schemes
  - more efficient handling of keys
  - usually not lightweight
- CANS 2009: Dahmen-Krauß Hash-Chain Signature scheme
- Implementation of the scheme on MSP430 CPU
- Correction of an error in the PRNG specification in the original paper
- A new time-memory tradeoff
- Evaluation of two block ciphers

Introduction

- lightweight public key signature schemes
  - more efficient handling of keys
  - usually not lightweight
- CANS 2009: Dahmen-Krauß Hash-Chain Signature scheme
- Implementation of the scheme on MSP430 CPU
- Correction of an error in the PRNG specification in the original paper
- A new time-memory tradeoff
- Evaluation of two block ciphers

Introduction

- lightweight public key signature schemes
  - more efficient handling of keys
  - usually not lightweight
- CANS 2009: Dahmen-Krauß Hash-Chain Signature scheme
- Implementation of the scheme on MSP430 CPU
- Correction of an error in the PRNG specification in the original paper
- A new time-memory tradeoff
- Evaluation of two block ciphers

Introduction

- lightweight public key signature schemes
  - more efficient handling of keys
  - usually not lightweight
- CANS 2009: Dahmen-Krauß Hash-Chain Signature scheme
- Implementation of the scheme on MSP430 CPU
- Correction of an error in the PRNG specification in the original paper
- A new time-memory tradeoff
- Evaluation of two block ciphers

Outline

verification keys v_1 v_2

$H^{256}()$ $\cdots$

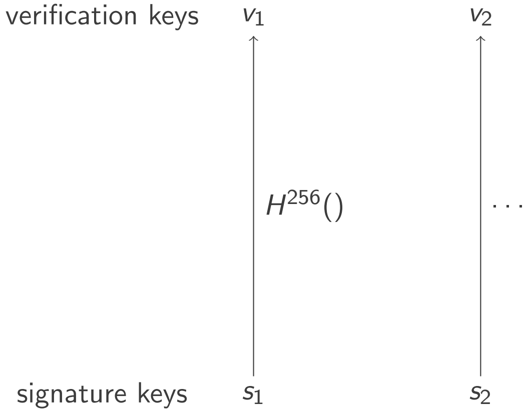
signature keys s_1 s_2

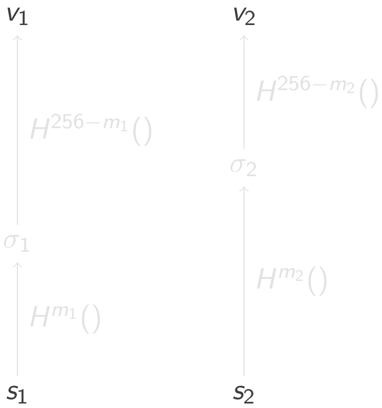
verification keys v_1 v_2

$$v_1 \xleftarrow{H^{256}()} s_1 \qquad \cdots$$

signature keys s_1 s_2

One-Time Signature Schemes – Key Generation





...

attacker can forge signatures
for $m_i' > m_i$
thus also a checksum must be signed:

$$c = \sum(256 - m_i)$$

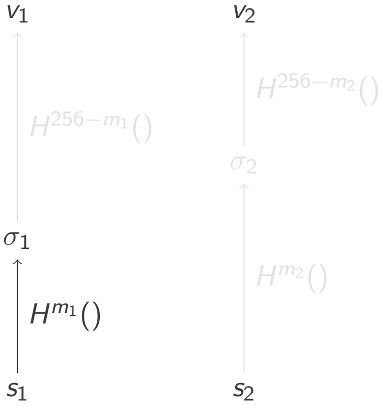
$$s_1 \xrightarrow{H^{m_1}()} \sigma_1 \xrightarrow{H^{256-m_1}()} v_1$$

$$s_2 \xrightarrow{H^{m_2}()} \sigma_2 \xrightarrow{H^{256-m_2}()} v_2$$

$\ldots$

attacker can forge signatures
for $m_i' > m_i$
thus also a checksum must be signed:

$$c = \sum(256 - m_i)$$



. . .

attacker can forge signatures
for $m_i' > m_i$
thus also a checksum must be signed:

$$c = \sum(256 - m_i)$$

One-Time Signature Scheme – Signing and Verification

$$
\begin{tikzcd}
v_1 & v_2 \\
\sigma_1 \arrow[u, "H^{256-m_1}()"'] & \sigma_2 \arrow[u, "H^{256-m_2}()"'] & \cdots \\
s_1 \arrow[u, "H^{m_1}()"'] & s_2 \arrow[u, "H^{m_2}()"']
\end{tikzcd}
$$

attacker can forge signatures
for $m_i' > m_i$
thus also a checksum must be signed:

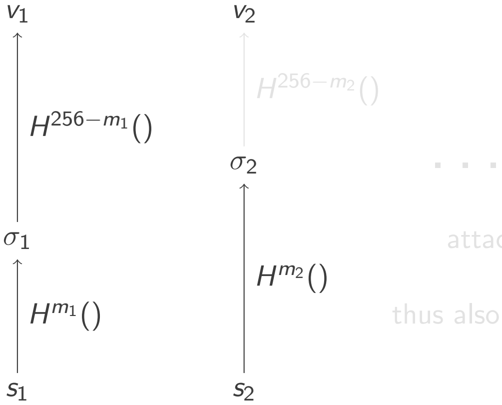
$$c = \sum(256 - m_i)$$

$$
\begin{tikzcd}
v_1 & v_2 & \\
 & \sigma_2 \arrow[u, "H^{256-m_2}()"'] & \cdots \\
\sigma_1 \arrow[uu, "H^{256-m_1}()"'] & & \\
s_1 \arrow[u, "H^{m_1}()"'] & s_2 \arrow[uu, "H^{m_2}()"'] &
\end{tikzcd}
$$

attacker can forge signatures
for $m_i' > m_i$
thus also a checksum must be signed:

$$c = \sum(256 - m_i)$$

One-Time Signature Scheme – Signing and Verification



attacker can forge signatures
for $m_i' > m_i$
thus also a checksum must be signed:

$$c = \sum(256 - m_i)$$

One-Time Signature Scheme – Signing and Verification

$$
\begin{tikzcd}
v_1 & v_2 & \cdots \\
\sigma_1 \arrow[u, "H^{256-m_1}()"'] & \sigma_2 \arrow[u, "H^{256-m_2}()"'] & \\
s_1 \arrow[u, "H^{m_1}()"'] & s_2 \arrow[u, "H^{m_2}()"'] &
\end{tikzcd}
$$

attacker can forge signatures
for $m_i' > m_i$
thus also a checksum must be signed:

$$c = \sum(256 - m_i)$$

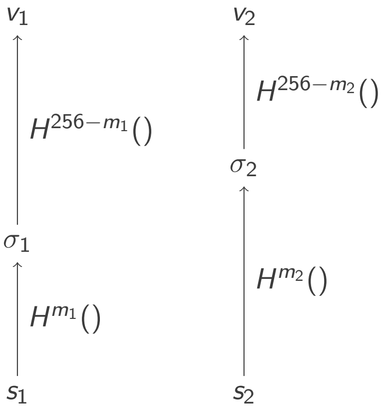
$$s_1 \xrightarrow{H^{m_1}()} \sigma_1 \xrightarrow{H^{256-m_1}()} v_1$$

$$s_2 \xrightarrow{H^{m_2}()} \sigma_2 \xrightarrow{H^{256-m_2}()} v_2$$

...

attacker can forge signatures
for $m_i' > m_i$
thus also a checksum must be signed:

$$c = \sum (256 - m_i)$$



...

attacker can forge signatures

for $m_i' > m_i$

thus also a checksum must be signed:

$$c = \sum(256 - m_i)$$

One-Time Signature Scheme – Signing and Verification

$$\begin{tikzcd} v_1 \\ \sigma_1 \arrow[u, "H^{256-m_1}()"'] \\ s_1 \arrow[u, "H^{m_1}()"'] \end{tikzcd} \qquad \begin{tikzcd} v_2 \\ \sigma_2 \arrow[u, "H^{256-m_2}()"'] \\ s_2 \arrow[u, "H^{m_2}()"'] \end{tikzcd} \qquad \ldots$$

attacker can forge signatures
for $m_i' > m_i$

thus also a checksum must be signed:

$$c = \sum (256 - m_i)$$

One-Time Signature Scheme – Signing and Verification

$$
\begin{tikzcd}
v_1 & v_2 & \\
 & \sigma_2 \arrow[u, "H^{256-m_2}()"'] & \cdots \\
\sigma_1 \arrow[uu, "H^{256-m_1}()"'] & & \\
s_1 \arrow[u, "H^{m_1}()"'] & s_2 \arrow[uu, "H^{m_2}()"'] &
\end{tikzcd}
$$

attacker can forge signatures
for $m_i' > m_i$
thus also a checksum must be signed:

$$c = \sum (256 - m_i)$$

One-Time Signature Scheme – Signing and Verification

$$s_1 \xrightarrow{H^{m_1}()} \sigma_1 \xrightarrow{H^{256-m_1}()} v_1$$

$$s_2 \xrightarrow{H^{m_2}()} \sigma_2 \xrightarrow{H^{256-m_2}()} v_2$$

...

attacker can forge signatures
for $m_i' > m_i$
thus also a checksum must be signed:

$$c = \sum(256 - m_i)$$

Multiple Signature Schemes from One-Time Signature Schemes

- efficient handling of signature keys:
- $\rightarrow$ on demand creation through PRNG
- small public key
- $\rightarrow$ Merkle Tree Scheme
- $\rightarrow$ Hash Chain Signature Scheme by Dahmen and Krauß

Multiple Signature Schemes from One-Time Signature Schemes

- efficient handling of signature keys:
- $\rightarrow$ on demand creation through PRNG
- small public key
- $\rightarrow$ Merkle Tree Scheme
- $\rightarrow$ Hash Chain Signature Scheme by Dahmen and Krauß

Multiple Signature Schemes from One-Time Signature Schemes

- efficient handling of signature keys:
- $\rightarrow$ on demand creation through PRNG
- small public key
- $\rightarrow$ Merkle Tree Scheme
- $\rightarrow$ Hash Chain Signature Scheme by Dahmen and Krauß

Multiple Signature Schemes from One-Time Signature Schemes

- efficient handling of signature keys:
- $\rightarrow$ on demand creation through PRNG
- small public key
- $\rightarrow$ Merkle Tree Scheme
- $\rightarrow$ Hash Chain Signature Scheme by Dahmen and Krauß

Multiple Signature Schemes from One-Time Signature Schemes

- efficient handling of signature keys:
- $\rightarrow$ on demand creation through PRNG
- small public key
- $\rightarrow$ Merkle Tree Scheme
- $\rightarrow$ Hash Chain Signature Scheme by Dahmen and Krauß

DKSS Key Generation

signature generation $\longrightarrow$

$$
\begin{tikzcd}
z_0 & z_1 \arrow[l] & z_2 \arrow[l] & \cdots \arrow[l] & z_{l-1} \arrow[l] & z_l \arrow[l] \\
 & Y_1 \arrow[ul] & Y_2 \arrow[ul] & & & Y_l \arrow[ul] \\
\mathrm{PRNG} \arrow[r] & X_1 \arrow[u] & X_2 \arrow[u] & & & X_l \arrow[u]
\end{tikzcd}
$$

Hash chain traversal algorithm: Yum et al., CT-RSA 2009:
Storage: $\lceil 1/2 \log l \rceil$ links z_i
Computational cost: $\lceil \log l \rceil$

DKSS Key Generation

$$\xrightarrow{\qquad\qquad\qquad\text{signature generation}\qquad\qquad\qquad}$$

$$
\begin{tikzcd}
z_0 & z_1 \arrow[l] & z_2 \arrow[l] & \cdots \arrow[l] & z_{l-1} \arrow[l] & z_l \arrow[l] \\
 & Y_1 \arrow[ul] & Y_2 \arrow[ul] & & & Y_l \arrow[ul] \\
\text{PRNG} \arrow[r] & X_1 \arrow[u] & X_2 \arrow[u] & & & X_l \arrow[u]
\end{tikzcd}
$$

Hash chain traversal algorithm: Yum et al., CT-RSA 2009:
Storage: $\lceil 1/2 \log l \rceil$ links z_i
Computational cost: $\lceil \log l \rceil$

DKSS Key Generation

$$
\begin{tikzcd}
z_0 & z_1 \arrow[l] & z_2 \arrow[l] & \cdots \arrow[l] & z_{l-1} \arrow[l] & z_l \arrow[l] \\
 & Y_1 \arrow[ul] & Y_2 \arrow[ul] & & & Y_l \arrow[ul] \\
\mathrm{PRNG} \arrow[r] & X_1 \arrow[u] & X_2 \arrow[u] & & & X_l \arrow[u]
\end{tikzcd}
$$

signature generation $\longrightarrow$

Hash chain traversal algorithm: Yum et al., CT-RSA 2009:

Storage: $\lceil 1/2 \log l \rceil$ links z_i

Computational cost: $\lceil \log l \rceil$

$$\xrightarrow{\hspace{12em}\text{signature generation}\hspace{12em}}$$

$$\begin{tikzcd}
z_0 & z_1 \arrow[l] & z_2 \arrow[l] & \cdots \arrow[l] & z_{l-1} \arrow[l] & z_l \arrow[l] \\
 & Y_1 \arrow[ul] & Y_2 \arrow[ul] & & & Y_l \arrow[ul] \\
\text{PRNG} \arrow[r] & X_1 \arrow[u] & X_2 \arrow[u] & & & X_l \arrow[u]
\end{tikzcd}$$

Hash chain traversal algorithm: Yum et al., CT-RSA 2009:
Storage: $\lceil 1/2 \log l \rceil$ links z_i
Computational cost: $\lceil \log l \rceil$

$$\xrightarrow{\text{signature generation}}$$

$$\begin{tikzcd}
& & \bullet & & & \bullet \\
z_0 & z_1 \arrow[l] & z_2 \arrow[l] & \cdots \arrow[l] & z_{l-1} \arrow[l] & z_l \arrow[l] \\
& Y_1 \arrow[ul] & Y_2 \arrow[ul] & & & Y_l \arrow[ul] \\
\text{PRNG} \arrow[r] & X_1 \arrow[u] & X_2 \arrow[u] & & & X_l \arrow[u]
\end{tikzcd}$$

Hash chain traversal algorithm: Yum et al., CT-RSA 2009:

Storage: $\lceil 1/2 \log l \rceil$ links z_i

Computational cost: $\lceil \log l \rceil$

DKSS Key Generation

signature generation $\longrightarrow$

$$
\begin{tikzcd}
z_0 & z_1 \arrow[l] & z_2 \arrow[l] & \cdots \arrow[l] & z_{l-1} \arrow[l] & z_l \arrow[l] \\
 & Y_1 \arrow[ul] & Y_2 \arrow[ul] & & & Y_l \arrow[ul] \\
\text{PRNG} \arrow[r] & X_1 \arrow[u] & X_2 \arrow[u] & & & X_l \arrow[u]
\end{tikzcd}
$$

Hash chain traversal algorithm: Yum et al., CT-RSA 2009:

Storage: $\lceil 1/2 \log l \rceil$ links z_i

Computational cost: $\lceil \log l \rceil$

DKSS Key Generation

$$
\begin{tikzcd}
{} \arrow[rrrrr, "\text{signature generation}"] & & & & & {} \\
 & & \bullet & & & \bullet \\
z_0 & z_1 \arrow[l] & z_2 \arrow[l] & \cdots \arrow[l] & z_{l-1} \arrow[l] & z_l \arrow[l] \\
 & Y_1 \arrow[ul] & Y_2 \arrow[ul] & & & Y_l \arrow[ul] \\
\text{PRNG} \arrow[r] & X_1 \arrow[u] & X_2 \arrow[u] & & & X_l \arrow[u]
\end{tikzcd}
$$

Hash chain traversal algorithm: Yum et al., CT-RSA 2009:
Storage: $\lceil 1/2 \log l \rceil$ links z_i
Computational cost: $\lceil \log l \rceil$

Parameters and Hash Functions of the Scheme

- w – bit size of message
- l – number of signatures
- n – security level (bit size of hash output)

- hash functions
  - $f : \{0,1\}^n \longrightarrow \{0,1\}^n$
  - $g : \{0,1\}^{4n} \longrightarrow \{0,1\}^n$

$$f(x) = \lfloor E_{IV}(x) \oplus x \rfloor_n$$

$$\begin{aligned} g(x_1, x_2, x_3, x_4) &= \lfloor E_{k_3}(x_4) \oplus x_4 \rfloor_n \\ \text{with } k_3 &= E_{k_2}(x_3) \oplus x_3 \\ k_2 &= E_{k_1}(x_2) \oplus x_2 \\ k_1 &= E_{IV}(x_1) \oplus x_1 \end{aligned}$$

Parameters and Hash Functions of the Scheme

- w – bit size of message
- l – number of signatures
- n – security level (bit size of hash output)

- hash functions
  - $f : \{0,1\}^n \longrightarrow \{0,1\}^n$
  - $g : \{0,1\}^{4n} \longrightarrow \{0,1\}^n$

-

$$f(x) = \lfloor E_{IV}(x) \oplus x \rfloor_n$$

$$\begin{aligned} g(x_1, x_2, x_3, x_4) &= \lfloor E_{k_3}(x_4) \oplus x_4 \rfloor_n \\ \text{with } k_3 &= E_{k_2}(x_3) \oplus x_3 \\ k_2 &= E_{k_1}(x_2) \oplus x_2 \\ k_1 &= E_{IV}(x_1) \oplus x_1 \end{aligned}$$

Parameters and Hash Functions of the Scheme

- w – bit size of message
- l – number of signatures
- n – security level (bit size of hash output)

- hash functions
  - $f : \{0,1\}^n \longrightarrow \{0,1\}^n$
  - $g : \{0,1\}^{4n} \longrightarrow \{0,1\}^n$
-

$$f(x) = \lfloor E_{IV}(x) \oplus x \rfloor_n$$

$$\begin{aligned} g(x_1, x_2, x_3, x_4) &= \lfloor E_{k_3}(x_4) \oplus x_4 \rfloor_n \\ \text{with } k_3 &= E_{k_2}(x_3) \oplus x_3 \\ k_2 &= E_{k_1}(x_2) \oplus x_2 \\ k_1 &= E_{IV}(x_1) \oplus x_1 \end{aligned}$$

Parameters and Hash Functions of the Scheme

- w – bit size of message
- l – number of signatures
- n – security level (bit size of hash output)

- hash functions
  - $f \;:\; \{0,1\}^n \longrightarrow \{0,1\}^n$
  - $g \;:\; \{0,1\}^{4n} \longrightarrow \{0,1\}^n$

$$f(x) = \lfloor E_{IV}(x) \oplus x \rfloor_n$$

$$\begin{aligned} g(x_1, x_2, x_3, x_4) &= \lfloor E_{k_3}(x_4) \oplus x_4 \rfloor_n \\ \text{with } k_3 &= E_{k_2}(x_3) \oplus x_3 \\ k_2 &= E_{k_1}(x_2) \oplus x_2 \\ k_1 &= E_{IV}(x_1) \oplus x_1 \end{aligned}$$

Parameters and Hash Functions of the Scheme

- w – bit size of message
- l – number of signatures
- n – security level (bit size of hash output)

- hash functions
  - $f \; : \; \{0,1\}^n \longrightarrow \{0,1\}^n$
  - $g \; : \; \{0,1\}^{4n} \longrightarrow \{0,1\}^n$

$$f(x) = \lfloor E_{IV}(x) \oplus x \rfloor_n$$

$$\begin{aligned} g(x_1, x_2, x_3, x_4) &= \lfloor E_{k_3}(x_4) \oplus x_4 \rfloor_n \\ \text{with } k_3 &= E_{k_2}(x_3) \oplus x_3 \\ k_2 &= E_{k_1}(x_2) \oplus x_2 \\ k_1 &= E_{IV}(x_1) \oplus x_1 \end{aligned}$$

Parameters and Hash Functions of the Scheme

- w – bit size of message
- l – number of signatures
- n – security level (bit size of hash output)

- hash functions
  - $f \ : \ \{0,1\}^n \longrightarrow \{0,1\}^n$
  - $g \ : \ \{0,1\}^{4n} \longrightarrow \{0,1\}^n$

$$f(x) = \lfloor E_{IV}(x) \oplus x \rfloor_n$$

$$g(x_1, x_2, x_3, x_4) = \lfloor E_{k_3}(x_4) \oplus x_4 \rfloor_n$$

$$\text{with } k_3 = E_{k_2}(x_3) \oplus x_3$$

$$k_2 = E_{k_1}(x_2) \oplus x_2$$

$$k_1 = E_{IV}(x_1) \oplus x_1$$

Parameters and Hash Functions of the Scheme

- w – bit size of message
- l – number of signatures
- n – security level (bit size of hash output)

- hash functions
  - $f \;:\; \{0,1\}^n \longrightarrow \{0,1\}^n$
  - $g \;:\; \{0,1\}^{4n} \longrightarrow \{0,1\}^n$

-

$$f(x) = \lfloor E_{IV}(x) \oplus x \rfloor_n$$

$$\begin{aligned} g(x_1, x_2, x_3, x_4) &= \lfloor E_{k_3}(x_4) \oplus x_4 \rfloor_n \\ \text{with } k_3 &= E_{k_2}(x_3) \oplus x_3 \\ k_2 &= E_{k_1}(x_2) \oplus x_2 \\ k_1 &= E_{IV}(x_1) \oplus x_1 \end{aligned}$$

- Signature Size
  - security level $n = 80$
  - signature size for $w = 16$: 336 bits
  - ECDSA 160: 320 bits
- signing of short fixed size messages ($\neq$ Merkle)
- Verifier need all previous signatures ($\neq$ Merkle)
- fixed number of signatures per public key ($=$ Merkle)
- small public key (80 bit) ($=$ Merkle)
- verification much faster than signature generation
- intended application: broadcast messages in WSNs

Features

- Signature Size
  - security level $n = 80$
  - signature size for $w = 16$: 336 bits
  - ECDSA 160: 320 bits
- signing of short fixed size messages ($\neq$ Merkle)
- Verifier need all previous signatures ($\neq$ Merkle)
- fixed number of signatures per public key ($=$ Merkle)
- small public key (80 bit) ($=$ Merkle)
- verification much faster than signature generation
- intended application: broadcast messages in WSNs

Features

- Signature Size
  - security level $n = 80$
  - signature size for $w = 16$: 336 bits
  - ECDSA 160: 320 bits
- signing of short fixed size messages ($\neq$ Merkle)
- Verifier need all previous signatures ($\neq$ Merkle)
- fixed number of signatures per public key ($=$ Merkle)
- small public key (80 bit) ($=$ Merkle)
- verification much faster than signature generation
- intended application: broadcast messages in WSNs

Features

- Signature Size
  - security level $n = 80$
  - signature size for $w = 16$: 336 bits
  - ECDSA 160: 320 bits
- signing of short fixed size messages ($\neq$ Merkle)
- Verifier need all previous signatures ($\neq$ Merkle)
- fixed number of signatures per public key ($=$ Merkle)
- small public key (80 bit) ($=$ Merkle)
- verification much faster than signature generation
- intended application: broadcast messages in WSNs

Features

- Signature Size
  - security level $n = 80$
  - signature size for $w = 16$: 336 bits
  - ECDSA 160: 320 bits
- signing of short fixed size messages ($\neq$ Merkle)
- Verifier need all previous signatures ($\neq$ Merkle)
- fixed number of signatures per public key ($=$ Merkle)
- small public key (80 bit) ($=$ Merkle)
- verification much faster than signature generation
- intended application: broadcast messages in WSNs

Features

- Signature Size
  - security level $n = 80$
  - signature size for $w = 16$: 336 bits
  - ECDSA 160: 320 bits
- signing of short fixed size messages ($\neq$ Merkle)
- Verifier need all previous signatures ($\neq$ Merkle)
- fixed number of signatures per public key ($=$ Merkle)
- small public key (80 bit) ($=$ Merkle)
- verification much faster than signature generation
- intended application: broadcast messages in WSNs

Features

- Signature Size
  - security level $n = 80$
  - signature size for $w = 16$: 336 bits
  - ECDSA 160: 320 bits
- signing of short fixed size messages ($\neq$ Merkle)
- Verifier need all previous signatures ($\neq$ Merkle)
- fixed number of signatures per public key ($=$ Merkle)
- small public key (80 bit) ($=$ Merkle)
- verification much faster than signature generation
- intended application: broadcast messages in WSNs

Features

- Signature Size
  - security level $n = 80$
  - signature size for $w = 16$: 336 bits
  - ECDSA 160: 320 bits
- signing of short fixed size messages ($\neq$ Merkle)
- Verifier need all previous signatures ($\neq$ Merkle)
- fixed number of signatures per public key ($=$ Merkle)
- small public key (80 bit) ($=$ Merkle)
- verification much faster than signature generation
- intended application: broadcast messages in WSNs

Features

- Signature Size
  - security level $n = 80$
  - signature size for $w = 16$: 336 bits
  - ECDSA 160: 320 bits
- signing of short fixed size messages ($\neq$ Merkle)
- Verifier need all previous signatures ($\neq$ Merkle)
- fixed number of signatures per public key ($=$ Merkle)
- small public key (80 bit) ($=$ Merkle)
- verification much faster than signature generation
- intended application: broadcast messages in WSNs

Features

- Signature Size
  - security level $n = 80$
  - signature size for $w = 16$: 336 bits
  - ECDSA 160: 320 bits
- signing of short fixed size messages ($\neq$ Merkle)
- Verifier need all previous signatures ($\neq$ Merkle)
- fixed number of signatures per public key ($=$ Merkle)
- small public key (80 bit) ($=$ Merkle)
- verification much faster than signature generation
- intended application: broadcast messages in WSNs

- Compute the one-time signature keys $X_i = (x_i[0], x_i[1], x_i[2]) \in \{0,1\}^{(n,3)}$:
- for $i = 1$ to l
  - $x_i[0] \leftarrow \mathrm{PRNG}(i, 0)$
  - $x_i[1] \leftarrow \mathrm{PRNG}(i, 1)$
  - $x_i[2] \leftarrow \mathrm{PRNG}(i, 2)$
- Calculate the one-time verification key $Y_i = (y_i[0], y_i[1], y_i[2]) \in \{0,1\}^{(n,3)}$:
- for $i = l$ to 1
  - $y_i[0] \longleftarrow f^{2^{\frac{w}{2}}-1}(x_i[0])$
  - $y_i[1] \longleftarrow f^{2^{\frac{w}{2}}-1}(x_i[1])$
  - $y_i[2] \longleftarrow f^{2^{\frac{w}{2}+1}-2}(x_i[2])$
  - $z_{i-1} \longleftarrow g(y_i[0] \parallel y_i[1] \parallel y_i[2] \parallel z_i)$

Details of DKSS Key Generation

- Compute the one-time signature keys $X_i = (x_i[0], x_i[1], x_i[2]) \in \{0,1\}^{(n,3)}$:
- for $i = 1$ to l
  - $x_i[0] \leftarrow \mathrm{PRNG}(i, 0)$
  - $x_i[1] \leftarrow \mathrm{PRNG}(i, 1)$
  - $x_i[2] \leftarrow \mathrm{PRNG}(i, 2)$
- Calculate the one-time verification key $Y_i = (y_i[0], y_i[1], y_i[2]) \in \{0,1\}^{(n,3)}$:
- for $i = l$ to 1
  - $y_i[0] \longleftarrow f^{2^{\frac{w}{2}}-1}(x_i[0])$
  - $y_i[1] \longleftarrow f^{2^{\frac{w}{2}}-1}(x_i[1])$
  - $y_i[2] \longleftarrow f^{2^{\frac{w}{2}+1}-2}(x_i[2])$
  - $z_{i-1} \longleftarrow g(y_i[0] \,\|\, y_i[1] \,\|\, y_i[2] \,\|\, z_i)$

- Compute the one-time signature keys $X_i = (x_i[0], x_i[1], x_i[2]) \in \{0,1\}^{(n,3)}$:
- for $i = 1$ to l
  - $x_i[0] \leftarrow \mathrm{PRNG}(i, 0)$
  - $x_i[1] \leftarrow \mathrm{PRNG}(i, 1)$
  - $x_i[2] \leftarrow \mathrm{PRNG}(i, 2)$
- Calculate the one-time verification key $Y_i = (y_i[0], y_i[1], y_i[2]) \in \{0,1\}^{(n,3)}$:
- for $i = l$ to 1
  - $y_i[0] \longleftarrow f^{2^{\frac{w}{2}}-1}(x_i[0])$
  - $y_i[1] \longleftarrow f^{2^{\frac{w}{2}}-1}(x_i[1])$
  - $y_i[2] \longleftarrow f^{2^{\frac{w}{2}+1}-2}(x_i[2])$
  - $z_{i-1} \longleftarrow g(y_i[0] \parallel y_i[1] \parallel y_i[2] \parallel z_i)$

Details of DKSS Key Generation

- Compute the one-time signature keys $X_i = (x_i[0], x_i[1], x_i[2]) \in \{0,1\}^{(n,3)}$:
- for $i = 1$ to l
  - $x_i[0] \leftarrow \mathrm{PRNG}(i, 0)$
  - $x_i[1] \leftarrow \mathrm{PRNG}(i, 1)$
  - $x_i[2] \leftarrow \mathrm{PRNG}(i, 2)$
- Calculate the one-time verification key $Y_i = (y_i[0], y_i[1], y_i[2]) \in \{0,1\}^{(n,3)}$:
- for $i = l$ to 1
  - $y_i[0] \longleftarrow f^{2^{\frac{w}{2}}-1}(x_i[0])$
  - $y_i[1] \longleftarrow f^{2^{\frac{w}{2}}-1}(x_i[1])$
  - $y_i[2] \longleftarrow f^{2^{\frac{w}{2}+1}-2}(x_i[2])$
  - $z_{i-1} \longleftarrow g(y_i[0] \parallel y_i[1] \parallel y_i[2] \parallel z_i)$

- Compute the one-time signature keys $X_i = (x_i[0], x_i[1], x_i[2]) \in \{0,1\}^{(n,3)}$:
- for $i = 1$ to l
  - $x_i[0] \leftarrow \mathrm{PRNG}(i, 0)$
  - $x_i[1] \leftarrow \mathrm{PRNG}(i, 1)$
  - $x_i[2] \leftarrow \mathrm{PRNG}(i, 2)$
- Calculate the one-time verification key $Y_i = (y_i[0], y_i[1], y_i[2]) \in \{0,1\}^{(n,3)}$:
- for $i = l$ to 1
  - $y_i[0] \longleftarrow f^{2^{\frac{w}{2}}-1}(x_i[0])$
  - $y_i[1] \longleftarrow f^{2^{\frac{w}{2}}-1}(x_i[1])$
  - $y_i[2] \longleftarrow f^{2^{\frac{w}{2}+1}-2}(x_i[2])$
  - $z_{i-1} \longleftarrow g(y_i[0] \parallel y_i[1] \parallel y_i[2] \parallel z_i)$

Details of DKSS Key Generation

- Compute the one-time signature keys $X_i = (x_i[0], x_i[1], x_i[2]) \in \{0,1\}^{(n,3)}$:
- for $i = 1$ to l
  - $x_i[0] \leftarrow \mathrm{PRNG}(i, 0)$
  - $x_i[1] \leftarrow \mathrm{PRNG}(i, 1)$
  - $x_i[2] \leftarrow \mathrm{PRNG}(i, 2)$
- Calculate the one-time verification key $Y_i = (y_i[0], y_i[1], y_i[2]) \in \{0,1\}^{(n,3)}$:
- for $i = l$ to 1
  - $y_i[0] \longleftarrow f^{2^{\frac{w}{2}}-1}(x_i[0])$
  - $y_i[1] \longleftarrow f^{2^{\frac{w}{2}}-1}(x_i[1])$
  - $y_i[2] \longleftarrow f^{2^{\frac{w}{2}+1}-2}(x_i[2])$
  - $z_{i-1} \longleftarrow g(y_i[0] \parallel y_i[1] \parallel y_i[2] \parallel z_i)$

Details of DKSS Key Generation

- Compute the one-time signature keys $X_i = (x_i[0], x_i[1], x_i[2]) \in \{0,1\}^{(n,3)}$:
- for $i = 1$ to l
  - $x_i[0] \leftarrow \mathrm{PRNG}(i, 0)$
  - $x_i[1] \leftarrow \mathrm{PRNG}(i, 1)$
  - $x_i[2] \leftarrow \mathrm{PRNG}(i, 2)$
- Calculate the one-time verification key $Y_i = (y_i[0], y_i[1], y_i[2]) \in \{0,1\}^{(n,3)}$:
- for $i = l$ to 1
  - $y_i[0] \longleftarrow f^{2^{\frac{w}{2}}-1}(x_i[0])$
  - $y_i[1] \longleftarrow f^{2^{\frac{w}{2}}-1}(x_i[1])$
  - $y_i[2] \longleftarrow f^{2^{\frac{w}{2}+1}-2}(x_i[2])$
  - $z_{i-1} \longleftarrow g(y_i[0] \parallel y_i[1] \parallel y_i[2] \parallel z_i)$

Details of DKSS Key Generation

- Compute the one-time signature keys $X_i = (x_i[0], x_i[1], x_i[2]) \in \{0,1\}^{(n,3)}$:
- for $i = 1$ to l
  - $x_i[0] \leftarrow \mathrm{PRNG}(i, 0)$
  - $x_i[1] \leftarrow \mathrm{PRNG}(i, 1)$
  - $x_i[2] \leftarrow \mathrm{PRNG}(i, 2)$
- Calculate the one-time verification key $Y_i = (y_i[0], y_i[1], y_i[2]) \in \{0,1\}^{(n,3)}$:
- for $i = l$ to 1
  - $y_i[0] \longleftarrow f^{2^{\frac{w}{2}}-1}(x_i[0])$
  - $y_i[1] \longleftarrow f^{2^{\frac{w}{2}}-1}(x_i[1])$
  - $y_i[2] \longleftarrow f^{2^{\frac{w}{2}+1}-2}(x_i[2])$
  - $z_{i-1} \longleftarrow g(y_i[0] \parallel y_i[1] \parallel y_i[2] \parallel z_i)$

Details of DKSS Key Generation

- Compute the one-time signature keys $X_i = (x_i[0], x_i[1], x_i[2]) \in \{0,1\}^{(n,3)}$:
- for $i = 1$ to l
  - $x_i[0] \leftarrow \mathrm{PRNG}(i, 0)$
  - $x_i[1] \leftarrow \mathrm{PRNG}(i, 1)$
  - $x_i[2] \leftarrow \mathrm{PRNG}(i, 2)$
- Calculate the one-time verification key $Y_i = (y_i[0], y_i[1], y_i[2]) \in \{0,1\}^{(n,3)}$:
- for $i = l$ to 1
  - $y_i[0] \longleftarrow f^{2^{\frac{w}{2}}-1}(x_i[0])$
  - $y_i[1] \longleftarrow f^{2^{\frac{w}{2}}-1}(x_i[1])$
  - $y_i[2] \longleftarrow f^{2^{\frac{w}{2}+1}-2}(x_i[2])$
  - $z_{i-1} \longleftarrow g(y_i[0] \parallel y_i[1] \parallel y_i[2] \parallel z_i)$

Details of DKSS Key Generation

- Compute the one-time signature keys $X_i = (x_i[0], x_i[1], x_i[2]) \in \{0,1\}^{(n,3)}$:
- for $i = 1$ to l
  - $x_i[0] \leftarrow \mathrm{PRNG}(i, 0)$
  - $x_i[1] \leftarrow \mathrm{PRNG}(i, 1)$
  - $x_i[2] \leftarrow \mathrm{PRNG}(i, 2)$
- Calculate the one-time verification key $Y_i = (y_i[0], y_i[1], y_i[2]) \in \{0,1\}^{(n,3)}$:
- for $i = l$ to 1
  - $y_i[0] \longleftarrow f^{2^{\frac{w}{2}}-1}(x_i[0])$
  - $y_i[1] \longleftarrow f^{2^{\frac{w}{2}}-1}(x_i[1])$
  - $y_i[2] \longleftarrow f^{2^{\frac{w}{2}+1}-2}(x_i[2])$
  - $z_{i-1} \longleftarrow g(y_i[0] \parallel y_i[1] \parallel y_i[2] \parallel z_i)$

Details of DKSS Key Generation

- Compute the one-time signature keys $X_i = (x_i[0], x_i[1], x_i[2]) \in \{0,1\}^{(n,3)}$:
- for $i = 1$ to l
  - $x_i[0] \leftarrow \mathrm{PRNG}(i, 0)$
  - $x_i[1] \leftarrow \mathrm{PRNG}(i, 1)$
  - $x_i[2] \leftarrow \mathrm{PRNG}(i, 2)$
- Calculate the one-time verification key $Y_i = (y_i[0], y_i[1], y_i[2]) \in \{0,1\}^{(n,3)}$:
- for $i = l$ to 1
  - $y_i[0] \longleftarrow f^{2^{\frac{w}{2}}-1}(x_i[0])$
  - $y_i[1] \longleftarrow f^{2^{\frac{w}{2}}-1}(x_i[1])$
  - $y_i[2] \longleftarrow f^{2^{\frac{w}{2}+1}-2}(x_i[2])$
  - $z_{i-1} \longleftarrow g(y_i[0] \parallel y_i[1] \parallel y_i[2] \parallel z_i)$

DKSS Signature Generation and Verification

$m = m_1 \| m_2$

$y_i[0]$ $\quad$ $y_i[1]$ $\quad$ $y_i[2]$

$f^{2^{\frac{w}{2}}-1-m_1}(\alpha_1)$ $\quad$ $f^{2^{\frac{w}{2}}-1-m_2}(\alpha_2)$ $\quad$ $f^{2^{\frac{w}{2}+1}-2-c}(\alpha_3)$

α_1 $\quad$ α_2 $\quad$ α_3 $\quad$ z_i

$f^{m_1}(x_i[0])$ $\quad$ $f^{m_2}(x_i[1])$ $\quad$ $f^{c}(x_i[2])$

$x_i[0]$ $\quad$ $x_i[1]$ $\quad$ $x_i[2]$

$c \longleftarrow 2^{\frac{w}{2}+1} - 2 - m_1 - m_2$

verification: $g(y_i[0] \| y_i[1] \| y_i[2] \| z_i) = z_{i-1}$?

$m = m_1 \| m_2$

$y_i[0]$ $\quad$ $y_i[1]$ $\quad$ $y_i[2]$

$f^{2^{\frac{w}{2}}-1-m_2}(\alpha_2)$

$f^{2^{\frac{w}{2}}-1-m_1}(\alpha_1)$

α_2

$f^{2^{\frac{w}{2}+1}-2-c}(\alpha_3)$

α_1

$f^{m_2}(x_i[1])$

$f^{m_1}(x_i[0])$

α_3 $\quad$ z_i

$f^{c}(x_i[2])$

$x_i[0]$ $\quad$ $x_i[1]$ $\quad$ $x_i[2]$

$c \longleftarrow 2^{\frac{w}{2}+1} - 2 - m_1 - m_2$

verification: $g(y_i[0] \| y_i[1] \| y_i[2] \| z_i) = z_{i-1}$?

DKSS Signature Generation and Verification

$m = m_1 \| m_2$

$y_i[0]$ $\quad$ $y_i[1]$ $\quad$ $y_i[2]$

$f^{2^{\frac{w}{2}}-1-m_2}(\alpha_2)$

$f^{2^{\frac{w}{2}}-1-m_1}(\alpha_1)$

α_2

$f^{2^{\frac{w}{2}+1}-2-c}(\alpha_3)$

α_1

$f^{m_2}(x_i[1])$

$f^{m_1}(x_i[0])$

α_3 $\quad$ z_i

$f^{c}(x_i[2])$

$x_i[0]$ $\quad$ $x_i[1]$ $\quad$ $x_i[2]$

$c \longleftarrow 2^{\frac{w}{2}+1} - 2 - m_1 - m_2$

verification: $g(y_i[0] \| y_i[1] \| y_i[2] \| z_i) = z_{i-1}$?

DKSS Signature Generation and Verification

$m = m_1 \| m_2$

$$
\begin{tikzcd}
y_i[0] & y_i[1] & y_i[2] & \\
\alpha_1 \arrow[u, "f^{2^{\frac{w}{2}}-1-m_1}(\alpha_1)"'] & \alpha_2 \arrow[u, "f^{2^{\frac{w}{2}}-1-m_2}(\alpha_2)"'] & \alpha_3 \arrow[u, "f^{2^{\frac{w}{2}+1}-2-c}(\alpha_3)"'] & z_i \\
x_i[0] \arrow[u, "f^{m_1}(x_i[0])"'] & x_i[1] \arrow[u, "f^{m_2}(x_i[1])"'] & x_i[2] \arrow[u, "f^{c}(x_i[2])"'] &
\end{tikzcd}
$$

$c \longleftarrow 2^{\frac{w}{2}+1} - 2 - m_1 - m_2$

verification: $g(y_i[0] \| y_i[1] \| y_i[2] \| z_i) = z_{i-1}$?

DKSS Signature Generation and Verification

$m = m_1 \| m_2$

$$
\begin{array}{ccccccc}
y_i[0] & & y_i[1] & & y_i[2] & & \\
\uparrow f^{2^{\frac{w}{2}}-1-m_1}(\alpha_1) & & \uparrow f^{2^{\frac{w}{2}}-1-m_2}(\alpha_2) & & \uparrow f^{2^{\frac{w}{2}+1}-2-c}(\alpha_3) & & \\
 & & \alpha_2 & & & & \\
\alpha_1 & & \uparrow f^{m_2}(x_i[1]) & & & & \\
\uparrow f^{m_1}(x_i[0]) & & & & \alpha_3 & & z_i \\
 & & & & \uparrow f^{c}(x_i[2]) & & \\
x_i[0] & & x_i[1] & & x_i[2] & &
\end{array}
$$

$c \longleftarrow 2^{\frac{w}{2}+1} - 2 - m_1 - m_2$

verification: $g(y_i[0] \| y_i[1] \| y_i[2] \| z_i) = z_{i-1}$?

$m = m_1 \| m_2$

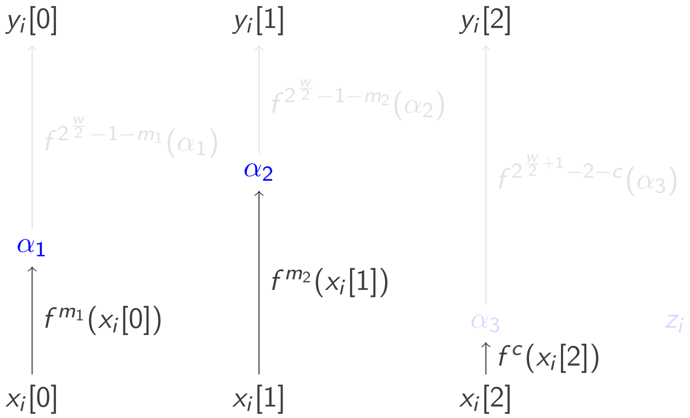
$$\begin{tikzcd}
y_i[0] & y_i[1] & y_i[2] & \\
& \alpha_2 \arrow[u, "f^{2^{\frac{w}{2}}-1-m_2}(\alpha_2)"'] & & \\
\alpha_1 \arrow[uu, "f^{2^{\frac{w}{2}}-1-m_1}(\alpha_1)"'] & & & \\
& & \alpha_3 \arrow[uuu, "f^{2^{\frac{w}{2}+1}-2-c}(\alpha_3)"'] & z_i \\
x_i[0] \arrow[uu, "f^{m_1}(x_i[0])"'] & x_i[1] \arrow[uuu, "f^{m_2}(x_i[1])"'] & x_i[2] \arrow[u, "f^{c}(x_i[2])"'] &
\end{tikzcd}$$

$c \longleftarrow 2^{\frac{w}{2}+1} - 2 - m_1 - m_2$

verification: $g(y_i[0] \| y_i[1] \| y_i[2] \| z_i) = z_{i-1}$?

DKSS Signature Generation and Verification

$m = m_1 \| m_2$



$c \longleftarrow 2^{\frac{w}{2}+1} - 2 - m_1 - m_2$

verification: $g(y_i[0] \| y_i[1] \| y_i[2] \| z_i) = z_{i-1}$?

DKSS Signature Generation and Verification

$m = m_1 \| m_2$

$$
\begin{tikzcd}
y_i[0] & y_i[1] & y_i[2] & \\
 & \alpha_2 \arrow[u, "f^{2^{\frac{w}{2}}-1-m_2}(\alpha_2)"'] & & \\
\alpha_1 \arrow[uu, "f^{2^{\frac{w}{2}}-1-m_1}(\alpha_1)"'] & & \alpha_3 \arrow[uu, "f^{2^{\frac{w}{2}+1}-2-c}(\alpha_3)"'] & z_i \\
x_i[0] \arrow[u, "f^{m_1}(x_i[0])"'] & x_i[1] \arrow[uu, "f^{m_2}(x_i[1])"'] & x_i[2] \arrow[u, "f^{c}(x_i[2])"'] &
\end{tikzcd}
$$

$c \longleftarrow 2^{\frac{w}{2}+1} - 2 - m_1 - m_2$

verification: $g(y_i[0] \| y_i[1] \| y_i[2] \| z_i) = z_{i-1}$?

$m = m_1 \| m_2$

$$y_i[0] \xleftarrow{f^{2^{\frac{w}{2}}-1-m_1}(\alpha_1)} \alpha_1 \xleftarrow{f^{m_1}(x_i[0])} x_i[0]$$

$$y_i[1] \xleftarrow{f^{2^{\frac{w}{2}}-1-m_2}(\alpha_2)} \alpha_2 \xleftarrow{f^{m_2}(x_i[1])} x_i[1]$$

$$y_i[2] \xleftarrow{f^{2^{\frac{w}{2}+1}-2-c}(\alpha_3)} \alpha_3 \xleftarrow{f^{c}(x_i[2])} x_i[2]$$

z_i

$c \longleftarrow 2^{\frac{w}{2}+1} - 2 - m_1 - m_2$

verification: $g(y_i[0] \| y_i[1] \| y_i[2] \| z_i) = z_{i-1}$?

$m = m_1 \| m_2$

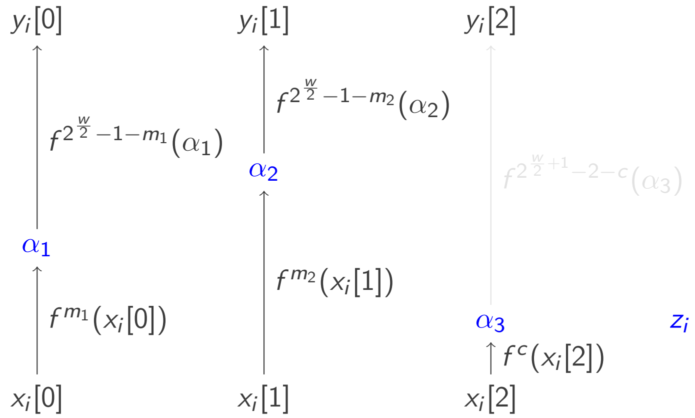
$$
\begin{array}{ccc}
y_i[0] & y_i[1] & y_i[2] \\
\uparrow f^{2^{\frac{w}{2}}-1-m_1}(\alpha_1) & \uparrow f^{2^{\frac{w}{2}}-1-m_2}(\alpha_2) & \\
\color{blue}{\alpha_1} & \color{blue}{\alpha_2} & \uparrow f^{2^{\frac{w}{2}+1}-2-c}(\alpha_3) \\
\uparrow f^{m_1}(x_i[0]) & \uparrow f^{m_2}(x_i[1]) & \color{blue}{\alpha_3} \qquad \color{blue}{z_i} \\
 & & \uparrow f^{c}(x_i[2]) \\
x_i[0] & x_i[1] & x_i[2]
\end{array}
$$

$c \longleftarrow 2^{\frac{w}{2}+1} - 2 - m_1 - m_2$

verification: $g(y_i[0] \| y_i[1] \| y_i[2] \| z_i) = z_{i-1}$?

DKSS Signature Generation and Verification

$m = m_1 \| m_2$



$c \longleftarrow 2^{\frac{w}{2}+1} - 2 - m_1 - m_2$

verification: $g(y_i[0] \| y_i[1] \| y_i[2] \| z_i) = z_{i-1}$?

$m = m_1 \| m_2$

$$
\begin{tikzcd}
y_i[0] & y_i[1] & y_i[2] & \\
\alpha_1 \arrow[u, "f^{2^{\frac{w}{2}}-1-m_1}(\alpha_1)"'] & \alpha_2 \arrow[u, "f^{2^{\frac{w}{2}}-1-m_2}(\alpha_2)"'] & \alpha_3 \arrow[u, "f^{2^{\frac{w}{2}+1}-2-c}(\alpha_3)"'] & z_i \\
x_i[0] \arrow[u, "f^{m_1}(x_i[0])"'] & x_i[1] \arrow[u, "f^{m_2}(x_i[1])"'] & x_i[2] \arrow[u, "f^{c}(x_i[2])"'] &
\end{tikzcd}
$$

$c \longleftarrow 2^{\frac{w}{2}+1} - 2 - m_1 - m_2$

verification: $g(y_i[0] \| y_i[1] \| y_i[2] \| z_i) = z_{i-1}$?

$m = m_1 \| m_2$

$$
\begin{tikzcd}
y_i[0] & y_i[1] & y_i[2] & \\
\color{blue}{\alpha_1} \arrow[u, "f^{2^{\frac{w}{2}}-1-m_1}(\alpha_1)"'] & \color{blue}{\alpha_2} \arrow[u, "f^{2^{\frac{w}{2}}-1-m_2}(\alpha_2)"'] & \color{blue}{\alpha_3} \arrow[u, "f^{2^{\frac{w}{2}+1}-2-c}(\alpha_3)"'] & \color{blue}{z_i} \\
x_i[0] \arrow[u, "f^{m_1}(x_i[0])"'] & x_i[1] \arrow[u, "f^{m_2}(x_i[1])"'] & x_i[2] \arrow[u, "f^{c}(x_i[2])"'] &
\end{tikzcd}
$$

$c \longleftarrow 2^{\frac{w}{2}+1} - 2 - m_1 - m_2$

verification: $g(y_i[0] \| y_i[1] \| y_i[2] \| z_i) = z_{i-1}$?

Correction of the PRNG Specification

- in the original DKSS paper:
- $\mathrm{PRNG}(\psi) = (\mathrm{rand}, \psi') = (f(\psi),\ f(\psi) + \psi + 1 \bmod 2^n)$
- forward secure
- cannot be realized:

- thus: $\mathrm{rand} \leftarrow \mathrm{PRNG}(i, j)$

- in the original DKSS paper:
- $\mathrm{PRNG}(\psi) = (\mathrm{rand}, \psi') = (f(\psi),\ f(\psi) + \psi + 1 \bmod 2^n)$
- forward secure
- cannot be realized:

signature generation $\longrightarrow$

$$\begin{tikzcd}
z_0 & z_1 \arrow[l] & z_2 \arrow[l] & \cdots \arrow[l] & z_{l-1} \arrow[l] & z_l \arrow[l] \arrow[l, bend right, "\times"'] \\
 & Y_1 \arrow[ul] & Y_2 \arrow[ul] & & & Y_l \arrow[ul] \\
 & X_1 \arrow[u] \arrow[r, "\mathrm{PRNG}"] & X_2 \arrow[u] \arrow[r, "\mathrm{PRNG}"] & \cdots & \arrow[r, "\mathrm{PRNG}"] & X_l \arrow[u] \arrow[l, bend left, "\times"]
\end{tikzcd}$$

- thus: $\mathrm{rand} \leftarrow \mathrm{PRNG}(i, j)$

- in the original DKSS paper:
- $\mathrm{PRNG}(\psi) = (\mathrm{rand}, \psi') = (f(\psi),\ f(\psi) + \psi + 1 \bmod 2^n)$
- forward secure
- cannot be realized:

- thus: $\mathrm{rand} \leftarrow \mathrm{PRNG}(i, j)$

- in the original DKSS paper:
- $\mathrm{PRNG}(\psi) = (\mathrm{rand}, \psi') = (f(\psi),\ f(\psi) + \psi + 1 \bmod 2^n)$
- forward secure
- cannot be realized:

signature generation

$$\xrightarrow{\hspace{12cm}}$$

$$\begin{tikzcd}
 & & \bullet & \cdots & \arrow[l, "\times" marking] & \bullet \\
z_0 & z_1 \arrow[l] & z_2 \arrow[l] & \cdots \arrow[l] & z_{l-1} \arrow[l] & z_l \arrow[l] \\
 & Y_1 \arrow[ul] & Y_2 \arrow[ul] & & & Y_l \arrow[ul] \\
 & X_1 \arrow[u] \arrow[r, "\mathrm{PRNG}"] & X_2 \arrow[u] \arrow[r, "\mathrm{PRNG}"] & \cdots & \arrow[r, "\mathrm{PRNG}"] & X_l \arrow[u] \arrow[l, bend left, "\times" marking]
\end{tikzcd}$$

- thus: $\mathrm{rand} \leftarrow \mathrm{PRNG}(i, j)$

- in the original DKSS paper:
- $\mathrm{PRNG}(\psi) = (\mathrm{rand}, \psi') = (f(\psi),\ f(\psi) + \psi + 1 \bmod 2^n)$
- forward secure
- cannot be realized:

signature generation $\longrightarrow$

$$\begin{tikzcd}
 & & \bullet & \cdots & & \bullet \arrow[l, "\times" marking] \\
z_0 & z_1 \arrow[l] & z_2 \arrow[l] & \cdots \arrow[l] & z_{l-1} \arrow[l] & z_l \arrow[l] \\
 & Y_1 \arrow[ul] & Y_2 \arrow[ul] & & & Y_l \arrow[ul] \\
 & X_1 \arrow[u] \arrow[r, "\mathrm{PRNG}"] & X_2 \arrow[u] \arrow[r, "\mathrm{PRNG}"] & \cdots & \arrow[r, "\mathrm{PRNG}"] & X_l \arrow[u] \arrow[l, "\times" marking, bend left]
\end{tikzcd}$$

- thus: $\mathrm{rand} \leftarrow \mathrm{PRNG}(i, j)$

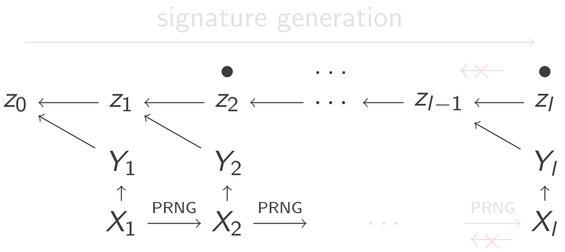
- in the original DKSS paper:
- $\mathrm{PRNG}(\psi) = (\mathrm{rand}, \psi') = (f(\psi),\ f(\psi) + \psi + 1 \bmod 2^n)$
- forward secure
- cannot be realized:

signature generation $\longrightarrow$

$$\begin{tikzcd}
z_0 & z_1 \arrow[l] & z_2 \arrow[l] & \cdots \arrow[l] & z_{l-1} \arrow[l] & z_l \arrow[l] \\
 & Y_1 \arrow[ul] & Y_2 \arrow[ul] & & & Y_l \arrow[ul] \\
 & X_1 \arrow[u] \arrow[r, "\mathrm{PRNG}"] & X_2 \arrow[u] \arrow[r, "\mathrm{PRNG}"] & \cdots \arrow[rr, "\mathrm{PRNG}"] & & X_l \arrow[u]
\end{tikzcd}$$

- thus: $\mathrm{rand} \leftarrow \mathrm{PRNG}(i, j)$

- in the original DKSS paper:
- $\mathrm{PRNG}(\psi) = (\mathrm{rand}, \psi') = (f(\psi),\ f(\psi) + \psi + 1 \bmod 2^n)$
- forward secure
- cannot be realized:



- thus: $\mathrm{rand} \leftarrow \mathrm{PRNG}(i, j)$

Correction of the PRNG Specification

- in the original DKSS paper:
- $\mathrm{PRNG}(\psi) = (\mathrm{rand}, \psi') = (f(\psi),\ f(\psi) + \psi + 1 \bmod 2^n)$
- forward secure
- cannot be realized:

signature generation

$$\begin{tikzcd}
z_0 & z_1 \arrow[l] & z_2 \arrow[l] & \cdots \arrow[l] & z_{l-1} \arrow[l] & z_l \arrow[l] \\
 & Y_1 \arrow[ul] & Y_2 \arrow[ul] & & & Y_l \arrow[ul] \\
 & X_1 \arrow[u] \arrow[r, "\mathrm{PRNG}"] & X_2 \arrow[u] \arrow[r, "\mathrm{PRNG}"] & \cdots \arrow[rr, "\mathrm{PRNG}"] & & X_l \arrow[u]
\end{tikzcd}$$

- thus: $\mathrm{rand} \leftarrow \mathrm{PRNG}(i, j)$

- in the original DKSS paper:
- $\mathrm{PRNG}(\psi) = (\mathrm{rand}, \psi') = (f(\psi),\ f(\psi) + \psi + 1 \bmod 2^n)$
- forward secure
- cannot be realized:

$$
\begin{tikzcd}
 & & \bullet & \cdots & & \bullet \\
z_0 & z_1 \arrow[l] & z_2 \arrow[l] & \cdots \arrow[l] & z_{l-1} \arrow[l] & z_l \arrow[l] \\
 & Y_1 \arrow[ul] & Y_2 \arrow[ul] & & & Y_l \arrow[ul] \\
 & X_1 \arrow[u] \arrow[r, "\mathrm{PRNG}"] & X_2 \arrow[u] \arrow[r, "\mathrm{PRNG}"] & \cdots \arrow[rr, "\mathrm{PRNG}"] & & X_l \arrow[u]
\end{tikzcd}
$$

signature generation $\longrightarrow$

- thus: $\mathrm{rand} \leftarrow \mathrm{PRNG}(i, j)$

- in the original DKSS paper:
- $\mathrm{PRNG}(\psi) = (\mathrm{rand}, \psi') = (f(\psi),\ f(\psi) + \psi + 1 \bmod 2^n)$
- forward secure
- cannot be realized:

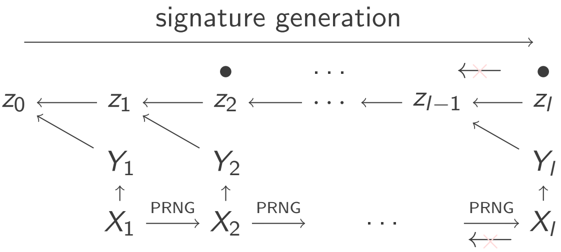
signature generation $\longrightarrow$

$$
\begin{tikzcd}
 & & \bullet & \cdots & \nleftarrow & \bullet \\
z_0 & z_1 \arrow[l] & z_2 \arrow[l] & \cdots \arrow[l] & z_{l-1} \arrow[l] & z_l \arrow[l] \\
 & Y_1 \arrow[ul] & Y_2 \arrow[ul] & & & Y_l \arrow[ul] \\
 & X_1 \arrow[u] \arrow[r, "\mathrm{PRNG}"] & X_2 \arrow[u] \arrow[r, "\mathrm{PRNG}"] & \cdots & \arrow[r, "\mathrm{PRNG}", swap, "\nleftarrow"] & X_l \arrow[u]
\end{tikzcd}
$$

- thus: $\mathrm{rand} \leftarrow \mathrm{PRNG}(i, j)$

Correction of the PRNG Specification

- in the original DKSS paper:
- $\mathrm{PRNG}(\psi) = (\mathrm{rand}, \psi') = (f(\psi),\ f(\psi) + \psi + 1 \bmod 2^n)$
- forward secure
- cannot be realized:



- thus: $\mathrm{rand} \leftarrow \mathrm{PRNG}(i, j)$

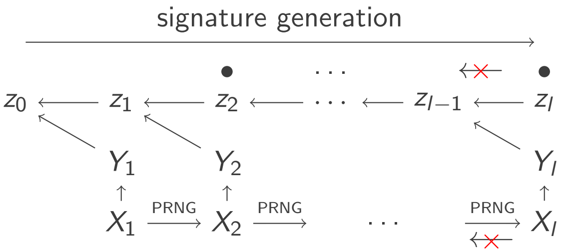
- in the original DKSS paper:
- $\mathrm{PRNG}(\psi) = (\mathrm{rand}, \psi') = (f(\psi),\ f(\psi) + \psi + 1 \bmod 2^n)$
- forward secure
- cannot be realized:

$$\xrightarrow{\text{signature generation}}$$

$$\begin{tikzcd}
 & & \bullet & \cdots & & \bullet \arrow[l, "\times" marking] \\
z_0 & z_1 \arrow[l] & z_2 \arrow[l] & \cdots \arrow[l] & z_{l-1} \arrow[l] & z_l \arrow[l] \\
 & Y_1 \arrow[ul] & Y_2 \arrow[ul] & & & Y_l \arrow[ul] \\
 & X_1 \arrow[u] \arrow[r, "\text{PRNG}"] & X_2 \arrow[u] \arrow[r, "\text{PRNG}"] & \cdots & {} \arrow[r, "\text{PRNG}"] & X_l \arrow[u] \arrow[l, shift left=2, "\times" marking]
\end{tikzcd}$$

- thus: $\mathrm{rand} \leftarrow \mathrm{PRNG}(i, j)$

- in the original DKSS paper:
- $\mathrm{PRNG}(\psi) = (\mathrm{rand}, \psi') = (f(\psi),\ f(\psi) + \psi + 1 \bmod 2^n)$
- forward secure
- cannot be realized:



- thus: $\mathrm{rand} \leftarrow \mathrm{PRNG}(i, j)$

$$
\begin{tikzcd}
z_0 & z_1 \arrow[l] & z_2 \arrow[l] & \cdots \arrow[l] & z_{r-1} \arrow[l] & z_r \arrow[l] & z'_0 \arrow[l] & z'_1 \arrow[l] & z'_2 \arrow[l] & \cdots \arrow[l] & z'_{r-1} \arrow[l] & z'_r \arrow[l] \\
 & Y_1 \arrow[ul] & Y_2 \arrow[ul] & & & Y_n \arrow[ul] & Y'_0 \arrow[ul] & Y'_1 \arrow[ul] & Y'_2 \arrow[ul] & & & Y'_r \arrow[ul] \\
 & X_1 \arrow[u] & X_2 \arrow[u] & & & X_r \arrow[u] & X'_0 \arrow[u] & X'_1 \arrow[u] & X'_2 \arrow[u] & & & X'_r \arrow[u]
\end{tikzcd}
$$

- advantage: faster computation
- disadvantage: more memory

Multiple Chains

$$
\begin{tikzcd}
z_0 & z_1 \arrow[l] & z_2 \arrow[l] & \cdots \arrow[l] & z_{r-1} \arrow[l] & z_r \arrow[l] & z'_0 \arrow[l] & z'_1 \arrow[l] & z'_2 \arrow[l] & \cdots \arrow[l] & z'_{r-1} \arrow[l] & z'_r \arrow[l] \\
 & Y_1 \arrow[ul] & Y_2 \arrow[ul] & & & Y_n \arrow[ul] & Y'_0 \arrow[ul] & Y'_1 \arrow[ul] & Y'_2 \arrow[ul] & & & Y'_r \arrow[ul] \\
 & X_1 \arrow[u] & X_2 \arrow[u] & & & X_r \arrow[u] & X'_0 \arrow[u] & X'_1 \arrow[u] & X'_2 \arrow[u] & & & X'_r \arrow[u]
\end{tikzcd}
$$

- advantage: faster computation
- disadvantage: more memory

Platform

- Tmote Sky WSN platform
- MSP430 16-bit Microcontroller
- 10 KB RAM
- 48 KB flash memory
- 8 MHz CPU speed
- OS: Contiki OS

Platform

- Tmote Sky WSN platform
- MSP430 16-bit Microcontroller
- 10 KB RAM
- 48 KB flash memory
- 8 MHz CPU speed
- OS: Contiki OS

Platform

- Tmote Sky WSN platform
- MSP430 16-bit Microcontroller
- 10 KB RAM
- 48 KB flash memory
- 8 MHz CPU speed
- OS: Contiki OS

Platform

- Tmote Sky WSN platform
- MSP430 16-bit Microcontroller
- 10 KB RAM
- 48 KB flash memory
- 8 MHz CPU speed
- OS: Contiki OS

- Tmote Sky WSN platform
- MSP430 16-bit Microcontroller
- 10 KB RAM
- 48 KB flash memory
- 8 MHz CPU speed
- OS: Contiki OS

Platform

- Tmote Sky WSN platform
- MSP430 16-bit Microcontroller
- 10 KB RAM
- 48 KB flash memory
- 8 MHz CPU speed
- OS: Contiki OS

Block Cipher Choices

- AES
  - fast
  - block size = 128 bit (needed: 80 bit)
  - needs tables (S-Box and MixColumn)
  - unquestioned security
- XXTEA
  - slower
  - block size = 96 bit possible
  - no tables
  - questionable security

Block Cipher Choices

- AES
  - fast
  - block size = 128 bit (needed: 80 bit)
  - needs tables (S-Box and MixColumn)
  - unquestioned security
- XXTEA
  - slower
  - block size = 96 bit possible
  - no tables
  - questionable security

Block Cipher Choices

- AES
  - fast
  - block size = 128 bit (needed: 80 bit)
  - needs tables (S-Box and MixColumn)
  - unquestioned security
- XXTEA
  - slower
  - block size = 96 bit possible
  - no tables
  - questionable security

Block Cipher Choices

- AES
  - fast
  - block size = 128 bit (needed: 80 bit)
  - needs tables (S-Box and MixColumn)
  - unquestioned security
- XXTEA
  - slower
  - block size = 96 bit possible
  - no tables
  - questionable security

Block Cipher Choices

- AES
  - fast
  - block size = 128 bit (needed: 80 bit)
  - needs tables (S-Box and MixColumn)
  - unquestioned security
- XXTEA
  - slower
  - block size = 96 bit possible
  - no tables
  - questionable security

Block Cipher Choices

- AES
  - fast
  - block size = 128 bit (needed: 80 bit)
  - needs tables (S-Box and MixColumn)
  - unquestioned security
- XXTEA
  - slower
  - block size = 96 bit possible
  - no tables
  - questionable security

Block Cipher Choices

- AES
  - fast
  - block size = 128 bit (needed: 80 bit)
  - needs tables (S-Box and MixColumn)
  - unquestioned security
- XXTEA
  - slower
  - block size = 96 bit possible
  - no tables
  - questionable security

Block Cipher Choices

- AES
  - fast
  - block size = 128 bit (needed: 80 bit)
  - needs tables (S-Box and MixColumn)
  - unquestioned security
- XXTEA
  - slower
  - block size = 96 bit possible
  - no tables
  - questionable security

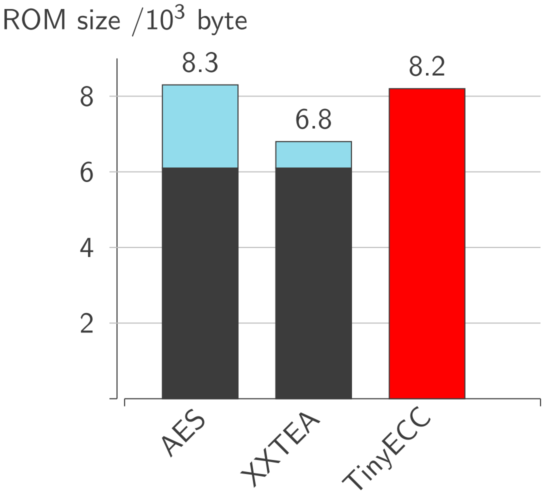
Block Cipher Choices

- AES
  - fast
  - block size = 128 bit (needed: 80 bit)
  - needs tables (S-Box and MixColumn)
  - unquestioned security
- XXTEA
  - slower
  - block size = 96 bit possible
  - no tables
  - questionable security

Block Cipher Choices

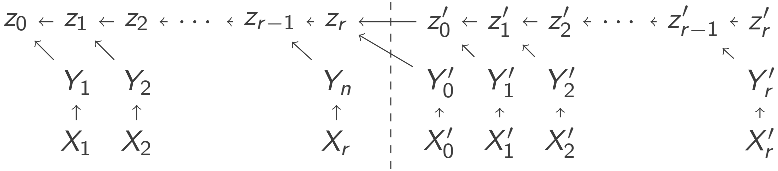
- AES
  - fast
  - block size = 128 bit (needed: 80 bit)
  - needs tables (S-Box and MixColumn)
  - unquestioned security
- XXTEA
  - slower
  - block size = 96 bit possible
  - no tables
  - questionable security

Code Sizes



- time-memory tradeoff employing multiple chains
- $l = 1024$
- AES

	1 chain	8 chains
Sign. Gen. time	5045 ms	4611 ms
Private Key	174 Bytes	924 Bytes



Multiple Chains

- time-memory tradeoff employing multiple chains
- $l = 1024$
- AES

	1 chain	8 chains
Sign. Gen. time	5045 ms	4611 ms
Private Key	174 Bytes	924 Bytes

$$
\begin{array}{cccccccccccccccccccccc}
z_0 & \leftarrow & z_1 & \leftarrow & z_2 & \leftarrow & \cdots & \leftarrow & z_{r-1} & \leftarrow & z_r & \longleftarrow & z'_0 & \leftarrow & z'_1 & \leftarrow & z'_2 & \leftarrow & \cdots & \leftarrow & z'_{r-1} & \leftarrow & z'_r \\
& \nwarrow & & \nwarrow & & & & & & \nwarrow & & \nwarrow & & \nwarrow & & \nwarrow & & & & & & \nwarrow & \\
& & Y_1 & & Y_2 & & & & & & Y_n & & Y'_0 & & Y'_1 & & Y'_2 & & & & & & Y'_r \\
& & \uparrow & & \uparrow & & & & & & \uparrow & & \uparrow & & \uparrow & & \uparrow & & & & & & \uparrow \\
& & X_1 & & X_2 & & & & & & X_r & & X'_0 & & X'_1 & & X'_2 & & & & & & X'_r
\end{array}
$$

- time-memory tradeoff employing multiple chains
- $l = 1024$
- AES

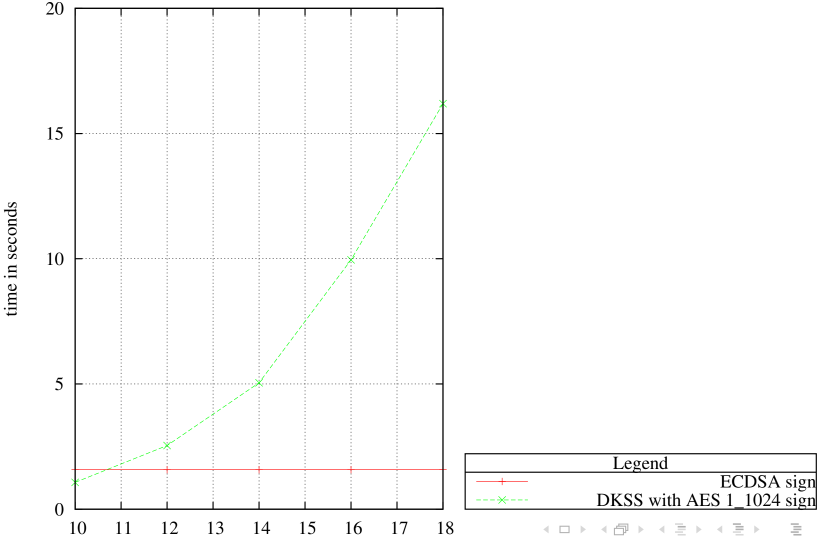
	1 chain	8 chains
Sign. Gen. time	5045 ms	4611 ms
Private Key	174 Bytes	924 Bytes

$$\begin{array}{ccccccccccccccccccccc}
z_0 & \leftarrow & z_1 & \leftarrow & z_2 & \leftarrow & \cdots & \leftarrow & z_{r-1} & \leftarrow & z_r & \longleftarrow & z'_0 & \leftarrow & z'_1 & \leftarrow & z'_2 & \leftarrow & \cdots & \leftarrow & z'_{r-1} \leftarrow z'_r \\
\nwarrow & & \nwarrow & & & & & & \nwarrow & & \nwarrow & & & \nwarrow & & \nwarrow & & & & \nwarrow & \\
 & Y_1 & & Y_2 & & & & & & Y_n & & & Y'_0 & & Y'_1 & & Y'_2 & & & & Y'_r \\
 & \uparrow & & \uparrow & & & & & & \uparrow & & & \uparrow & & \uparrow & & \uparrow & & & & \uparrow \\
 & X_1 & & X_2 & & & & & & X_r & & & X'_0 & & X'_1 & & X'_2 & & & & X'_r
\end{array}$$

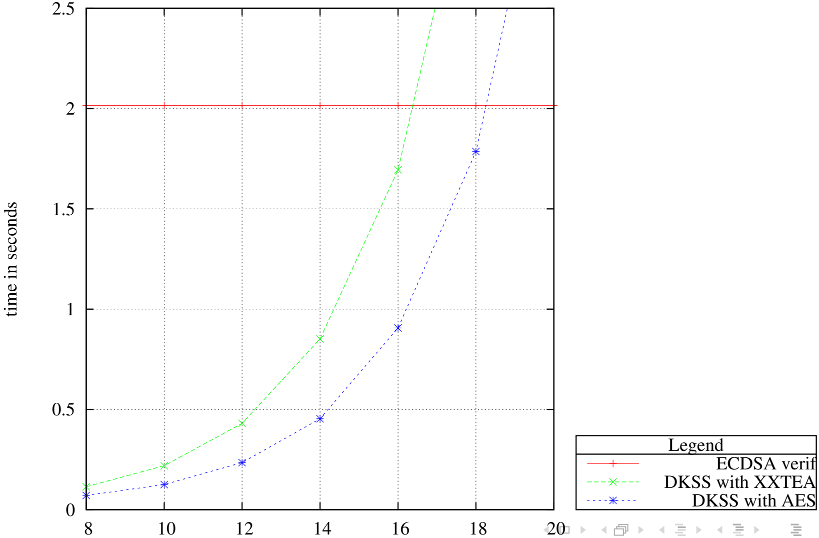
Comparison of AES and XXTEA Running Times

		Tmote Sky node	
Message bit length w		8 bits	16 bits
XXTEA	Sign	0.383	5.770
	Verify	0.098	1.394
AES	Sign	0.279	4.135
	Verify	0.071	1.007

Signature Generation Times



Signature Verfication Times



Conclusion

- compact implementation possible
- speed predictions from original paper not met
- narrow application conditions because of message size
- → integration with broadcast protocol difficult
- most efficient public key signature scheme for short messages
- speedup through AES coprocessor!

Conclusion

- compact implementation possible
- speed predictions from original paper not met
- narrow application conditions because of message size
- $\rightarrow$ integration with broadcast protocol difficult
- most efficient public key signature scheme for short messages
- speedup through AES coprocessor!

Conclusion

- compact implementation possible
- speed predictions from original paper not met
- narrow application conditions because of message size
- $\rightarrow$ integration with broadcast protocol difficult
- most efficient public key signature scheme for short messages
- speedup through AES coprocessor!

Conclusion

- compact implementation possible
- speed predictions from original paper not met
- narrow application conditions because of message size
- $\rightarrow$ integration with broadcast protocol difficult
- most efficient public key signature scheme for short messages
- speedup through AES coprocessor!

Conclusion

- compact implementation possible
- speed predictions from original paper not met
- narrow application conditions because of message size
- $\rightarrow$ integration with broadcast protocol difficult
- most efficient public key signature scheme for short messages
- speedup through AES coprocessor!

Conclusion

- compact implementation possible
- speed predictions from original paper not met
- narrow application conditions because of message size
- $\rightarrow$ integration with broadcast protocol difficult
- most efficient public key signature scheme for short messages
- speedup through AES coprocessor!

- Thank You!