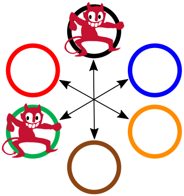
Towards Standardization of Threshold Schemes for Cryptographic Primitives at NIST

Luís Brandão

Joint work with: Apostol Vassilev, Nicky Mouha, Michael Davidson



The red dancing devil is from clker.com/clipart-13643.html

National Institute of Standards and Technology (Gaithersburg MD, USA)

Presentation at ICMC19

International Cryptographic Module Conference
May 16, 2019 @ Vancouver, Canada

Outline

Outline

Should we share a secret?

openclipart.org/detail/76603

Should we share a secret?

openclipart.org/detail/76603

"*Three may keep a secret* "

(In: *"Poor Richard's Almanack."* Benjamin Franklin, 1735) [Sau34]

"*Two may keep counsel* "

(In: *"Romeo and Juliet."* William Shakespeare, 1597) [Sha97]

"*For three may kepe counseil* "

(In: *The Ten Commandments of Love*. Geoffrey Chaucer, 1340–1400) [Cha00]

Should we share a secret?

openclipart.org/detail/76603

"*Three may keep a secret, if two of them are dead.*"

(In: *"Poor Richard's Almanack."* Benjamin Franklin, 1735) [Sau34]

"*Two may keep counsel, putting one away.*"

(In: *"Romeo and Juliet."* William Shakespeare, 1597) [Sha97]

"*For three may kepe counseil if twain be away!*"

(In: *The Ten Commandments of Love*. Geoffrey Chaucer, 1340–1400) [Cha00]

Should we share a secret?

Proverbial wisdom tells us to be careful

openclipart.org/detail/76603

"*Three may keep a secret, if two of them are dead.*"

(In: *"Poor Richard's Almanack."* Benjamin Franklin, 1735) [Sau34]

∗/mw02322/Benjamin-Franklin.jpg

"*Two may keep counsel, putting one away.*"

(In: *"Romeo and Juliet."* William Shakespeare, 1597) [Sha97]

∗/mw11574/William-Shakespeare.jpg

"*For three may kepe counseil if twain be away!*"

(In: *The Ten Commandments of Love*. Geoffrey Chaucer, 1340–1400) [Cha00]

∗/mw01262/Geoffrey-Chaucer.jpg

∗ = https://collectionimages.npg.org.uk/large/

Should we share a secret?

Proverbial wisdom tells us to be careful

openclipart.org/detail/76603

"*Three may keep a secret, if two of them are dead.*"

(In: *"Poor Richard's Almanack."* Benjamin Franklin, 1735) [Sau34]

*/mw02322/Benjamin-Franklin.jpg

"*Two may keep counsel, putting one away.*"

(In: *"Romeo and Juliet."* William Shakespeare, 1597) [Sha97]

*/mw11574/William-Shakespeare.jpg

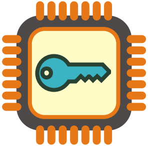
"*For three may kepe counseil if twain be away!*"

(In: *The Ten Commandments of Love*. Geoffrey Chaucer, 1340–1400) [Cha00]

*/mw01262/Geoffrey-Chaucer.jpg

* = https://collectionimages.npg.org.uk/large/

This is specially relevant for *secret keys* in modern cryptography.



openclipart.org/detail/101407

Should we share a secret?

Proverbial wisdom tells us to be careful

openclipart.org/detail/76603

"*Three may keep a secret, if two of them are dead.*"

(In: *"Poor Richard's Almanack."* Benjamin Franklin, 1735) [Sau34]

*/mw02322/Benjamin-Franklin.jpg

"*Two may keep counsel, putting one away.*"

(In: *"Romeo and Juliet."* William Shakespeare, 1597) [Sha97]

*/mw11574/William-Shakespeare.jpg

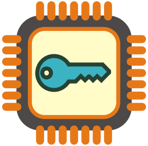
"*For three may kepe counseil if twain be away!*"

(In: *The Ten Commandments of Love*. Geoffrey Chaucer, 1340–1400) [Cha00]

*/mw01262/Geoffrey-Chaucer.jpg

* = https://collectionimages.npg.org.uk/large/

This is specially relevant for *secret keys* in modern cryptography.



openclipart.org/detail/101407

Cryptography relies on:

- secrecy, correctness, availability ... of cryptographic **keys**

Should we share a secret?

Proverbial wisdom tells us to be careful

openclipart.org/detail/76603

"*Three may keep a secret, if two of them are dead.*"
(In: *"Poor Richard's Almanack."* Benjamin Franklin, 1735) [Sau34]

*/mw02322/Benjamin-Franklin.jpg

"*Two may keep counsel, putting one away.*"
(In: *"Romeo and Juliet."* William Shakespeare, 1597) [Sha97]

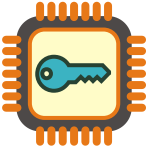
*/mw11574/William-Shakespeare.jpg

"*For three may kepe counseil if twain be away!*"
(In: *The Ten Commandments of Love*. Geoffrey Chaucer, 1340–1400) [Cha00]

*/mw01262/Geoffrey-Chaucer.jpg

* = https://collectionimages.npg.org.uk/large/

This is specially relevant for *secret keys* in modern cryptography.



openclipart.org/detail/101407

Cryptography relies on:

- secrecy, correctness, availability ... of cryptographic **keys**
- **implementations** that use **keys** to **operate** an algorithm

Should we share a secret?

Proverbial wisdom tells us to be careful

openclipart.org/detail/76603

"*Three may keep a secret, if two of them are dead.*"

(In: *"Poor Richard's Almanack."* Benjamin Franklin, 1735) [Sau34]

*/mw02322/Benjamin-Franklin.jpg

"*Two may keep counsel, putting one away.*"

(In: *"Romeo and Juliet."* William Shakespeare, 1597) [Sha97]

*/mw11574/William-Shakespeare.jpg

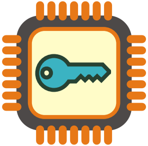
"*For three may kepe counseil if twain be away!*"

(In: *The Ten Commandments of Love*. Geoffrey Chaucer, 1340–1400) [Cha00]

*/mw01262/Geoffrey-Chaucer.jpg

* = https://collectionimages.npg.org.uk/large/

This is specially relevant for *secret keys* in modern cryptography.



openclipart.org/detail/101407

Cryptography relies on:

- secrecy, correctness, availability ... of cryptographic **keys**
- **implementations** that use **keys** to **operate** an algorithm

Crypto can be affected by vulnerabilities!

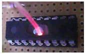
Crypto can be affected by vulnerabilities!

Attacks can exploit differences between ideal vs. real **implementations**

Crypto can be affected by vulnerabilities!

Attacks can exploit differences between ideal vs. real **implementations**

"Bellcore attack" (1997)
[BDL97]



[SH07]

Cold-boot attacks (2009)
[HSH+09]

[Don13]

Heartbleed bug (2014)
[DLK+14]

heartbleed.com

"ZigBee Chain reaction" (2017)
[RSWO17]

[RSWO17]

Meltdown & Spectre (2017)
[LSG+18, KGG+18]

meltdownattack.com

Foreshadow (2018)
[BMW+18, WBM+18]

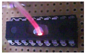
foreshadowattack.eu

Microarchitectural Data Sampling (2019)
[MDS19]

mdsattacks.com

Crypto can be affected by vulnerabilities!

Attacks can exploit differences between ideal vs. real **implementations**

"Bellcore attack" (1997)
[BDL97]

[SH07]

Cold-boot attacks (2009)
[HSH+09]

[Don13]

Heartbleed bug (2014)
[DLK+14]
heartbleed.com

"ZigBee Chain reaction" (2017)
[RSWO17]

[RSWO17]

Meltdown & Spectre (2017)
[LSG+18, KGG+18]
meltdownattack.com

Foreshadow (2018)
[BMW+18, WBM+18]
foreshadowattack.eu

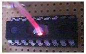
Microarchitectural Data Sampling (2019)
[MDS19]
mdsattacks.com

Also, **operators** of cryptographic implementations can go rogue

Crypto can be affected by vulnerabilities!

Attacks can exploit differences between ideal vs. real **implementations**

"Bellcore attack" (1997) [BDL97]



[SH07]

Cold-boot attacks (2009) [HSH+09]

[Don13]

Heartbleed bug (2014) [DLK+14]

heartbleed.com

"ZigBee Chain reaction" (2017) [RSWO17]

[RSWO17]

Meltdown & Spectre (2017) [LSG+18, KGG+18]

meltdownattack.com

Foreshadow (2018) [BMW+18, WBM+18]

foreshadowattack.eu

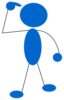
Microarchitectural Data Sampling (2019) [MDS19]

mdsattacks.com

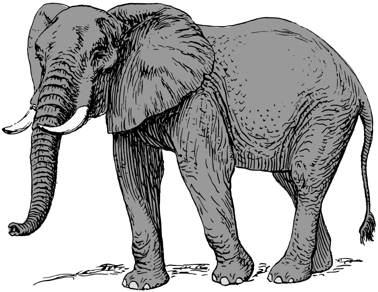
Also, **operators** of cryptographic implementations can go rogue

How can we address single-points of failure?

*question-2.html



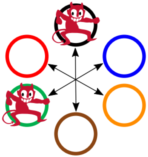
*4296.html



*colored-elephant.html

* = clker.com/clipart-

The threshold approach

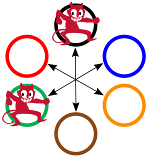


The red dancing devil is from
clker.com/clipart-13643.html

The threshold approach

At high-level:

use redundancy & diversity to mitigate the *compromise* of up to a threshold number (f-out-of-n) of components

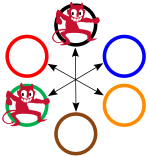


The red dancing devil is from clker.com/clipart-13643.html

The threshold approach

At high-level:

use redundancy & diversity to mitigate the *compromise* of up to a threshold number (f-out-of-n) of components



The red dancing devil is from clker.com/clipart-13643.html

The intuitive aim:

improve security

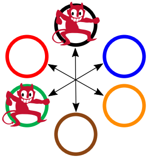
vs.

a non-threshold scheme

The threshold approach

At high-level:

use redundancy & diversity to mitigate the *compromise* of up to a threshold number (f-out-of-n) of components



The intuitive aim:

improve security

vs.

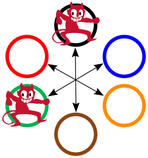
a non-threshold scheme

NIST-CSD wants to standardize
threshold schemes for cryptographic primitives

The threshold approach

At high-level:

use redundancy & diversity to mitigate the *compromise* of up to a threshold number (f-out-of-n) of components



The red dancing devil is from clker.com/clipart-13643.html

The intuitive aim:

improve security

vs.

a non-threshold scheme

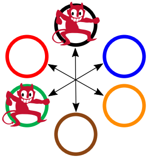
NIST-CSD wants to standardize
threshold schemes for cryptographic primitives

Potential primitives: signing, decryption, enciphering, key-generation, ...

The threshold approach

At high-level:

use redundancy & diversity to mitigate the *compromise* of up to a threshold number (f-out-of-n) of components



The red dancing devil is from clker.com/clipart-13643.html

The intuitive aim:

improve security

vs.

a non-threshold scheme

NIST-CSD wants to standardize
threshold schemes for cryptographic primitives

Potential primitives: signing, decryption, enciphering, key-generation, ...

Some properties:

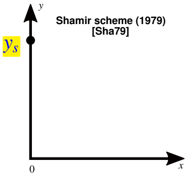
- **withstands** several *compromised* components;
- **needs** several *uncompromised* components;
- **prevents** secret keys from being in one place;
- **enhances** resistance against side-channel attacks; ...

Secret Sharing Schemes (a starting point)

Split a secret key into n secret “shares” for storage at rest.

Secret Sharing Schemes (a starting point)

Split a secret key into n secret "shares" for storage at rest.

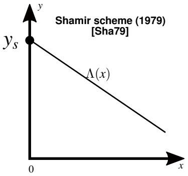


Example 2-out-of-n secret sharing

- The secret y_s is placed in the y-axis;

Secret Sharing Schemes (a starting point)

Split a secret key into n secret "shares" for storage at rest.

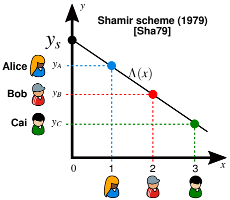


Example 2-out-of-n secret sharing

- The secret y_s is placed in the y-axis;
- A random line Λ is drawn crossing the secret;

Secret Sharing Schemes (a starting point)

Split a secret key into n secret "shares" for storage at rest.

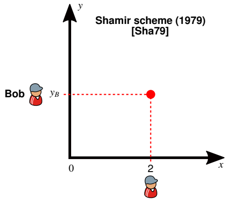


Example 2-out-of-n secret sharing

- The secret y_s is placed in the y-axis;
- A random line Λ is drawn crossing the secret;
- Each share is a point $(\Lambda(i), i)$ in the line Λ;

Secret Sharing Schemes (a starting point)

Split a secret key into n secret "shares" for storage at rest.



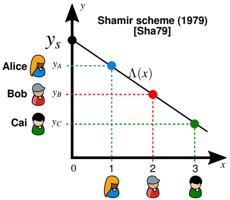
Example 2-out-of-n secret sharing

- The secret y_s is placed in the y-axis;
- A random line Λ is drawn crossing the secret;
- Each share is a point $(\Lambda(i), i)$ in the line Λ;

Each share alone has no information about the secret.

Secret Sharing Schemes (a starting point)

Split a secret key into n secret "shares" for storage at rest.



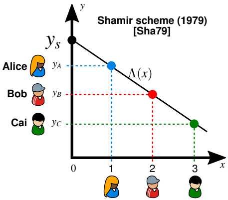
Example 2-out-of-n secret sharing

- The secret y_s is placed in the y-axis;
- A random line Λ is drawn crossing the secret;
- Each share is a point $(\Lambda(i), i)$ in the line Λ;

Each share alone has no information about the secret.
Any pair of shares allows recovering the secret

Secret Sharing Schemes (a starting point)

Split a secret key into n secret “shares” for storage at rest.



Humanoid cliparts: clker.com/clipart-*.html
Alice: *=2478
Bob: *=2482
Cai: *=2479

Example 2-out-of-n secret sharing

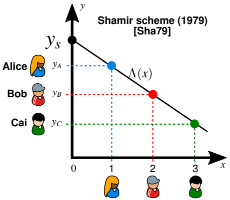
- The secret y_s is placed in the y-axis;
- A random line Λ is drawn crossing the secret;
- Each share is a point $(\Lambda(i), i)$ in the line Λ;

Each share alone has no information about the secret.
Any pair of shares allows recovering the secret

But how to avoid recombining the key when the key is needed by an algorithm?

Secret Sharing Schemes (a starting point)

Split a secret key into n secret "shares" for storage at rest.



Humanoid cliparts:
clker.com/clipart-*.html
Alice: *=2478
Bob: *=2482
Cai: *=2479

Example 2-out-of-n secret sharing

- The secret y_s is placed in the y-axis;
- A random line Λ is drawn crossing the secret;
- Each share is a point $(\Lambda(i), i)$ in the line Λ;

Each share alone has no information about the secret.
Any pair of shares allows recovering the secret

But how to avoid recombining the key when the key is needed by an algorithm?

Use threshold schemes for cryptographic primitives (next)

Goal(s) for this presentation

Overview the NIST effort towards standardization of threshold schemes

clker.com/clipart-purple-mountain.html

Goal(s) for this presentation

Overview the NIST effort towards standardization of threshold schemes

1. Convey high-dimensionality of the threshold space

clker.com/clipart-purple-mountain.html

Goal(s) for this presentation

Overview the NIST effort towards standardization of threshold schemes

1. Convey high-dimensionality of the threshold space
2. Describe the steps so far and ahead

clker.com/clipart-purple-mountain.html

Goal(s) for this presentation

Overview the NIST effort towards standardization of threshold schemes

1. Convey high-dimensionality of the threshold space
2. Describe the steps so far and ahead
3. Motivate feedback and engagement from stakeholders

clker.com/clipart-purple-mountain.html

Outline

A simple example: RSA signature (or decryption) [RSA78]

A simple example: RSA signature (or decryption) [RSA78]

Conventional scheme $(k = n = 1)$

A 3-out-of-3 threshold scheme $(k = n = 3)$

A simple example: RSA signature (or decryption) [RSA78]

Conventional scheme $(k = n = 1)$

- KeyGen
- Sign
- Verify

A 3-out-of-3 threshold scheme $(k = n = 3)$

- KeyGen
- Sign
- Verify

A simple example: RSA signature (or decryption) [RSA78]

Conventional scheme $(k = n = 1)$

- KeyGen
  - Public Modulus: $N = p \cdot q$
  - Secret SignKey: d
  - Public VerKey: $e \;\; (= d^{-1} \pmod{\phi})$
- Sign(m): $\sigma = m^d \pmod{N}$
- Verify(σ, m): $\sigma^e \stackrel{?}{=} m \pmod{N}$

A 3-out-of-3 threshold scheme $(k = n = 3)$

- KeyGen
- Sign
- Verify

A simple example: RSA signature (or decryption) [RSA78]

Conventional scheme $(k = n = 1)$

- KeyGen (by signer):
  - Public Modulus: $N = p \cdot q$
  - Secret SignKey: d
  - Public VerKey: $e \quad (= d^{-1} \pmod{\phi})$
- Sign(m): $\sigma = m^d \pmod{N}$
- Verify(σ, m): $\sigma^e \stackrel{?}{=} m \pmod{N}$

A 3-out-of-3 threshold scheme $(k = n = 3)$

- KeyGen (by dealer):
  - Same N, d, e
  - SubKeys: $d_1, d_2, d_3 \; : \; d_1 + d_2 + d_3 = d \pmod{\phi}$
- Sign
- Verify

A simple example: RSA signature (or decryption) [RSA78]

Conventional scheme $(k = n = 1)$

- KeyGen (by signer):
  - Public Modulus: $N = p \cdot q$
  - Secret SignKey: d
  - Public VerKey: $e \ \ (= d^{-1} \text{ (mod } \phi))$
- Sign(m): $\sigma = m^{d} \text{ (mod } N)$
- Verify(σ, m): $\sigma^{e} \stackrel{?}{=} m \text{ (mod } N)$

A 3-out-of-3 threshold scheme $(k = n = 3)$

- KeyGen (by dealer):
  - Same N, d, e
  - SubKeys: $d_1, d_2, d_3 \ : \ d_1 + d_2 + d_3 = d \text{ (mod } \phi)$
- Sign(m): { separate: $s_i = m^{d_i} \text{ (mod } N) : i = 1, 2, 3$
  combine: $\sigma = s_1 \cdot s_2 \cdot s_3 \text{ (mod } N)$ }
- Verify

A simple example: RSA signature (or decryption) [RSA78]

Conventional scheme $(k = n = 1)$

- KeyGen (by signer):
  - Public Modulus: $N = p \cdot q$
  - Secret SignKey: d
  - Public VerKey: $e \ \ (= d^{-1} \pmod{\phi})$
- Sign(m): $\sigma = m^d \pmod{N}$
- Verify(σ, m): $\sigma^e \stackrel{?}{=} m \pmod{N}$

A 3-out-of-3 threshold scheme $(k = n = 3)$

- KeyGen (by dealer):
  - Same N, d, e
  - SubKeys: $d_1, d_2, d_3 \ : \ d_1 + d_2 + d_3 = d \pmod{\phi}$
- Sign(m): $\left\{ \begin{array}{l} \text{separate: } s_i = m^{d_i} \pmod{N} : i = 1, 2, 3 \\ \text{combine: } \sigma = s_1 \cdot s_2 \cdot s_3 \pmod{N} \end{array} \right\}$
- Verify(σ, m): $\sigma^e \stackrel{?}{=} m \pmod{N}$

A simple example: RSA signature (or decryption) [RSA78]

Conventional scheme $(k = n = 1)$

- KeyGen (by signer):
  - Public Modulus: $N = p \cdot q$
  - Secret SignKey: d
  - Public VerKey: $e \ \ (= d^{-1} \pmod{\phi})$
- Sign(m): $\sigma = m^{d} \pmod{N}$
- Verify(σ, m): $\sigma^{e} =^{?} m \pmod{N}$

A 3-out-of-3 threshold scheme $(k = n = 3)$

- KeyGen (by dealer):
  - Same N, d, e
  - SubKeys: $d_1, d_2, d_3 \ : \ d_1 + d_2 + d_3 = d \pmod{\phi}$
- Sign(m): $\{$ separate: $s_i = m^{d_i} \pmod{N} : i = 1, 2, 3$
  combine: $\sigma = s_1 \cdot s_2 \cdot s_3 \pmod{N}$ $\}$
- Verify(σ, m): $\sigma^{e} =^{?} m \pmod{N}$

About this threshold scheme:

SignKey d not recombined;

A simple example: RSA signature (or decryption) [RSA78]

Conventional scheme $(k = n = 1)$

- KeyGen (by signer):
  - Public Modulus: $N = p \cdot q$
  - Secret SignKey: d
  - Public VerKey: $e \quad (= d^{-1} \pmod{\phi})$
- Sign(m): $\sigma = m^d \pmod{N}$
- Verify(σ, m): $\sigma^e \stackrel{?}{=} m \pmod{N}$

A 3-out-of-3 threshold scheme $(k = n = 3)$

- KeyGen (by dealer):
  - Same N, d, e
  - SubKeys: $d_1, d_2, d_3 \ : \ d_1 + d_2 + d_3 = d \pmod{\phi}$
- Sign(m): { separate: $s_i = m^{d_i} \pmod{N} : i = 1, 2, 3$
  combine: $\sigma = s_1 \cdot s_2 \cdot s_3 \pmod{N}$ }
- Verify(σ, m): $\sigma^e \stackrel{?}{=} m \pmod{N}$

About this threshold scheme:

SignKey d not recombined; can *reshare* d leaving e fixed;

A simple example: RSA signature (or decryption) [RSA78]

Conventional scheme $(k = n = 1)$

- KeyGen (by signer):
  - Public Modulus: $N = p \cdot q$
  - Secret SignKey: d
  - Public VerKey: $e \ \ (= d^{-1} \text{ (mod } \phi))$
- Sign(m): $\sigma = m^d$ (mod N)
- Verify(σ, m): $\sigma^e =^? m$ (mod N)

A 3-out-of-3 threshold scheme $(k = n = 3)$

- KeyGen (by dealer):
  - Same N, d, e
  - SubKeys: $d_1, d_2, d_3 \ : \ d_1 + d_2 + d_3 = d$ (mod ϕ)
- Sign(m): { separate: $s_i = m^{d_i}$ (mod N) $: i = 1, 2, 3$
  combine: $\sigma = s_1 \cdot s_2 \cdot s_3$ (mod N) }
- Verify(σ, m): $\sigma^e =^? m$ (mod N)

About this threshold scheme:

SignKey d not recombined; can *reshare* d leaving e fixed; same σ;

A simple example: RSA signature (or decryption) [RSA78]

Conventional scheme $(k = n = 1)$

- KeyGen (by signer):
  - Public Modulus: $N = p \cdot q$
  - Secret SignKey: d
  - Public VerKey: $e \ \ (= d^{-1} \ (\text{mod } \phi))$
- Sign(m): $\sigma = m^d \ (\text{mod } N)$
- Verify(σ, m): $\sigma^e =^? m \ (\text{mod } N)$

A 3-out-of-3 threshold scheme $(k = n = 3)$

- KeyGen (by dealer):
  - Same N, d, e
  - SubKeys: $d_1, d_2, d_3 \ : \ d_1 + d_2 + d_3 = d \ (\text{mod } \phi)$
- Sign(m): $\{$ separate: $s_i = m^{d_i} \ (\text{mod } N) \ : i = 1, 2, 3$

  combine: $\sigma = s_1 \cdot s_2 \cdot s_3 \ (\text{mod } N) \ \}$
- Verify(σ, m): $\sigma^e =^? m \ (\text{mod } N)$

About this threshold scheme:

SignKey d not recombined; can *reshare* d leaving e fixed; same σ; efficient!

A simple example: RSA signature (or decryption) [RSA78]

Conventional scheme $(k = n = 1)$

- KeyGen (by signer):
  - Public Modulus: $N = p \cdot q$
  - Secret SignKey: d
  - Public VerKey: $e \quad (= d^{-1} \pmod{\phi})$
- Sign(m): $\sigma = m^d \pmod{N}$
- Verify(σ, m): $\sigma^e \stackrel{?}{=} m \pmod{N}$

A 3-out-of-3 threshold scheme $(k = n = 3)$

- KeyGen (by dealer):
  - Same N, d, e
  - SubKeys: $d_1, d_2, d_3 \ : \ d_1 + d_2 + d_3 = d \pmod{\phi}$
- Sign(m): { separate: $s_i = m^{d_i} \pmod{N} : i = 1, 2, 3$
  combine: $\sigma = s_1 \cdot s_2 \cdot s_3 \pmod{N}$ }
- Verify(σ, m): $\sigma^e \stackrel{?}{=} m \pmod{N}$

About this threshold scheme:

SignKey d not recombined; can *reshare* d leaving e fixed; same σ; efficient!

Facilitating setting: $\exists$ dealer;

A simple example: RSA signature (or decryption) [RSA78]

Conventional scheme $(k = n = 1)$

- KeyGen (by signer):
  - Public Modulus: $N = p \cdot q$
  - Secret SignKey: d
  - Public VerKey: $e \quad (= d^{-1} \pmod{\phi})$
- Sign(m): $\sigma = m^d \pmod{N}$
- Verify(σ, m): $\sigma^e \stackrel{?}{=} m \pmod{N}$

A 3-out-of-3 threshold scheme $(k = n = 3)$

- KeyGen (by dealer):
  - Same N, d, e
  - SubKeys: $d_1, d_2, d_3 \ : \ d_1 + d_2 + d_3 = d \pmod{\phi}$
- Sign(m): { separate: $s_i = m^{d_i} \pmod{N} : i = 1, 2, 3$

  combine: $\sigma = s_1 \cdot s_2 \cdot s_3 \pmod{N}$ }
- Verify(σ, m): $\sigma^e \stackrel{?}{=} m \pmod{N}$

About this threshold scheme:

SignKey d not recombined; can *reshare* d leaving e fixed; same σ; efficient!

Facilitating setting: $\exists$ dealer; $\exists$ homomorphism;

A simple example: RSA signature (or decryption) [RSA78]

Conventional scheme $(k = n = 1)$

- KeyGen (by signer):
  - Public Modulus: $N = p \cdot q$
  - Secret SignKey: d
  - Public VerKey: $e \quad (= d^{-1} \pmod{\phi})$
- Sign(m): $\sigma = m^d \pmod{N}$
- Verify(σ, m): $\sigma^e =^? m \pmod{N}$

A 3-out-of-3 threshold scheme $(k = n = 3)$

- KeyGen (by dealer):
  - Same N, d, e
  - SubKeys: $d_1, d_2, d_3 \;:\; d_1 + d_2 + d_3 = d \pmod{\phi}$
- Sign(m): { separate: $s_i = m^{d_i} \pmod{N} : i = 1, 2, 3$
  combine: $\sigma = s_1 \cdot s_2 \cdot s_3 \pmod{N}$ }
- Verify(σ, m): $\sigma^e =^? m \pmod{N}$

About this threshold scheme:

SignKey d not recombined; can *reshare* d leaving e fixed; same σ; efficient!

Facilitating setting: $\exists$ dealer; $\exists$ homomorphism; all parties learn m.

A simple example: RSA signature (or decryption) [RSA78]

Conventional scheme $(k = n = 1)$

- KeyGen (by signer):
  - Public Modulus: $N = p \cdot q$
  - Secret SignKey: d
  - Public VerKey: $e \;\; (= d^{-1} \text{ (mod } \phi))$
- Sign(m): $\sigma = m^d$ (mod N)
- Verify(σ, m): $\sigma^e =^? m$ (mod N)

A 3-out-of-3 threshold scheme $(k = n = 3)$

- KeyGen (by dealer):
  - Same N, d, e
  - SubKeys: $d_1, d_2, d_3 \;:\; d_1 + d_2 + d_3 = d$ (mod ϕ)
- Sign(m): $\{$ separate: $s_i = m^{d_i}$ (mod N) $: i = 1, 2, 3$
  combine: $\sigma = s_1 \cdot s_2 \cdot s_3$ (mod N) $\}$
- Verify(σ, m): $\sigma^e =^? m$ (mod N)

About this threshold scheme:

SignKey d not recombined; can *reshare* d leaving e fixed; same σ; efficient!

Facilitating setting: $\exists$ dealer; $\exists$ homomorphism; all parties learn m.

Not fault-tolerant: a single sub-signer can boycott a correct signing.

A simple example: RSA signature (or decryption) [RSA78]

Conventional scheme $(k = n = 1)$

- KeyGen (by signer):
  - Public Modulus: $N = p \cdot q$
  - Secret SignKey: d
  - Public VerKey: $e \quad (= d^{-1} \pmod{\phi})$
- Sign(m): $\sigma = m^d \pmod{N}$
- Verify(σ, m): $\sigma^e \stackrel{?}{=} m \pmod{N}$

A 3-out-of-3 threshold scheme $(k = n = 3)$

- KeyGen (by dealer):
  - Same N, d, e
  - SubKeys: $d_1, d_2, d_3 \ : \ d_1 + d_2 + d_3 = d \pmod{\phi}$
- Sign(m): { separate: $s_i = m^{d_i} \pmod{N} : i = 1, 2, 3$
  combine: $\sigma = s_1 \cdot s_2 \cdot s_3 \pmod{N}$ }
- Verify(σ, m): $\sigma^e \stackrel{?}{=} m \pmod{N}$

About this threshold scheme:

SignKey d not recombined; can *reshare* d leaving e fixed; same σ; efficient!

Facilitating setting: $\exists$ dealer; $\exists$ homomorphism; all parties learn m.

Not fault-tolerant: a single sub-signer can boycott a correct signing.

Can other threshold schemes be implemented:

?

A simple example: RSA signature (or decryption) [RSA78]

Conventional scheme $(k = n = 1)$

- KeyGen (by signer):
  - Public Modulus: $N = p \cdot q$
  - Secret SignKey: d
  - Public VerKey: $e \quad (= d^{-1} \pmod{\phi})$
- Sign(m): $\sigma = m^d \pmod{N}$
- Verify(σ, m): $\sigma^e =^? m \pmod{N}$

A 3-out-of-3 threshold scheme $(k = n = 3)$

- KeyGen (by dealer):
  - Same N, d, e
  - SubKeys: $d_1, d_2, d_3 \; : \; d_1 + d_2 + d_3 = d \pmod{\phi}$
- Sign(m): { separate: $s_i = m^{d_i} \pmod{N} : i = 1, 2, 3$
  combine: $\sigma = s_1 \cdot s_2 \cdot s_3 \pmod{N}$ }
- Verify(σ, m): $\sigma^e =^? m \pmod{N}$

About this threshold scheme:

SignKey d not recombined; can *reshare* d leaving e fixed; same σ; efficient!

Facilitating setting: $\exists$ dealer; $\exists$ homomorphism; all parties learn m.

Not fault-tolerant: a single sub-signer can boycott a correct signing.

Can other threshold schemes be implemented:
$\nexists$ dealer, $\nexists$ homomorphisms, secret-shared m, withstanding f malicious signers**?**

A simple example: RSA signature (or decryption) [RSA78]

Conventional scheme $(k = n = 1)$

- KeyGen (by signer):
  - Public Modulus: $N = p \cdot q$
  - Secret SignKey: d
  - Public VerKey: $e \quad (= d^{-1} \text{ (mod } \phi))$
- Sign(m): $\sigma = m^d$ (mod N)
- Verify(σ, m): $\sigma^e =^? m$ (mod N)

A 3-out-of-3 threshold scheme $(k = n = 3)$

- KeyGen (by dealer):
  - Same N, d, e
  - SubKeys: $d_1, d_2, d_3 \ : \ d_1 + d_2 + d_3 = d$ (mod ϕ)
- Sign(m): { separate: $s_i = m^{d_i}$ (mod N) : $i = 1, 2, 3$
  combine: $\sigma = s_1 \cdot s_2 \cdot s_3$ (mod N) }
- Verify(σ, m): $\sigma^e =^? m$ (mod N)

About this threshold scheme:

SignKey d not recombined; can *reshare* d leaving e fixed; same σ; efficient!

Facilitating setting: $\exists$ dealer; $\exists$ homomorphism; all parties learn m.

Not fault-tolerant: a single sub-signer can boycott a correct signing.

Can other threshold schemes be implemented:
$\nexists$ dealer, $\nexists$ homomorphisms, secret-shared m, withstanding f malicious signers**?**

Yes, using threshold cryptography

A simple example: RSA signature (or decryption) [RSA78]

Conventional scheme $(k = n = 1)$

- KeyGen (by signer):
  - Public Modulus: $N = p \cdot q$
  - Secret SignKey: d
  - Public VerKey: $e \quad (= d^{-1} \pmod{\phi})$
- Sign(m): $\sigma = m^d \pmod{N}$
- Verify(σ, m): $\sigma^e =^? m \pmod{N}$

A 3-out-of-3 threshold scheme $(k = n = 3)$

- KeyGen (by dealer):
  - Same N, d, e
  - SubKeys: $d_1, d_2, d_3 \;:\; d_1 + d_2 + d_3 = d \pmod{\phi}$
- Sign(m): { separate: $s_i = m^{d_i} \pmod{N} : i = 1, 2, 3$
  combine: $\sigma = s_1 \cdot s_2 \cdot s_3 \pmod{N}$ }
- Verify(σ, m): $\sigma^e =^? m \pmod{N}$

About this threshold scheme:

SignKey d not recombined; can *reshare* d leaving e fixed; same σ; efficient!

Facilitating setting: $\exists$ dealer; $\exists$ homomorphism; all parties learn m.

Not fault-tolerant: a single sub-signer can boycott a correct signing.

Can other threshold schemes be implemented:
$\nexists$ dealer, $\nexists$ homomorphisms, secret-shared m, withstanding f malicious signers**?**

Yes, using threshold cryptography (with more complicated schemes)

What do thresholds k and f mean?

What do thresholds k and f mean?

3-out-of-3 decryption:

- **Availability:** 3 nodes needed to decrypt
- **Key secrecy:** okay while 1 share is secret

clker.com/clipart-encryption.html

What do thresholds k and f mean?

3-out-of-3 decryption:

- **Availability:** 3 nodes needed to decrypt $(k = 3,\ f = 0)$
- **Key secrecy:** okay while 1 share is secret

clker.com/clipart-encryption.html

What do thresholds k and f mean?

3-out-of-3 decryption:

clker.com/clipart-encryption.html

- **Availability:** 3 nodes needed to decrypt ($k = 3$, $f = 0$)
- **Key secrecy:** okay while 1 share is secret $(k = 1, f = 2)$

What do thresholds k and f mean?

3-out-of-3 decryption:

clker.com/clipart-encryption.html

- **Availability:** 3 nodes needed to decrypt ($k = 3$, $f = 0$)
- **Key secrecy:** okay while 1 share is secret ($k = 1$, $f = 2$)

(Each security property has its own k and f)

What do thresholds k and f mean?

3-out-of-3 decryption:

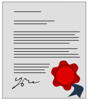
- **Availability:** 3 nodes needed to decrypt ($k = 3$, $f = 0$)
- **Key secrecy:** okay while 1 share is secret ($k = 1$, $f = 2$)

clker.com/clipart-encryption.html

(Each security property has its own k and f)

2-out-of-3 signature:

- **Availability:** 2 nodes needed to sign
- **Key secrecy:** okay while 2 shares are secret


clker.com/clipart-3712.html

What do thresholds k and f mean?

3-out-of-3 decryption:

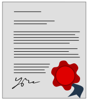
- **Availability:** 3 nodes needed to decrypt ($k = 3$, $f = 0$)
- **Key secrecy:** okay while 1 share is secret ($k = 1$, $f = 2$)

clker.com/clipart-encryption.html

(Each security property has its own k and f)

2-out-of-3 signature:

- **Availability:** 2 nodes needed to sign $(k = 2, f = 1)$
- **Key secrecy:** okay while 2 shares are secret



clker.com/clipart-3712.html

What do thresholds k and f mean?

3-out-of-3 decryption:

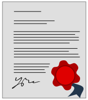
- **Availability:** 3 nodes needed to decrypt ($k = 3$, $f = 0$)
- **Key secrecy:** okay while 1 share is secret ($k = 1$, $f = 2$)

clker.com/clipart-encryption.html

(Each security property has its own k and f)

2-out-of-3 signature:

- **Availability:** 2 nodes needed to sign ($k = 2, f = 1$)
- **Key secrecy:** okay while 2 shares are secret ($k = 2, f = 1$)



clker.com/clipart-3712.html

What do thresholds k and f mean?

3-out-of-3 decryption:

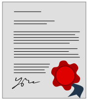
- **Availability:** 3 nodes needed to decrypt ($k = 3$, $f = 0$)
- **Key secrecy:** okay while 1 share is secret ($k = 1$, $f = 2$)

clker.com/clipart-encryption.html

(Each security property has its own k and f)

2-out-of-3 signature:

- **Availability:** 2 nodes needed to sign ($k = 2, f = 1$)
- **Key secrecy:** okay while 2 shares are secret ($k = 2, f = 1$)



clker.com/clipart-3712.html

But does any of these schemes improve security?

(compared with a non-threshold scheme ($n = k = 1, f = 0$))

What do thresholds k and f mean?

3-out-of-3 decryption:

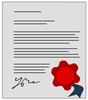
- **Availability:** 3 nodes needed to decrypt ($k = 3,\ f = 0$)
- **Key secrecy:** okay while 1 share is secret ($k = 1,\ f = 2$)

clker.com/clipart-encryption.html

(Each security property has its own k and f)

2-out-of-3 signature:

- **Availability:** 2 nodes needed to sign ($k = 2, f = 1$)
- **Key secrecy:** okay while 2 shares are secret ($k = 2, f = 1$)



clker.com/clipart-3712.html

But does any of these schemes improve security?

(compared with a non-threshold scheme ($n = k = 1, f = 0$))

It depends: "k-out-of-n" or "f-out-of-n" is not a sufficient characterization for a comprehensive security assertion

What do thresholds k and f mean?

3-out-of-3 decryption:

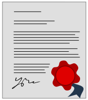
- **Availability:** 3 nodes needed to decrypt ($k = 3,\ f = 0$)
- **Key secrecy:** okay while 1 share is secret ($k = 1,\ f = 2$)

clker.com/clipart-encryption.html

(Each security property has its own k and f)

2-out-of-3 signature:

- **Availability:** 2 nodes needed to sign ($k = 2, f = 1$)
- **Key secrecy:** okay while 2 shares are secret ($k = 2, f = 1$)



clker.com/clipart-3712.html

But does any of these schemes improve security?

(compared with a non-threshold scheme ($n = k = 1, f = 0$))

It depends: "k-out-of-n" or "f-out-of-n" is not a sufficient characterization for a comprehensive security assertion

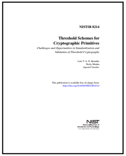
Depends on attack model (e.g., attack surface, ...), system model (e.g., rejuvenations, ...), ...

Outline

NIST Internal Report (NISTIR) 8214

NIST Internal Report (NISTIR) 8214

Threshold Schemes for Cryptographic Primitives — Challenges and Opportunities in Standardization and Validation of Threshold Cryptography. [BMV18] doi:10.6028/NIST.IR.8214


NISTIR 8214

Threshold Schemes for Cryptographic Primitives

Challenges and Opportunities in Standardization and Validation of Threshold Cryptography

Luís T. A. N. Brandão
Nicky Mouha
Apostol Vassilev

This publication is available free of charge from:
https://doi.org/10.6028/NIST.IR.8214

National Institute of Standards and Technology
U.S. Department of Commerce

https://csrc.nist.gov/publications/detail/nistir/8214/final

NIST Internal Report (NISTIR) 8214

Threshold Schemes for Cryptographic Primitives — Challenges and Opportunities in Standardization and Validation of Threshold Cryptography. [BMV18] doi:10.6028/NIST.IR.8214

The report sets a basis for discussion:

- need to characterize threshold schemes
- need to engage with stakeholders
- need to define criteria for standardization

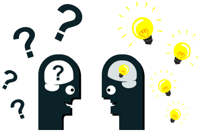
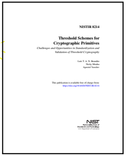


Image adapted from: openclipart.org/detail/283392


NISTIR 8214

Threshold Schemes for Cryptographic Primitives

Challenges and Opportunities in Standardization and Validation of Threshold Cryptography

Luís T. A. N. Brandão
Nicky Mouha
Apostol Vassilev

This publication is available free of charge from:
https://doi.org/10.6028/NIST.IR.8214

NIST
National Institute of Standards and Technology
U.S. Department of Commerce

https://csrc.nist.gov/publications/detail/nistir/8214/final

NIST Internal Report (NISTIR) 8214

Threshold Schemes for Cryptographic Primitives — Challenges and Opportunities in Standardization and Validation of Threshold Cryptography. [BMV18] doi:10.6028/NIST.IR.8214

The report sets a basis for discussion:

- need to <u>characterize</u> threshold schemes
- need to <u>engage</u> with stakeholders
- need to <u>define</u> criteria for standardization

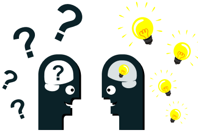
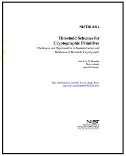


Image adapted from: openclipart.org/detail/283392



NISTIR 8214

Threshold Schemes for Cryptographic Primitives

Challenges and Opportunities in Standardization and Validation of Threshold Cryptography

Luís T. A. N. Brandão
Nicky Mouha
Apostol Vassilev

This publication is available free of charge from: https://doi.org/10.6028/NIST.IR.8214

NIST
National Institute of Standards and Technology
U.S. Department of Commerce

Past timeline:

- 2018-July: Draft online 3 months for public comments
- 2018-October: Received comments from 13 external sources
- 2019-March: Final version online, along with “diff” and received comments

https://csrc.nist.gov/publications/detail/nistir/8214/final

Characterizing threshold schemes

Characterizing threshold schemes

To reflect on a threshold scheme, start by characterizing **4 main features**:

- Kinds of threshold 
- Executing platform 
- Communication interfaces 
- Setup and maintenance  

The cliparts are from openclipart.org/detail/$*$, with $* \in \{71491, 190624, 101407, 161401, 161389\}$

Characterizing threshold schemes

To reflect on a threshold scheme, start by characterizing **4 main features**:

- Kinds of threshold 
- Communication interfaces 
- Executing platform 
- Setup and maintenance 

The cliparts are from openclipart.org/detail/$*$, with $* \in \{71491, 190624, 101407, 161401, 161389\}$

Each feature spans distinct options that affect security in different ways.

Characterizing threshold schemes

To reflect on a threshold scheme, start by characterizing **4 main features**:

- Kinds of threshold
- Communication interfaces
- Executing platform
- Setup and maintenance

The cliparts are from openclipart.org/detail/$*$, with $* \in \{71491, 190624, 101407, 161401, 161389\}$

Each feature spans distinct options that affect security in different ways.

A characterization provides a better context for security assertions.

Characterizing threshold schemes

To reflect on a threshold scheme, start by characterizing **4 main features**:

- Kinds of threshold
- Executing platform
- Communication interfaces
- Setup and maintenance

The cliparts are from openclipart.org/detail/$*$, with $* \in \{71491, 190624, 101407, 161401, 161389\}$

Each feature spans distinct options that affect security in different ways.

A characterization provides a better context for security assertions.

But there are other factors ...

Deployment context

Deployment context

- **Application context.** Should it affect security requirements?

Deployment context

- **Application context.** Should it affect security requirements?
  - signature correctness — may be deferred to client
  - decryption correctness — may require *robust* protocol

clker.com/clipart-3712.html

clker.com/clipart-encryption.html

Deployment context

- **Application context.** Should it affect security requirements?
  - signature correctness — may be deferred to client
  - decryption correctness — may require *robust* protocol

clker.com/clipart-3712.html

clker.com/clipart-encryption.html

clker.com/clipart-10778

- **Conceivable attack types.**
  - Active vs. passive
  - Static vs. adaptive
  - Stealth vs. detected
  - Invasive (physical) vs. non-invasive
  - Side-channel vs. communication interfaces
  - Parallel vs. sequential (wrt attacking nodes)

Deployment context

- **Application context.** Should it affect security requirements?
  - signature correctness — may be deferred to client
  - decryption correctness — may require *robust* protocol

clker.com/clipart-3712.html

clker.com/clipart-encryption.html

- **Conceivable attack types.**

clker.com/clipart-10778

  - Active vs. passive
  - Static vs. adaptive
  - Stealth vs. detected
  - Invasive (physical) vs. non-invasive
  - Side-channel vs. communication interfaces
  - Parallel vs. sequential (wrt attacking nodes)

A threshold scheme **improving** security against an attack in an application **may be powerless or degrade** security for another attack in another application

The validation challenge

The validation challenge

Devise standards of testable and validatable threshold schemes

vs.

devise testing and validation for standardized threshold schemes

The validation challenge

Devise standards of testable and validatable threshold schemes
vs.
devise testing and validation for standardized threshold schemes

Validation is needed in the federal context:

- need to use **validated** implementations [tC96] of **standardized** algorithms
- FIPS 140-2/3 defines, for cryptographic modules, 4 security levels: subsets of applicable security assertions [NIS01]

(FIPS = Federal Information Processing Standards)

Outline

#NTCW2019

NIST Threshold Cryptography Workshop 2019

https://csrc.nist.gov/Events/2019/NTCW19

#NTCW2019

NIST Threshold Cryptography Workshop 2019

March 11–12, 2019 @
NIST Gaithersburg MD, USA

www.nist.gov/image/surfgaithersburgjpg

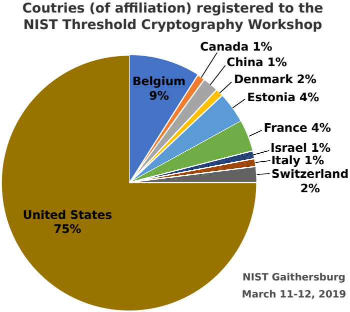
https://csrc.nist.gov/Events/2019/NTCW19

#NTCW2019

NIST Threshold Cryptography Workshop 2019

March 11–12, 2019 @
NIST Gaithersburg MD, USA

www.nist.gov/image/surfgaithersburgjpg



About 80 attendees

https://csrc.nist.gov/Events/2019/NTCW19

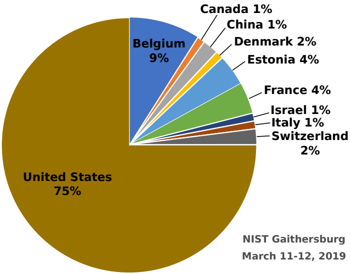
#NTCW2019

NIST Threshold Cryptography Workshop 2019

March 11–12, 2019 @
NIST Gaithersburg MD, USA

www.nist.gov/image/surfgaithersburgjpg

Coutries (of affiliation) registered to the NIST Threshold Cryptography Workshop



About 80 attendees

A platform for open interaction:

- hear about experiences with threshold crypto;
- get to know stakeholders;
- get input to reflect on roadmap and criteria.

https://csrc.nist.gov/Events/2019/NTCW19

Format and content

Format and content

Accepted 15 external submissions:

- 2 panels
- 5 papers
- 8 presentations

Format and content

Accepted 15 external submissions:

- 2 panels
- 5 papers
- 8 presentations

Plus:

- 2 invited keynotes
- 4 NIST talks
- 2 feedback moments

Format and content

Accepted 15 external submissions:

- 2 panels
- 5 papers
- 8 presentations

Plus:

- 2 invited keynotes
- 4 NIST talks
- 2 feedback moments

Videos, papers and presentations online at the NTCW webpage: https://csrc.nist.gov/Events/2019/NTCW19

Format and content

Accepted 15 external submissions:

- 2 panels
- 5 papers
- 8 presentations

Plus:

- 2 invited keynotes
- 4 NIST talks
- 2 feedback moments

Videos, papers and presentations online at the NTCW webpage: https://csrc.nist.gov/Events/2019/NTCW19

Discussion of diverse topics:

- threshold schemes in general (motivation and implementation feasibility);
- NIST standardization of cryptographic primitives
- a post-quantum threshold public-key encryption scheme;
- threshold signatures (adaptive security; elliptic curve digital signature algorithm);
- validation of cryptographic implementations;
- threshold circuit design (tradeoffs, pitfalls, combined attacks, verification tools);
- secret-sharing with leakage resilience;
- distributed symmetric-key encryption;
- applications and experience with threshold cryptography.

Results

Results

A step in *driving an open and transparent process towards standardization of threshold schemes for cryptographic primitives*. (See NISTIR 7977)

Results

A step in *driving an open and transparent process towards standardization of threshold schemes for cryptographic primitives*. (See NISTIR 7977)

Some notes:

- differences in granularity (building blocks vs. full functionalities);
- separation of single-device vs. multi-party;
- importance of envisioning applications;
- stakeholders' willingness to contribute;
- usefulness of explaining rationale (e.g., as complimented for the NISTIR);
- encouragement to move forward.

Results

A step in *driving an open and transparent process towards standardization of threshold schemes for cryptographic primitives*. (See NISTIR 7977)

Some notes:

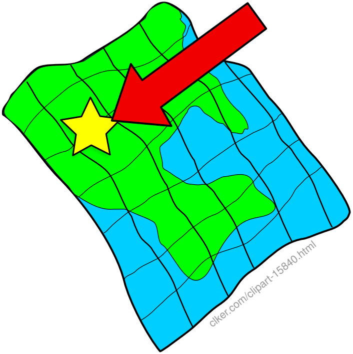
- differences in granularity (building blocks vs. full functionalities);
- separation of single-device vs. multi-party;
- importance of envisioning applications;
- stakeholders' willingness to contribute;
- usefulness of explaining rationale (e.g., as complimented for the NISTIR);
- encouragement to move forward.

These elements are helpful for the next step ... designing a roadmap

Outline

Preliminary roadmap (ongoing)

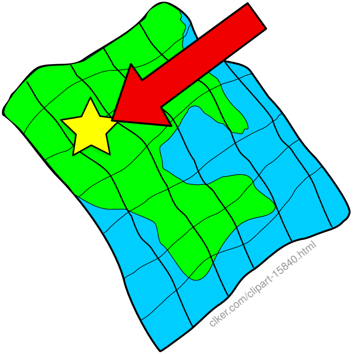
We are writing a draft “preliminary roadmap”



Preliminary roadmap (ongoing)

We are writing a draft “preliminary roadmap”

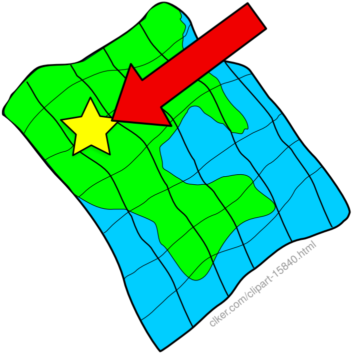
(getting a map; deciding where to go; thinking how to get there)



Preliminary roadmap (ongoing)

We are writing a draft "preliminary roadmap"

(getting a map; deciding where to go; thinking how to get there)

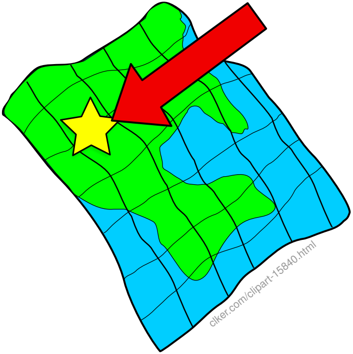


Need: ***mapping layers*** (coordinates) and ***weighing factors***

Preliminary roadmap (ongoing)

We are writing a draft “preliminary roadmap”

(getting a map; deciding where to go; thinking how to get there)



Need: ***mapping layers*** (coordinates) and ***weighing factors***

Disclaimer: the structure suggested in the next slides is still subject to change.

Mapping layers

An abstract layered decomposition of the threshold standardization space

Four layers

$\boxed{\text{Standardization space for threshold schemes for cryptographic primitives}}$

Mapping layers

An abstract layered decomposition of the threshold standardization space

Four layers: domains

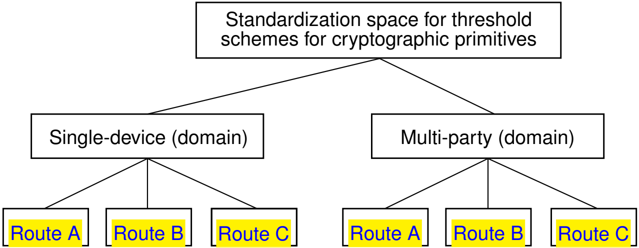
Standardization space for threshold schemes for cryptographic primitives

- Single-device (domain)
- Multi-party (domain)

Mapping layers

An abstract layered decomposition of the threshold standardization space

Four layers: domains, routes

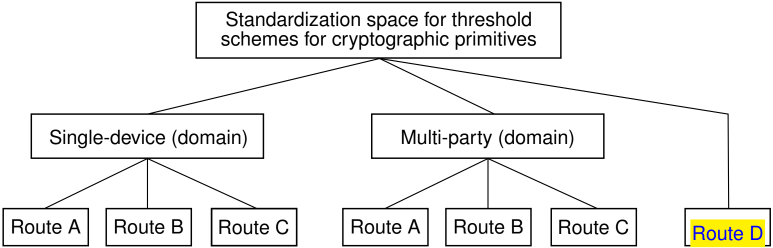


- Route A: simple thresholdization
- Route B: compositional designs
- Route C: new primitives

Mapping layers

An abstract layered decomposition of the threshold standardization space

Four layers: domains, routes

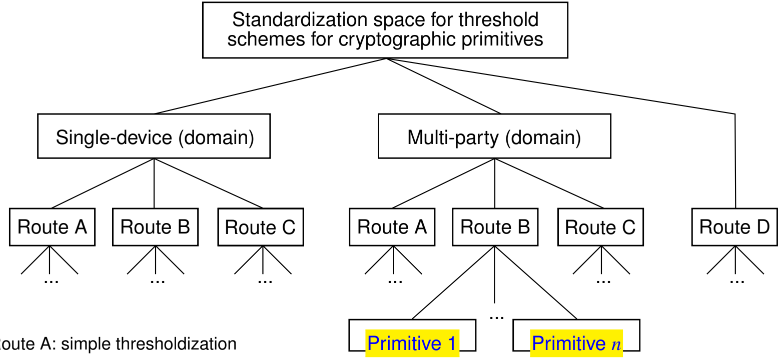


- Route A: simple thresholdization
- Route B: compositional designs
- Route C: new primitives
- Route D: gadgets

Mapping layers

An abstract layered decomposition of the threshold standardization space

Four layers: domains, routes, primitives

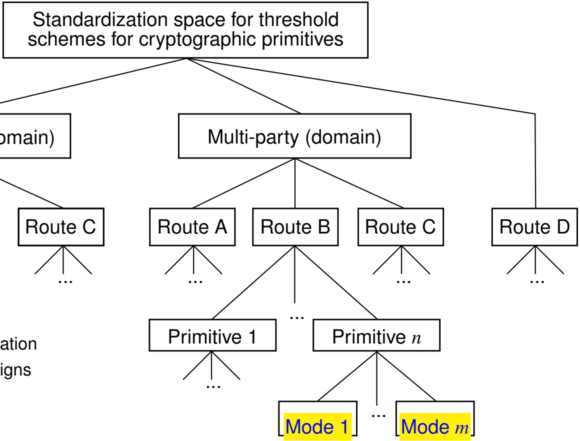


- Route A: simple thresholdization
- Route B: compositional designs
- Route C: new primitives
- Route D: gadgets

Mapping layers

An abstract layered decomposition of the threshold standardization space

Four layers: domains, routes, primitives, modes



- Route A: simple thresholdization
- Route B: compositional designs
- Route C: new primitives
- Route D: gadgets

Some conceived examples

***Primitives* across *routes*:**

***Modes*:**

Some conceived examples

***Primitives* across *routes*:**

- **A:** RSA decryption & signature; Schnorr signature; ECC key-gen; AES (single-device) threshold circuit design against leakage.
- **B:**
- **C:**
- **D:**

***Modes*:**

Some conceived examples

***Primitives* across *routes*:**

- **A:** RSA decryption & signature; Schnorr signature; ECC key-gen;
  AES (single-device) threshold circuit design against leakage.
- **B:** ECDSA signature; RSA key-gen; AES enciphering;
  AES (single-device) threshold circuit against combined attacks.
- **C:**
- **D:**

***Modes*:**

Some conceived examples

***Primitives* across *routes*:**

- **A:** RSA decryption & signature; Schnorr signature; ECC key-gen; AES (single-device) threshold circuit design against leakage.
- **B:** ECDSA signature; RSA key-gen; AES enciphering; AES (single-device) threshold circuit against combined attacks.
- **C:** post-quantum signing & decryption; lightweight-crypto threshold.
- **D:**

***Modes*:**

Some conceived examples

***Primitives* across *routes*:**

- **A:** RSA decryption & signature; Schnorr signature; ECC key-gen;
  AES (single-device) threshold circuit design against leakage.
- **B:** ECDSA signature; RSA key-gen; AES enciphering;
  AES (single-device) threshold circuit against combined attacks.
- **C:** post-quantum signing & decryption; lightweight-crypto threshold.
- **D:** secret sharing; distributed RNG; consensus.

***Modes*:**

Some conceived examples

***Primitives* across *routes*:**

- **A:** RSA decryption & signature; Schnorr signature; ECC key-gen;
  AES (single-device) threshold circuit design against leakage.
- **B:** ECDSA signature; RSA key-gen; AES enciphering;
  AES (single-device) threshold circuit against combined attacks.
- **C:** post-quantum signing & decryption; lightweight-crypto threshold.
- **D:** secret sharing; distributed RNG; consensus.

***Modes*:**

- threshold signature with secret-shared key vs. multi-signature (independent keys);

Some conceived examples

***Primitives* across *routes*:**

- **A:** RSA decryption & signature; Schnorr signature; ECC key-gen; AES (single-device) threshold circuit design against leakage.
- **B:** ECDSA signature; RSA key-gen; AES enciphering; AES (single-device) threshold circuit against combined attacks.
- **C:** post-quantum signing & decryption; lightweight-crypto threshold.
- **D:** secret sharing; distributed RNG; consensus.

***Modes*:**

- threshold signature with secret-shared key vs. multi-signature (independent keys);
- operation on secret-shared plaintext;

Some conceived examples

***Primitives* across *routes*:**

- **A:** RSA decryption & signature; Schnorr signature; ECC key-gen;
  AES (single-device) threshold circuit design against leakage.
- **B:** ECDSA signature; RSA key-gen; AES enciphering;
  AES (single-device) threshold circuit against combined attacks.
- **C:** post-quantum signing & decryption; lightweight-crypto threshold.
- **D:** secret sharing; distributed RNG; consensus.

***Modes*:**

- threshold signature with secret-shared key vs. multi-signature (independent keys);
- operation on secret-shared plaintext;
- honest majority; robust with fault detection;

Some conceived examples

***Primitives* across *routes*:**

- **A:** RSA decryption & signature; Schnorr signature; ECC key-gen;
  AES (single-device) threshold circuit design against leakage.
- **B:** ECDSA signature; RSA key-gen; AES enciphering;
  AES (single-device) threshold circuit against combined attacks.
- **C:** post-quantum signing & decryption; lightweight-crypto threshold.
- **D:** secret sharing; distributed RNG; consensus.

***Modes*:**

- threshold signature with secret-shared key vs. multi-signature (independent keys);
- operation on secret-shared plaintext;
- honest majority; robust with fault detection;
- asynchronous environment.

Some conceived examples

***Primitives* across *routes*:**

- **A:** RSA decryption & signature; Schnorr signature; ECC key-gen;
  AES (single-device) threshold circuit design against leakage.
- **B:** ECDSA signature; RSA key-gen; AES enciphering;
  AES (single-device) threshold circuit against combined attacks.
- **C:** post-quantum signing & decryption; lightweight-crypto threshold.
- **D:** secret sharing; distributed RNG; consensus.

***Modes*:**

- threshold signature with secret-shared key vs. multi-signature (independent keys);
- operation on secret-shared plaintext;
- honest majority; robust with fault detection;
- asynchronous environment.

Not every possible combination needs to be a standardization goal

Weighing factors

The four layers provide a map. But where to look in the map?

openclipart.org/detail/281637

Weighing factors

The four layers provide a map. But where to look in the map?


openclipart.org/detail/281637

- **Application motivations:**

- **Useful features:**

Weighing factors

The four layers provide a map. But where to look in the map?

- **Application motivations:**
  - threshold circuit design in single-device (address side-channel leakage)
  - distribute trust across several operators of crypto primitives*
  - multi-signatures in crypto currencies
  - privacy preserving modes (e.g., secret-shared plaintext)
  - ...

*(emphasis on approved conventional primitives)

- **Useful features:**

Weighing factors

The four layers provide a map. But where to look in the map?

openclipart.org/detail/281637

- **Application motivations:**
  - threshold circuit design in single-device (address side-channel leakage)
  - distribute trust across several operators of crypto primitives*
  - multi-signatures in crypto currencies
  - privacy preserving modes (e.g., secret-shared plaintext)
  - ...

*(emphasis on approved conventional primitives)

- **Useful features:**
  - efficiency and practicality
  - suitability for automated testing
  - ability to rejuvenate components
  - ...

Hereafter

Hereafter

Soon: Draft “preliminary roadmap” asking feedback

Hereafter

Soon: Draft "preliminary roadmap" asking feedback, e.g., on:

- elements within *layers*, *application motivations* and *other factors*
- primitives/modes to focus on (and respective security properties)
- possible elements to adopt/adapt from other standards

Hereafter

Soon: Draft “preliminary roadmap” asking feedback, e.g., on:

- elements within *layers*, *application motivations* and *other factors*
- primitives/modes to focus on (and respective security properties)
- possible elements to adopt/adapt from other standards

Later: separate criteria for separate focuses; calls for contributions

Hereafter

Soon: Draft "preliminary roadmap" asking feedback, e.g., on:

- elements within *layers*, *application motivations* and *other factors*
- primitives/modes to focus on (and respective security properties)
- possible elements to adopt/adapt from other standards

Later: separate criteria for separate focuses; calls for contributions

Example *routes* for calls for contributions:

- algorithms for standardization
- reference implementations and comparisons
- research contributions
- ...

Possibly fit some of these in a 2^{nd} workshop (?)

Outline

Final remarks

Final remarks

- Threshold schemes have potential to address single-points of failure:
  - in technology ... when crypto implementations have vulnerabilities
  - at the human level ... when crypto operators go rogue

Final remarks

- Threshold schemes have potential to address single-points of failure:
  - in technology ... when crypto implementations have vulnerabilities
  - at the human level ... when crypto operators go rogue
- There exist numerous researched threshold schemes

Final remarks

- Threshold schemes have potential to address single-points of failure:
  - in technology ... when crypto implementations have vulnerabilities
  - at the human level ... when crypto operators go rogue
- There exist numerous researched threshold schemes
- It is time to move towards (some) standardization

Final remarks

- Threshold schemes have potential to address single-points of failure:
  - in technology ... when crypto implementations have vulnerabilities
  - at the human level ... when crypto operators go rogue
- There exist numerous researched threshold schemes
- It is time to move towards (some) standardization

We would like to have a process in collaboration with stakeholders!

- Project webpage: https://csrc.nist.gov/Projects/Threshold-Cryptography
- Project email adress: threshold-crypto@nist.gov
- NISTIR 8214: https://csrc.nist.gov/publications/detail/nistir/8214/final
- NTCW webpage: https://csrc.nist.gov/Events/2019/NTCW19
- Forum: https://groups.google.com/a/list.nist.gov/forum/#!forum/tc-forum
  (register for announcements; we can add your email if you send us a request)

- Project webpage: https://csrc.nist.gov/Projects/Threshold-Cryptography
- Project email adress: threshold-crypto@nist.gov
- NISTIR 8214: https://csrc.nist.gov/publications/detail/nistir/8214/final
- NTCW webpage: https://csrc.nist.gov/Events/2019/NTCW19
- Forum: https://groups.google.com/a/list.nist.gov/forum/#!forum/tc-forum
  (register for announcements; we can add your email if you send us a request)

Word cloud based on the NISTIR 8214

Thank you for your attention

- Project webpage: https://csrc.nist.gov/Projects/Threshold-Cryptography
- Project email adress: threshold-crypto@nist.gov
- NISTIR 8214: https://csrc.nist.gov/publications/detail/nistir/8214/final
- NTCW webpage: https://csrc.nist.gov/Events/2019/NTCW19
- Forum: https://groups.google.com/a/list.nist.gov/forum/#!forum/tc-forum
  (register for announcements; we can add your email if you send us a request)

Word cloud based on the NISTIR 8214

Presentation at the International Cryptographic Module Conference
May 16, 2019 @ Vancouver, Canada

luis.brandao@nist.gov

Thank you for your attention

- Project webpage: https://csrc.nist.gov/Projects/Threshold-Cryptography
- Project email adress: threshold-crypto@nist.gov
- NISTIR 8214: https://csrc.nist.gov/publications/detail/nistir/8214/final
- NTCW webpage: https://csrc.nist.gov/Events/2019/NTCW19
- Forum: https://groups.google.com/a/list.nist.gov/forum/#!forum/tc-forum
  (register for announcements; we can add your email if you send us a request)

Word cloud based on the NISTIR 8214

Presentation at the International Cryptographic Module Conference
May 16, 2019 @ Vancouver, Canada

luis.brandao@nist.gov

References

[BB12] L. T. A. N. Brandão and A. N. Bessani. *On the reliability and availability of replicated and rejuvenating systems under stealth attacks and intrusions*. Journal of the Brazilian Computer Society, 18(1):61–80, 2012. DOI:10.1007/s13173-012-0062-x.

[BDL97] D. Boneh, R. A. DeMillo, and R. J. Lipton. *On the Importance of Checking Cryptographic Protocols for Faults*. In W. Fumy (ed.), Advances in Cryptology — EUROCRYPT '97, pages 37–51, Berlin, Heidelberg, 1997. Springer Berlin Heidelberg. DOI:10.1007/3-540-69053-0˙4.

[BMV18] L. T. A. N. Brandão, N. Mouha, and A. Vassilev. *Threshold Schemes for Cryptographic Primitives — Challenges and Opportunities in Standardization and Validation of Threshold Cryptography*. Draft NISTIR 8214, July 2018.

[BMW+18] J. v. Bulck, M. Minkin, O. Weisse, D. Genkin, B. Kasikci, F. Piessens, M. Silberstein, T. F. Wenisch, Y. Yarom, and R. Strackx. *Foreshadow: Extracting the Keys to the Intel SGX Kingdom with Transient Out-of-Order Execution*. In 27th USENIX Security Symposium (USENIX Security 18), page 991–1008, Baltimore, MD, 2018. USENIX Association.

[BN06] M. Bellare and G. Neven. *Multi-signatures in the Plain public-Key Model and a General Forking Lemma*. In Proceedings of the 13th ACM Conference on Computer and Communications Security, CCS '06, pages 390–399, New York, NY, USA, 2006. ACM. DOI:10.1145/1180405.1180453.

[Cha00] G. Chaucer. *The Ten Commandments of Love*, 1340–1400. See "For three may kepe counseil if twain be away!" in the "Secretnesse" stanza of the poem. https://sites.fas.harvard.edu/ chaucer/special/lifemann/love/ten-comm.html. Accessed: July 2018.

[DLK+14] Z. Durumeric, F. Li, J. Kasten, J. Amann, J. Beekman, M. Payer, N. Weaver, D. Adrian, V. Paxson, M. Bailey, and J. A. Halderman. *The Matter of Heartbleed*. In Proceedings of the 2014 Conference on Internet Measurement Conference, IMC '14, pages 475–488, New York, NY, USA, 2014. ACM. DOI:10.1145/2663716.2663755.

[Don13] D. Donzai. *Using Cold Boot Attacks and Other Forensic Techniques in Penetration Tests*, 2013. https://www.ethicalhacker.net/features/root/using-cold-boot-attacks-forensic-techniques-penetration-tests/. Accessed: July 2018.

[Gro16] C. T. Group. *NIST Cryptographic Standards and Guidelines Development Process*. NISTIR 7977, March 2016. DOI:10.6028/NIST.IR.7977.

[HSH+09] J. A. Halderman, S. D. Schoen, N. Heninger, W. Clarkson, W. Paul, J. A. Calandrino, A. J. Feldman, J. Appelbaum, and E. W. Felten. *Lest We Remember: Cold-boot Attacks on Encryption Keys*. Commun. ACM, 52(5):91–98, May 2009. DOI:10.1145/1506409.1506429.

[KGG+18] P. Kocher, D. Genkin, D. Gruss, W. Haas, M. Hamburg, M. Lipp, S. Mangard, T. Prescher, M. Schwarz, and Y. Yarom. *Spectre Attacks: Exploiting Speculative Execution*. ArXiv e-prints, January 2018. arXiv:1801.01203.

[LSG+18] M. Lipp, M. Schwarz, D. Gruss, T. Prescher, W. Haas, S. Mangard, P. Kocher, D. Genkin, Y. Yarom, and M. Hamburg. *Meltdown*. ArXiv e-prints, jan 2018. arXiv:1801.01207.

[MDS19] MDS. *RIDL and Fallout: MDS attacks*, 2019. https://mdsattacks.com/.

[NIS01] NIST. *Security Requirements for Cryptographic Modules, Federal Information Processing Standard (FIPS) 140-2*, 2001. DOI:10.6028/NIST.FIPS.140-2.

[RSA78] R. L. Rivest, A. Shamir, and L. Adleman. *A method for obtaining digital signatures and public-key cryptosystems*. Communications of the ACM, 21(2):120–126, 1978. DOI:10.1145/359340.359342.

[RSWO17] E. Ronen., A. Shamir, A.-O. Weingarten, and C. O'Flynn. *IoT Goes Nuclear: Creating a ZigBee Chain Reaction*. IEEE Symposium on Security and Privacy, pages 195–212, 2017. DOI:10.1109/SP.2017.14.

[Sau34] R. Saunders. *Poor Richard's Almanack — 1735*. Benjamin Franklin, 1734.

[Sch90] C. P. Schnorr. *Efficient Identification and Signatures for Smart Cards*. In G. Brassard (ed.), Advances in Cryptology — CRYPTO' 89 Proceedings, pages 239–252, New York, NY, 1990. Springer New York. DOI:10.1007/0-387-34805-0˙22.

[SH07] J.-M. Schmidt and M. Hutter. *Optical and EM Fault-Attacks on CRT-based RSA: Concrete Results*, pages 61–67. Verlag der Technischen Universität Graz, 2007.

[Sha97] W. Shakespeare. *An excellent conceited Tragedie of Romeo and Juliet*. Printed by John Danter, London, 1597.

[Sha79] A. Shamir. *How to Share a Secret*. Communications of the ACM, 22(11):612–613, Nov 1979. DOI:10.1145/359168.359176.

[Sho00] V. Shoup. *Practical Threshold Signatures*. In B. Preneel (ed.), Advances in Cryptology — EUROCRYPT 2000, pages 207–220, Berlin, Heidelberg, 2000. Springer Berlin Heidelberg. DOI:10.1007/3-540-45539-6˙15.

[tC96] U. S. 104th Congress. *Information Technology Management Reform Act. Public Law 104–106, Section 5131*, 1996. https://www.dol.gov/ocfo/media/regs/ITMRA.pdf.

[WBM+18] O. Weisse, J. v. Bulck, M. Minkin, D. Genkin, B. Kasikci, F. Piessens, M. Silberstein, R. Strackx, T. F. Wenisch, and Y. Yarom. *Foreshadow-NG: Breaking the Virtual Memory Abstraction with Transient Out-of-Order Execution*. Technical Report, 2018.

Extra slides

Next follow some extra slides

Reliability $(\mathcal{R})$ — one metric of security

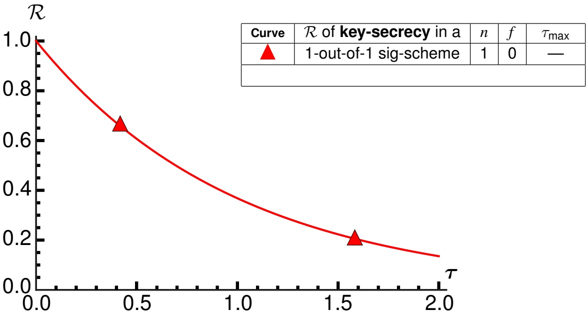
Probability that a security property (e.g., secrecy) never fails during a mission time

Time normalized: $\tau = 1$ is the expected time to failure (ETTF) of a node

Reliability $(\mathcal{R})$ — one metric of security

Probability that a security property (e.g., secrecy) never fails during a mission time

A possible model: each node fails (independently) with constant rate probability

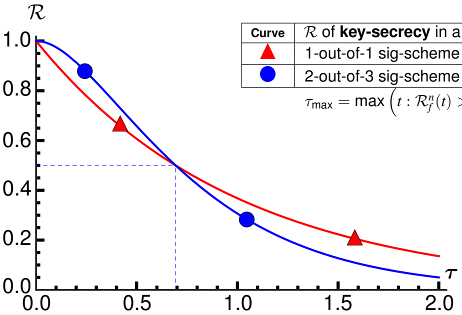


[BB12] Time normalized: $\tau = 1$ is the expected time to failure (ETTF) of a node

Reliability $(\mathcal{R})$ — one metric of security

Probability that a security property (e.g., secrecy) never fails during a mission time

A possible model: each node fails (independently) with constant rate probability



Curve	$\mathcal{R}$ of **key-secrecy** in a	n	f	$\tau_{\max}$
▲	1-out-of-1 sig-scheme	1	0	—
●	2-out-of-3 sig-scheme	3	1	0.693

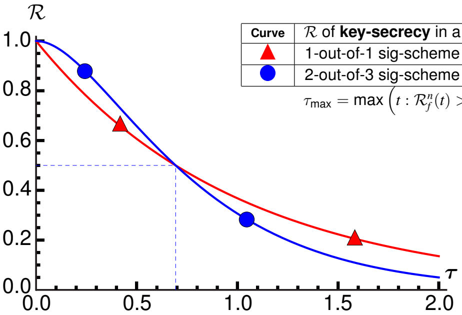
$\tau_{\max} = \max\left(t : \mathcal{R}^n_f(t) > \mathcal{R}^1_0(t)\right)$

[BB12] Time normalized: $\tau = 1$ is the expected time to failure (ETTF) of a node

Reliability $(\mathcal{R})$ — one metric of security

Probability that a security property (e.g., secrecy) never fails during a mission time

A possible model: each node fails (independently) with constant rate probability



Curve	$\mathcal{R}$ of **key-secrecy** in a	n	f	τ_{max}
▲	1-out-of-1 sig-scheme	1	0	—
●	2-out-of-3 sig-scheme	3	1	0.693

$\tau_{\text{max}} = \max\left(t : \mathcal{R}_f^n(t) > \mathcal{R}_0^1(t)\right)$

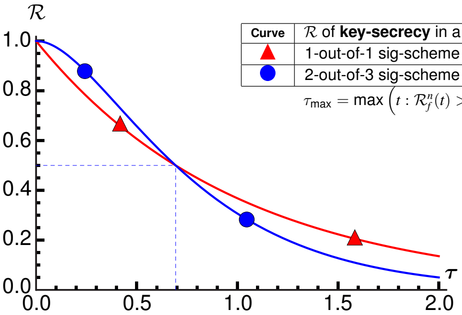
[BB12] Time normalized: $\tau = 1$ is the expected time to failure (ETTF) of a node

Reliability can be degraded when increasing the fault-tolerance threshold f

Reliability $(\mathcal{R})$ — one metric of security

Probability that a security property (e.g., secrecy) never fails during a mission time

A possible model: each node fails (independently) with constant rate probability



Curve	$\mathcal{R}$ of **key-secrecy** in a	n	f	τ_{max}
▲	1-out-of-1 sig-scheme	1	0	—
●	2-out-of-3 sig-scheme	3	1	0.693

$\tau_{\text{max}} = \max\left(t : \mathcal{R}^n_f(t) > \mathcal{R}^1_0(t)\right)$

[BB12] Time normalized: $\tau = 1$ is the expected time to failure (ETTF) of a node

Reliability can be degraded when increasing the fault-tolerance threshold f

Note: rejuvenation of nodes can attenuate the reliability-degradation

Another model

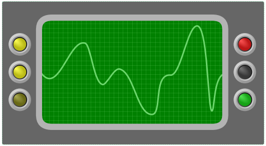
What if all nodes are compromised (e.g., leaky) from the start?

Another model

What if all nodes are compromised (e.g., leaky) from the start?

Threshold scheme may still be effective,
if it increases the cost of exploitation!

(e.g., if exploiting a leakage vulnerability requires exponential number of traces for high-order Differential Power Analysis)



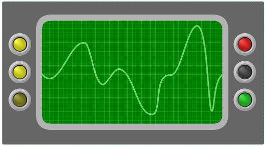
openclipart.org/detail/172330

Another model

What if all nodes are compromised (e.g., leaky) from the start?

Threshold scheme may still be effective, if it increases the cost of exploitation!

(e.g., if exploiting a leakage vulnerability requires exponential number of traces for high-order Differential Power Analysis)



openclipart.org/detail/172330

Challenge questions:

- which models are realistic / match state-of-the-art attacks?
- what concrete parameters (e.g., n) thwart real attacks?

Two hints

Two hints

Robust k-out-of-n Threshold RSA Signature [Sho00]

Two hints

Robust k-out-of-n Threshold RSA Signature [Sho00]

- Works iff $\geq k$ parties are available: **homomorphism** allows combining (slightly tweaked) sub-signatures.

Two hints

Robust k-out-of-n Threshold RSA Signature [Sho00]

- Works iff $\geq k$ parties are available: **homomorphism** allows combining (slightly tweaked) sub-signatures.
- Robust: sub-signers prove (efficient **NIZKP**) correct sub-signatures.

(NIZK = non-interactive zero-knowledge proof of knowledge)

Two hints

Robust k-out-of-n Threshold RSA Signature [Sho00]

- Works iff $\geq k$ parties are available: **homomorphism** allows combining (slightly tweaked) sub-signatures.
- Robust: sub-signers prove (efficient **NIZKP**) correct sub-signatures.

(NIZK = non-interactive zero-knowledge proof of knowledge)

Threshold Schnorr (multi-)signature [BN06]

Two hints

Robust k-out-of-n Threshold RSA Signature [Sho00]

- Works iff $\geq k$ parties are available: **homomorphism** allows combining (slightly tweaked) sub-signatures.
- Robust: sub-signers prove (efficient **NIZKP**) correct sub-signatures.

(NIZK = non-interactive zero-knowledge proof of knowledge)

Threshold Schnorr (multi-)signature [BN06]

- Different public key per signer $\rightarrow$ no dealer, dynamic signer-set

Two hints

Robust k-out-of-n Threshold RSA Signature [Sho00]

- Works iff $\geq k$ parties are available: **homomorphism** allows combining (slightly tweaked) sub-signatures.
- Robust: sub-signers prove (efficient **NIZKP**) correct sub-signatures.

(NIZK = non-interactive zero-knowledge proof of knowledge)

Threshold Schnorr (multi-)signature [BN06]

- Different public key per signer $\rightarrow$ no dealer, dynamic signer-set
- Verifier decides the threshold and knows who signed

Two hints

Robust k-out-of-n Threshold RSA Signature [Sho00]

- Works iff $\geq k$ parties are available: **homomorphism** allows combining (slightly tweaked) sub-signatures.
- Robust: sub-signers prove (efficient **NIZKP**) correct sub-signatures.

(NIZK = non-interactive zero-knowledge proof of knowledge)

Threshold Schnorr (multi-)signature [BN06]

- Different public key per signer $\rightarrow$ no dealer, dynamic signer-set
- Verifier decides the threshold and knows who signed
- DL-based homomorphism $\rightarrow$ size equal to 1 signature

(DL = Discrete-Logarithm)

A DL-based example: threshold Schnorr signature

(DL = Discrete-Logarithm)

A DL-based example: threshold Schnorr signature

(DL = Discrete-Logarithm)

(Next: ignore details — just making comparative remarks)

A DL-based example: threshold Schnorr signature

(DL = Discrete-Logarithm)

(Next: ignore details — just making comparative remarks)

Non-threshold scheme [Sch90]

- Space: G, g (group, generator)
- KeyGen (by signer):
  - Secret SignKey: $x \in Z_q$
  - Public VerKey: $X = g^{-x}$
- Sign$_x(m)$ by signer:
  - $R = g^r$
  - $c =_q H(R||m)$
  - $s =_q r + x \cdot c$
  - output $\sigma = (s, c)$
- Verify$_X(\sigma, m)$:
  - calculate $R = g^s X^c$
  - check $H(R||m) =^? c$

A multi-signature scheme [BN06]

- Space: same G, g
- KeyGen (by parties $i = 1, ..., n$):
  - Secret SignKey: $x_i \in Z_q$
  - Public VerKey: $X_i = g^{x_i}$
- Sign$_{x,L}(m)$ by subset $I \subseteq \{1, ..., n\}$
  - $R = \prod_{i \in I} R_i = \prod_{i \in I} g^{r_i}$
  - $c_i =_q H(X_i||R||I||m)$
  - $s =_q \sum_{i \in L} s_i = \sum_{i \in I}(r_i + x_i c_i)$
  - output $\sigma = (R, s)$
- Verify(σ, m):
  - calculate $c_i = H(X_i||R||M||I||m)$
  - check $g^s =^? R \prod_{i \in I} X_i^{c_i}$

A DL-based example: threshold Schnorr signature

(DL = Discrete-Logarithm)

(Next: ignore details — just making comparative remarks)

Non-threshold scheme [Sch90]

- Space: G, g (group, generator)
- KeyGen (by signer):
  - Secret SignKey: $x \in Z_q$
  - Public VerKey: $X = g^{-x}$
- Sign$_x(m)$ by signer:
  - $R = g^r$
  - $c =_q H(R||m)$
  - $s =_q r + x \cdot c$
  - output $\sigma = (s, c)$
- Verify$_X(\sigma, m)$:
  - calculate $R = g^s X^c$
  - check $H(R||m) =^? c$

A multi-signature scheme [BN06]

- Space: same G, g
- KeyGen (by parties $i = 1, ..., n$):
  - Secret SignKey: $x_i \in Z_q$
  - Public VerKey: $X_i = g^{x_i}$
- Sign$_{x,L}(m)$ by subset $I \subseteq \{1, ..., n\}$
  - $R = \prod_{i \in I} R_i = \prod_{i \in I} g^{r_i}$
  - $c_i =_q H(X_i||R||I||m)$
  - $s =_q \sum_{i \in L} s_i = \sum_{i \in I}(r_i + x_i c_i)$
  - output $\sigma = (R, s)$
- Verify(σ, m):
  - calculate $c_i = H(X_i||R||M||I||m)$
  - check $g^s =^? R \prod_{i \in I} X_i^{c_i}$

A DL-based example: threshold Schnorr signature

(DL = Discrete-Logarithm)

(Next: ignore details — just making comparative remarks)

Non-threshold scheme [Sch90]

- Space: G, g (group, generator)
- KeyGen (by signer):
  - Secret SignKey: $x \in Z_q$
  - Public VerKey: $X = g^{-x}$
- Sign$_x(m)$ by signer:
  - $R = g^r$
  - $c =_q H(R||m)$
  - $s =_q r + x \cdot c$
  - output $\sigma = (s, c)$
- Verify$_X(\sigma, m)$:
  - calculate $R = g^s X^c$
  - check $H(R||m) =^? c$

A multi-signature scheme [BN06]

- Space: same G, g
- KeyGen (by parties $i = 1, ..., n$):
  - Secret SignKey: $x_i \in Z_q$
  - Public VerKey: $X_i = g^{x_i}$
- Sign$_{x,L}(m)$ by subset $I \subseteq \{1, ..., n\}$
  - $R = \prod_{i \in I} R_i = \prod_{i \in I} g^{r_i}$
  - $c_i =_q H(X_i||R||I||m)$
  - $s =_q \sum_{i \in L} s_i = \sum_{i \in I}(r_i + x_i c_i)$
  - output $\sigma = (R, s)$
- Verify(σ, m):
  - calculate $c_i = H(X_i||R||M||I||m)$
  - check $g^s =^? R \prod_{i \in I} X_i^{c_i}$

A DL-based example: threshold Schnorr signature

(DL = Discrete-Logarithm)

(Next: ignore details — just making comparative remarks)

Non-threshold scheme [Sch90]

- Space: G, g (group, generator)
- KeyGen (by signer):
  - Secret SignKey: $x \in Z_q$
  - Public VerKey: $X = g^{-x}$
- Sign$_x(m)$ by signer:
  - $R = g^r$
  - $c =_q H(R||m)$
  - $s =_q r + x \cdot c$
  - output $\sigma = (s, c)$
- Verify$_X(\sigma, m)$:
  - calculate $R = g^s X^c$
  - check $H(R||m) =^? c$

A multi-signature scheme [BN06]

- Space: same G, g
- KeyGen (by parties $i = 1, ..., n$):
  - Secret SignKey: $x_i \in Z_q$
  - Public VerKey: $X_i = g^{x_i}$
- Sign$_{x,L}(m)$ by subset $I \subseteq \{1, ..., n\}$
  - $R = \prod_{i \in I} R_i = \prod_{i \in I} g^{r_i}$
  - $c_i =_q H(X_i||R||I||m)$
  - $s =_q \sum_{i \in L} s_i = \sum_{i \in I} (r_i + x_i c_i)$
  - output $\sigma = (R, s)$
- Verify(σ, m):
  - calculate $c_i = H(X_i||R||M||I||m)$
  - check $g^s =^? R \prod_{i \in I} X_i^{c_i}$

A DL-based example: threshold Schnorr signature

(DL = Discrete-Logarithm)

(Next: ignore details — just making comparative remarks)

Non-threshold scheme [Sch90]

- Space: G, g (group, generator)
- KeyGen (by signer):
  - Secret SignKey: $x \in Z_q$
  - Public VerKey: $X = g^{-x}$
- Sign$_x(m)$ by signer:
  - $R = g^r$
  - $c =_q H(R||m)$
  - $s =_q r + x \cdot c$
  - output $\sigma = (s, c)$
- Verify$_X(\sigma, m)$:
  - calculate $R = g^s X^c$
  - check $H(R||m) =^? c$

A multi-signature scheme [BN06]

- Space: same G, g
- KeyGen (by parties $i = 1, ..., n$):
  - Secret SignKey: $x_i \in Z_q$
  - Public VerKey: $X_i = g^{x_i}$
- Sign$_{x,L}(m)$ by subset $I \subseteq \{1, ..., n\}$
  - $R = \prod_{i \in I} R_i = \prod_{i \in I} g^{r_i}$
  - $c_i =_q H(X_i||R||I||m)$
  - $s =_q \sum_{i \in L} s_i = \sum_{i \in I}(r_i + x_i c_i)$
  - output $\sigma = (R, s)$
- Verify(σ, m):
  - calculate $c_i = H(X_i||R||M||I||m)$
  - check $g^s =^? R \prod_{i \in I} X_i^{c_i}$

A DL-based example: threshold Schnorr signature

(DL = Discrete-Logarithm)

(Next: ignore details — just making comparative remarks)

Non-threshold scheme [Sch90]

- Space: G, g (group, generator)
- KeyGen (by signer):
  - Secret SignKey: $x \in Z_q$
  - Public VerKey: $X = g^{-x}$
- Sign$_x(m)$ by signer:
  - $R = g^r$
  - $c =_q H(R||m)$
  - $s =_q r + x \cdot c$
  - output $\sigma = (s, c)$
- Verify$_X(\sigma, m)$:
  - calculate $R = g^s X^c$
  - check $H(R||m) =^? c$

A multi-signature scheme [BN06]

- Space: same G, g
- KeyGen (by parties $i = 1, ..., n$):
  - Secret SignKey: $x_i \in Z_q$
  - Public VerKey: $X_i = g^{x_i}$
- Sign$_{x,L}(m)$ by subset $I \subseteq \{1, ..., n\}$
  - $R = \prod_{i \in I} R_i = \prod_{i \in I} g^{r_i}$
  - $c_i =_q H(X_i||R||I||m)$
  - $s =_q \sum_{i \in L} s_i = \sum_{i \in I}(r_i + x_i c_i)$
  - output $\sigma = (R, s)$
- Verify(σ, m):
  - calculate $c_i = H(X_i||R||M||I||m)$
  - check $g^s =^? R \prod_{i \in I} X_i^{c_i}$

A DL-based example: threshold Schnorr signature

(DL = Discrete-Logarithm)

(Next: ignore details — just making comparative remarks)

Non-threshold scheme [Sch90]

- Space: G, g (group, generator)
- KeyGen (by signer):
  - Secret SignKey: $x \in Z_q$
  - Public VerKey: $X = g^{-x}$
- Sign$_x(m)$ by signer:
  - $R = g^r$
  - $c =_q H(R||m)$
  - $s =_q r + x \cdot c$
  - output $\sigma = (s, c)$
- Verify$_X(\sigma, m)$:
  - calculate $R = g^s X^c$
  - check $H(R||m) =^? c$

A multi-signature scheme [BN06]

- Space: same G, g
- KeyGen (by parties $i = 1, ..., n$):
  - Secret SignKey: $x_i \in Z_q$
  - Public VerKey: $X_i = g^{x_i}$
- Sign$_{x,L}(m)$ by subset $I \subseteq \{1, ..., n\}$
  - $R = \prod_{i \in I} R_i = \prod_{i \in I} g^{r_i}$
  - $c_i =_q H(X_i||R||I||m)$
  - $s =_q \sum_{i \in L} s_i = \sum_{i \in I}(r_i + x_i c_i)$
  - output $\sigma = (R, s)$
- Verify(σ, m):
  - calculate $c_i = H(X_i||R||M||I||m)$
  - check $g^s =^? R \prod_{i \in I} X_i^{c_i}$

A DL-based example: threshold Schnorr signature

(DL = Discrete-Logarithm)

(Next: ignore details — just making comparative remarks)

Non-threshold scheme [Sch90]

- Space: G, g (group, generator)
- KeyGen (by signer):
  - Secret SignKey: $x \in Z_q$
  - Public VerKey: $X = g^{-x}$
- Sign$_x(m)$ by signer:
  - $R = g^r$
  - $c =_q H(R||m)$
  - $s =_q r + x \cdot c$
  - output $\sigma = (s, c)$
- Verify$_X(\sigma, m)$:
  - calculate $R = g^s X^c$
  - check $H(R||m) =^? c$

A multi-signature scheme [BN06]

- Space: same G, g
- KeyGen (by parties $i = 1, ..., n$):
  - Secret SignKey: $x_i \in Z_q$
  - Public VerKey: $X_i = g^{x_i}$
- Sign$_{x,L}(m)$ by subset $I \subseteq \{1, ..., n\}$
  - $R = \prod_{i \in I} R_i = \prod_{i \in I} g^{r_i}$
  - $c_i =_q H(X_i||R||I||m)$
  - $s =_q \sum_{i \in L} s_i = \sum_{i \in I}(r_i + x_i c_i)$
  - output $\sigma = (R, s)$
- Verify(σ, m):
  - calculate $c_i = H(X_i||R||M||I||m)$
  - check $g^s =^? R \prod_{i \in I} X_i^{c_i}$

A DL-based example: threshold Schnorr signature

(DL = Discrete-Logarithm)

(Next: ignore details — just making comparative remarks)

Non-threshold scheme [Sch90]

- Space: G, g (group, generator)
- KeyGen (by signer):
  - Secret SignKey: $x \in Z_q$
  - Public VerKey: $X = g^{-x}$
- $\text{Sign}_x(m)$ by signer:
  - $R = g^r$
  - $c =_q H(R||m)$
  - $s =_q r + x \cdot c$
  - output $\sigma = (s, c)$
- $\text{Verify}_X(\sigma, m)$:
  - calculate $R = g^s X^c$
  - check $H(R||m) =^? c$

A multi-signature scheme [BN06]

- Space: same G, g
- KeyGen (by parties $i = 1, \ldots, n$):
  - Secret SignKey: $x_i \in Z_q$
  - Public VerKey: $X_i = g^{x_i}$
- $\text{Sign}_{x,L}(m)$ by subset $I \subseteq \{1, \ldots, n\}$
  - $R = \prod_{i \in I} R_i = \prod_{i \in I} g^{r_i}$
  - $c_i =_q H(X_i||R||I||m)$
  - $s =_q \sum_{i \in L} s_i = \sum_{i \in I}(r_i + x_i c_i)$
  - output $\sigma = (R, s)$
- $\text{Verify}(\sigma, m)$:
  - calculate $c_i = H(X_i||R||M||I||m)$
  - check $g^s =^? R \prod_{i \in I} X_i^{c_i}$

A DL-based example: threshold Schnorr signature

(DL = Discrete-Logarithm)

(Next: ignore details — just making comparative remarks)

Non-threshold scheme [Sch90]

- Space: G, g (group, generator)
- KeyGen (by signer):
  - Secret SignKey: $x \in Z_q$
  - Public VerKey: $X = g^{-x}$
- Sign$_x(m)$ by signer:
  - $R = g^r$
  - $c =_q H(R||m)$
  - $s =_q r + x \cdot c$
  - output $\sigma = (s, c)$
- Verify$_X(\sigma, m)$:
  - calculate $R = g^s X^c$
  - check $H(R||m) =^? c$

A multi-signature scheme [BN06]

- Space: same G, g
- KeyGen (by parties $i = 1, \ldots, n$):
  - Secret SignKey: $x_i \in Z_q$
  - Public VerKey: $X_i = g^{x_i}$
- Sign$_{x,L}(m)$ by subset $I \subseteq \{1, \ldots, n\}$
  - $R = \prod_{i \in I} R_i = \prod_{i \in I} g^{r_i}$
  - $c_i =_q H(X_i||R||I||m)$
  - $s =_q \sum_{i \in L} s_i = \sum_{i \in I}(r_i + x_i c_i)$
  - output $\sigma = (R, s)$
- Verify(σ, m):
  - calculate $c_i = H(X_i||R||M||I||m)$
  - check $g^s =^? R \prod_{i \in I} X_i^{c_i}$

A DL-based example: threshold Schnorr signature

(DL = Discrete-Logarithm)

(Next: ignore details — just making comparative remarks)

Non-threshold scheme [Sch90]

- Space: G, g (group, generator)
- KeyGen (by signer):
  - Secret SignKey: $x \in Z_q$
  - Public VerKey: $X = g^{-x}$
- Sign$_x(m)$ by signer:
  - $R = g^r$
  - $c =_q H(R||m)$
  - $s =_q r + x \cdot c$
  - output $\sigma = (s, c)$
- Verify$_X(\sigma, m)$:
  - calculate $R = g^s X^c$
  - check $H(R||m) =^? c$

A multi-signature scheme [BN06]

- Space: same G, g
- KeyGen (by parties $i = 1, \ldots, n$):
  - Secret SignKey: $x_i \in Z_q$
  - Public VerKey: $X_i = g^{x_i}$
- Sign$_{x,L}(m)$ by subset $I \subseteq \{1, \ldots, n\}$
  - $R = \prod_{i \in I} R_i = \prod_{i \in I} g^{r_i}$
  - $c_i =_q H(X_i||R||I||m)$
  - $s =_q \sum_{i \in L} s_i = \sum_{i \in I}(r_i + x_i c_i)$
  - output $\sigma = (R, s)$
- Verify(σ, m):
  - calculate $c_i = H(X_i||R||M||I||m)$
  - check $g^s =^? R \prod_{i \in I} X_i^{c_i}$

A DL-based example: threshold Schnorr signature

(DL = Discrete-Logarithm)

(Next: ignore details — just making comparative remarks)

Non-threshold scheme [Sch90]

- Space: G, g (group, generator)
- KeyGen (by signer):
  - Secret SignKey: $x \in Z_q$
  - Public VerKey: $X = g^{-x}$
- $\text{Sign}_x(m)$ by signer:
  - $R = g^r$
  - $c =_q H(R||m)$
  - $s =_q r + x \cdot c$
  - output $\sigma = (s, c)$
- $\text{Verify}_X(\sigma, m)$:
  - calculate $R = g^s X^c$
  - check $H(R||m) =^? c$

A multi-signature scheme [BN06]

- Space: same G, g
- KeyGen (by parties $i = 1, ..., n$):
  - Secret SignKey: $x_i \in Z_q$
  - Public VerKey: $X_i = g^{x_i}$
- $\text{Sign}_{x,L}(m)$ by subset $I \subseteq \{1, ..., n\}$
  - $R = \prod_{i \in I} R_i = \prod_{i \in I} g^{r_i}$
  - $c_i =_q H(X_i||R||I||m)$
  - $s =_q \sum_{i \in L} s_i = \sum_{i \in I}(r_i + x_i c_i)$
  - output $\sigma = (R, s)$
- $\text{Verify}(\sigma, m)$:
  - calculate $c_i = H(X_i||R||M||I||m)$
  - check $g^s =^? R \prod_{i \in I} X_i^{c_i}$

A DL-based example: threshold Schnorr signature

(DL = Discrete-Logarithm)

(Next: ignore details — just making comparative remarks)

Non-threshold scheme [Sch90]

- Space: G, g (group, generator)
- KeyGen (by signer):
  - Secret SignKey: $x \in Z_q$
  - Public VerKey: $X = g^{-x}$
- Sign$_x(m)$ by signer:
  - $R = g^r$
  - $c =_q H(R||m)$
  - $s =_q r + x \cdot c$
  - output $\sigma = (s, c)$
- Verify$_X(\sigma, m)$:
  - calculate $R = g^s X^c$
  - check $H(R||m) =^? c$

A multi-signature scheme [BN06]*

- Space: same G, g
- KeyGen (by parties $i = 1, ..., n$):
  - Secret SignKey: $x_i \in Z_q$
  - Public VerKey: $X_i = g^{x_i}$
- Sign$_{x,L}(m)$ by subset $I \subseteq \{1, ..., n\}$
  - $R = \prod_{i\in I} R_i = \prod_{i \in I} g^{r_i}$
  - $c_i =_q H(X_i||R||I||m)$
  - $s =_q \sum_{i \in L} s_i = \sum_{i\in I}(r_i + x_i c_i)$
  - output $\sigma = (R, s)$
- Verify(σ, m):
  - calculate $c_i = H(X_i||R||M||I||m)$
  - check $g^s =^? R \prod_{i \in I} X_i^{c_i}$

*Some features:

A DL-based example: threshold Schnorr signature

(DL = Discrete-Logarithm)

(Next: ignore details — just making comparative remarks)

Non-threshold scheme [Sch90]

- Space: G, g (group, generator)
- KeyGen (by signer):
  - Secret SignKey: $x \in Z_q$
  - Public VerKey: $X = g^{-x}$
- Sign$_x(m)$ by signer:
  - $R = g^r$
  - $c =_q H(R||m)$
  - $s =_q r + x \cdot c$
  - output $\sigma = (s, c)$
- Verify$_X(\sigma, m)$:
  - calculate $R = g^s X^c$
  - check $H(R||m) =^? c$

A multi-signature scheme [BN06]*

- Space: same G, g
- KeyGen (by parties $i = 1, ..., n$):
  - Secret SignKey: $x_i \in Z_q$
  - Public VerKey: $X_i = g^{x_i}$
- Sign$_{x,L}(m)$ by subset $I \subseteq \{1, ..., n\}$
  - $R = \prod_{i \in I} R_i = \prod_{i \in I} g^{r_i}$
  - $c_i =_q H(X_i||R||I||m)$
  - $s =_q \sum_{i \in L} s_i = \sum_{i \in I}(r_i + x_i c_i)$
  - output $\sigma = (R, s)$
- Verify(σ, m):
  - calculate $c_i = H(X_i||R||M||I||m)$
  - check $g^s =^? R \prod_{i \in I} X_i^{c_i}$

***Some features:** no dealer;

A DL-based example: threshold Schnorr signature

(DL = Discrete-Logarithm)

(Next: ignore details — just making comparative remarks)

Non-threshold scheme [Sch90]

- Space: G, g (group, generator)
- KeyGen (by signer):
  - Secret SignKey: $x \in Z_q$
  - Public VerKey: $X = g^{-x}$
- Sign$_x(m)$ by signer:
  - $R = g^r$
  - $c =_q H(R||m)$
  - $s =_q r + x \cdot c$
  - output $\sigma = (s, c)$
- Verify$_X(\sigma, m)$:
  - calculate $R = g^s X^c$
  - check $H(R||m) =^? c$

A multi-signature scheme [BN06]*

- Space: same G, g
- KeyGen (by parties $i = 1, ..., n$):
  - Secret SignKey: $x_i \in Z_q$
  - Public VerKey: $X_i = g^{x_i}$
- Sign$_{x,L}(m)$ by subset $I \subseteq \{1, ..., n\}$
  - $R = \prod_{i \in I} R_i = \prod_{i \in I} g^{r_i}$
  - $c_i =_q H(X_i||R||I||m)$
  - $s =_q \sum_{i \in L} s_i = \sum_{i \in I}(r_i + x_i c_i)$
  - output $\sigma = (R, s)$
- Verify(σ, m):
  - calculate $c_i = H(X_i||R||M||I||m)$
  - check $g^s =^? R \prod_{i \in I} X_i^{c_i}$

***Some features:** no dealer; dynamic threshold (verifier decides what is acceptable);

A DL-based example: threshold Schnorr signature

(DL = Discrete-Logarithm)

(Next: ignore details — just making comparative remarks)

Non-threshold scheme [Sch90]

- Space: G, g (group, generator)
- KeyGen (by signer):
  - Secret SignKey: $x \in Z_q$
  - Public VerKey: $X = g^{-x}$
- Sign$_x(m)$ by signer:
  - $R = g^r$
  - $c =_q H(R||m)$
  - $s =_q r + x \cdot c$
  - output $\sigma = (s, c)$
- Verify$_X(\sigma, m)$:
  - calculate $R = g^s X^c$
  - check $H(R||m) =^? c$

A multi-signature scheme [BN06]*

- Space: same G, g
- KeyGen (by parties $i = 1, ..., n$):
  - Secret SignKey: $x_i \in Z_q$
  - Public VerKey: $X_i = g^{x_i}$
- Sign$_{x,L}(m)$ by subset $I \subseteq \{1, ..., n\}$
  - $R = \prod_{i \in I} R_i = \prod_{i \in I} g^{r_i}$
  - $c_i =_q H(X_i||R||I||m)$
  - $s =_q \sum_{i \in L} s_i = \sum_{i \in I}(r_i + x_i c_i)$
  - output $\sigma = (R, s)$
- Verify(σ, m):
  - calculate $c_i = H(X_i||R||M||I||m)$
  - check $g^s =^? R \prod_{i \in I} X_i^{c_i}$

*__Some features:__ no dealer; dynamic threshold (verifier decides what is acceptable); dynamic set of signers;

A DL-based example: threshold Schnorr signature

(DL = Discrete-Logarithm)

(Next: ignore details — just making comparative remarks)

Non-threshold scheme [Sch90]

- Space: G, g (group, generator)
- KeyGen (by signer):
  - Secret SignKey: $x \in Z_q$
  - Public VerKey: $X = g^{-x}$
- Sign$_x(m)$ by signer:
  - $R = g^r$
  - $c =_q H(R||m)$
  - $s =_q r + x \cdot c$
  - output $\sigma = (s, c)$
- Verify$_X(\sigma, m)$:
  - calculate $R = g^s X^c$
  - check $H(R||m) =^? c$

A multi-signature scheme [BN06]*

- Space: same G, g
- KeyGen (by parties $i = 1, ..., n$):
  - Secret SignKey: $x_i \in Z_q$
  - Public VerKey: $X_i = g^{x_i}$
- Sign$_{x,L}(m)$ by subset $I \subseteq \{1, ..., n\}$
  - $R = \prod_{i \in I} R_i = \prod_{i \in I} g^{r_i}$
  - $c_i =_q H(X_i||R||I||m)$
  - $s =_q \sum_{i \in L} s_i = \sum_{i \in I}(r_i + x_i c_i)$
  - output $\sigma = (R, s)$
- Verify(σ, m):
  - calculate $c_i = H(X_i||R||M||I||m)$
  - check $g^s =^? R \prod_{i \in I} X_i^{c_i}$

***Some features:** no dealer; dynamic threshold (verifier decides what is acceptable); dynamic set of signers; verifying $\Rightarrow$ knowing who signed.

A DL-based example: threshold Schnorr signature

(DL = Discrete-Logarithm)

(Next: ignore details — just making comparative remarks)

Non-threshold scheme [Sch90]

- Space: G, g (group, generator)
- KeyGen (by signer):
  - Secret SignKey: $x \in Z_q$
  - Public VerKey: $X = g^{-x}$
- Sign$_x(m)$ by signer:
  - $R = g^r$
  - $c =_q H(R||m)$
  - $s =_q r + x \cdot c$
  - output $\sigma = (s, c)$
- Verify$_X(\sigma, m)$:
  - calculate $R = g^s X^c$
  - check $H(R||m) =^? c$

A multi-signature scheme [BN06]*

- Space: same G, g
- KeyGen (by parties $i = 1, ..., n$):
  - Secret SignKey: $x_i \in Z_q$
  - Public VerKey: $X_i = g^{x_i}$
- Sign$_{x,L}(m)$ by subset $I \subseteq \{1, ..., n\}$
  - $R = \prod_{i \in I} R_i = \prod_{i \in I} g^{r_i}$
  - $c_i =_q H(X_i||R||I||m)$
  - $s =_q \sum_{i \in L} s_i = \sum_{i \in I}(r_i + x_i c_i)$
  - output $\sigma = (R, s)$
- Verify(σ, m):
  - calculate $c_i = H(X_i||R||M||I||m)$
  - check $g^s =^? R \prod_{i \in I} X_i^{c_i}$

***Some features:** no dealer; dynamic threshold (verifier decides what is acceptable); dynamic set of signers; verifying ⇒ knowing who signed.