Sizing up the threshold

Challenges and opportunities in the standardization of threshold schemes for cryptographic primitives

Apostol Vassilev and Luís Brandão, joint work w/ Nicky Mouha

National Institute of Standards and Technology (Gaithersburg, United States)

2nd Theory of Implementation Security Workshop
January 09, 2018 (Zurich, Switzerland)

The cybersecurity challenge today

Key finding:

the overall cybersecurity picture remains grim;

Key recommendations:

- encrypt sensitive data
- patch promptly

Image source: http://www.verizonenterprise.com/verizon-insights-lab/dbir/2017/

Observation

In modern cryptography the algorithms are known

- ➢ **security relies on keys**
  - must be unpredictable and inaccessible to attackers
- ➢ **whole keys are stored in some place(s)**
  - on a single computer
- ➢ **black-box assumption**
  - theory and practice
  - two different stories

Image source: https://xkcd.com/257

Real-world examples of black-box failures

Image source: heartbleed.com

Heartbleed bug (2014)
Server private key revealed

Meltdown & Spectre (2017)
All memory (including keys) revealed

https://www.windowscentral.com/all-modern-processors-impacted-new-meltdown-and-spectre-exploits

Image source: https://regmedia.co.uk/2015/09/24/segula_bulb_648.jpg

"ZigBee Chain reaction" (2017)
Phillips Hue light-bulbs secret key revealed

Bellcore attack (1997) on RSA-CRT
An injected fault corrupts part of computation, enabling factorization of the modulus and private key compromise.

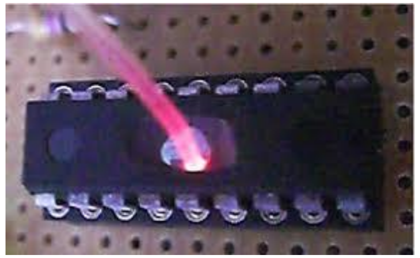


Image source: Schmidt, Hutter: Optical and EM Fault-Attacks on CRT-based RSA: Concrete Results

It is essential to have reliable implementations of cryptographic primitives, e.g., encrypt, sign, generate randomness, immune to breaches in the computational environment

Can we standardize threshold schemes to promote their use in real life as a way to improve security

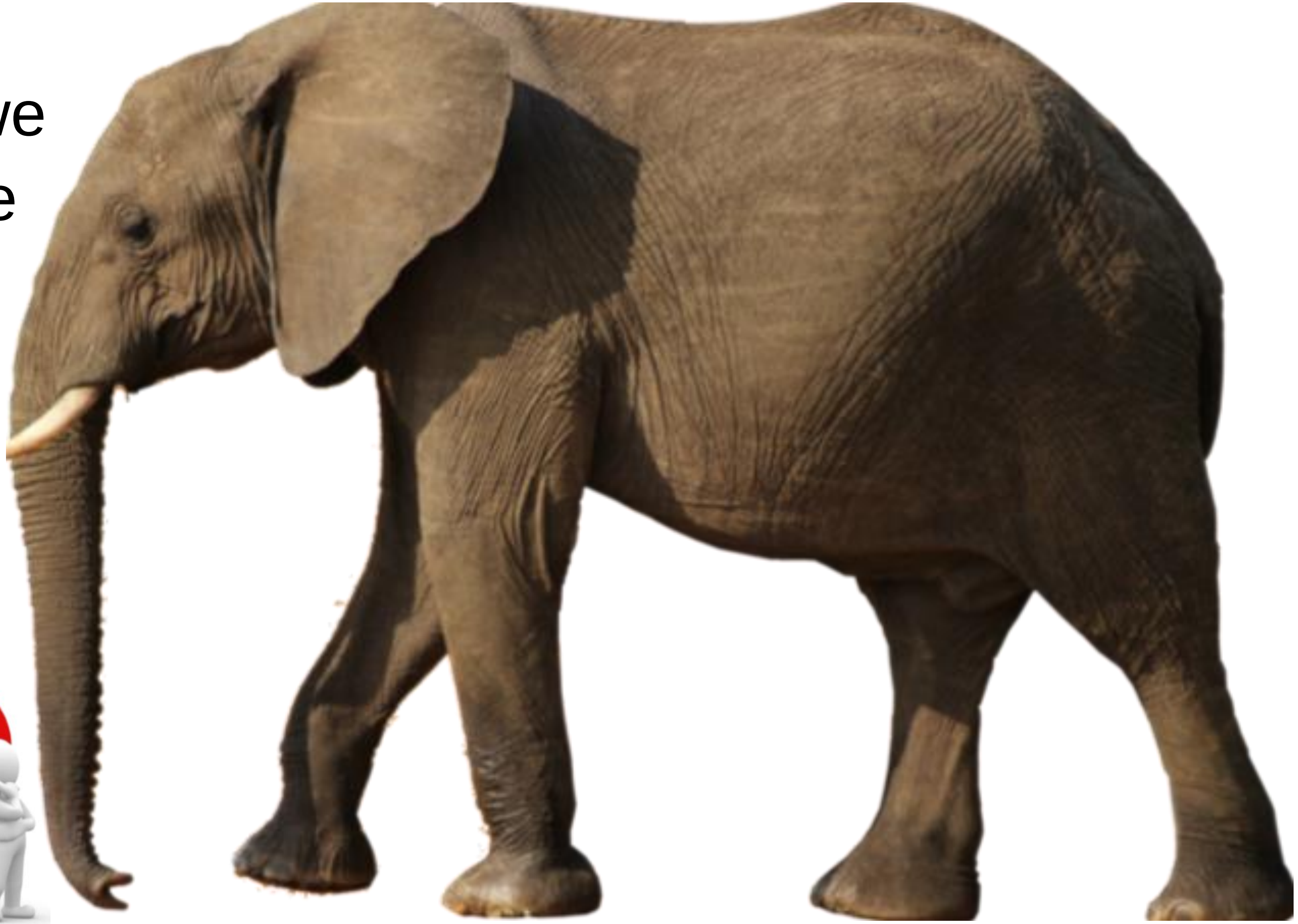


Image source: https://pngimg.com/download/18792

NIST cryptographic standards: why do they matter?

NIST develops standards for crypto primitives (a.k.a. approved primitives).

- Digital signature
- Encryption
- Hash
- PRGen
- Key establishment
- Key derivation

By law (FISMA 2002, 2014), crypto primitives used in federal systems must be NIST-approved and their implementation must be FIPS 140-2 validated

- Validation means the security assertions specified by the standard for a specific primitive implementation must be tested and verified to hold

Industries and countries have also voluntarily adopted FIPS 140-2 validations

- financial
- Canada

Current "FIPS 140-2" validation process / CMVP

Vendor

Designs and Produces

Hardware · Software · Firmware

Define Boundary

Define Approved Mode of operation

Security Policy

CST Lab

Tests for Conformance

Derived Test Requirements

CAVP Algorithm Testing

Documentation Review

Source Code Review

Operations and Physical Testing

CMVP

NIST and CSEC

Validates

Review Test Results

Ongoing NVLAP Assessment

Issue Certificates

NIST Cost Recovery Fee

User

Specifies and Purchases

Security and Assurance

Applications or products with embedded modules

Human-centric approach to testing and validation

Legend:
- CAVP = Cryptographic Algorithm Validation Program
- CMVP = Cryptographic Module Validation Program
- CSEC = Communications Security Establishment (Canada)
- CST = Cryptographic and Security Testing
- FIPS = Federal Information Processing Standards
- NIST = National Institute of Standards and Technology
- NVLAP = National Voluntary Laboratory Accreditation Program

What can go wrong?

Long Validation Cycles

Well beyond product development cycles

Hinder adoption of new technology by the Federal Government

Shallow Depth of Testing

Software and hardware testing methodology inadequate for today's complexity of crypto implementations

Costly and Rigid

Difficult to obtain compliance assurance on platforms of actual use

Limits the industry's efforts to validate more products

Prevents the industry from fixing critical problems, e.g. CVE, without breaking program rules, i.e. hinders rapid patching by relying organizations

Impossible to fix within the existing box

Some improvements help but fall short of solving the problems

Image source: https://pixabay.com/en/thinking-out-of-the-box-2958103

Automate as much as possible

Image source: https://pixabay.com/en/mechanics-hand-finger-touch-2170638/

- Reduce the validation cycle length
- Increase the depth of testing
- Enable Just-In-Time validations
- Reduce the cost of validations

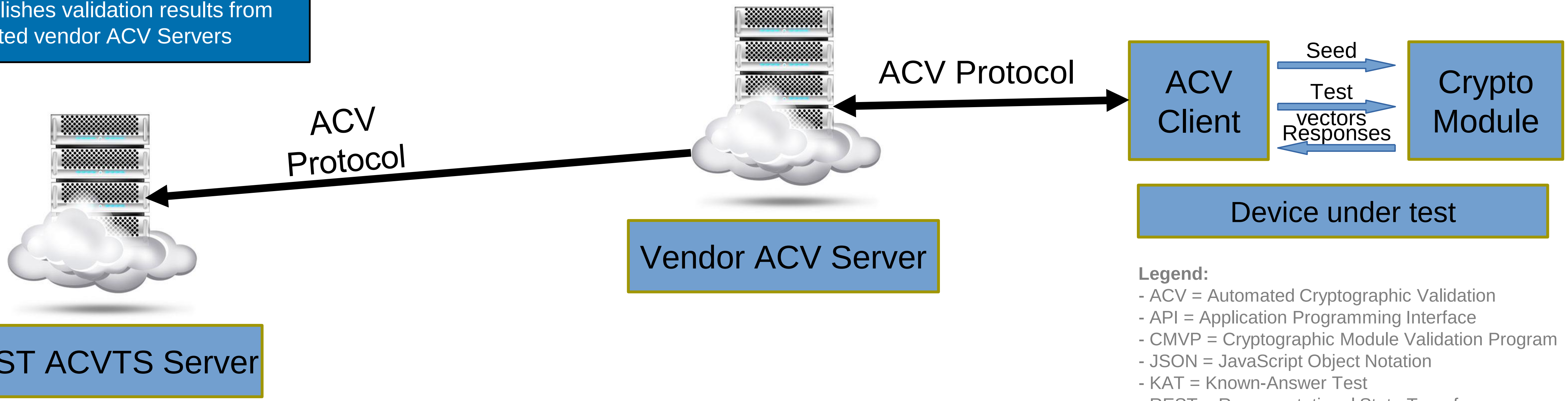
Powerful economic incentives for the industry

Future CMVP Validation Structure

Validation Authority Server:
- Web hosted service w/ REST API
- Registers ACV Servers
- Generates JSON KAT vectors
- Validates JSON KAT results
- Publishes validation results from trusted vendor ACV Servers

ACV Proxy/Server:
- Web hosted service
- Interacts with NIST ACV Server to obtain JSON KAT data
- Optionally generates JSON test vectors
- Optionally performs results verification
- Reports JSON KAT results to NIST ACV Server

ACV Client:
- Integrated into Device under test
- May convert JSON test vectors to format acceptable by crypto module under test
- Returns KAT answers to ACV server in JSON format



Legend:
- ACV = Automated Cryptographic Validation
- API = Application Programming Interface
- CMVP = Cryptographic Module Validation Program
- JSON = JavaScript Object Notation
- KAT = Known-Answer Test
- REST = Representational State Transfer
- ACVTS = Automated Crypto Validation Testing Service

Computer-based testing and validation

Where are we today?

Making progress towards the desired goals

- ACVP actively developed - https://github.com/usnistgov/ACVP
  - NIST team (feds and contractors) in place and funded, collaborating w/ Cisco
  - Open to others to join in
  - Targeting replicating complete CAVP testing in Q3, 2018

- Pilot CMVP validations started
  - One open source module (Red Hat, NSS lib), one proprietary (Apple)
  - Targeting rolling out CMVP auto validation in Q2, 2019.
  - Actual date depends on findings in pilot validations

- Vendor Criteria for participation has been developed
  - Coordinated with NVLAP at NIST
  - Targeting criteria rollout in Q2, 2018

- Public update planned for ICMC 2018 , May 8-11 2018, Ottawa, Canada

See also the high-level public project plan at http://csrc.nist.gov/projects/acvt/ for further details

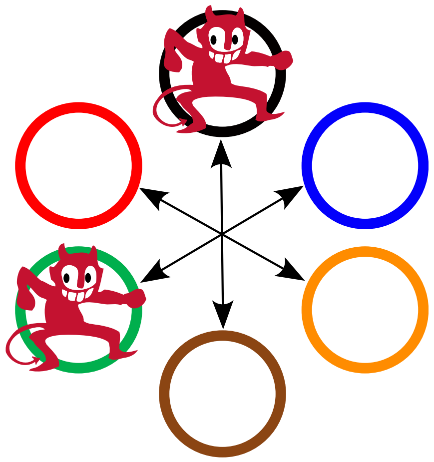
Outline

1. Introduction: we need reliable crypto
2. Validating a crypto module (the CMVP at NIST)
3. The threshold approach
4. Characterizing a threshold scheme
5. The threshold validation challenge
6. Concluding remarks

Threshold approach (high level idea)

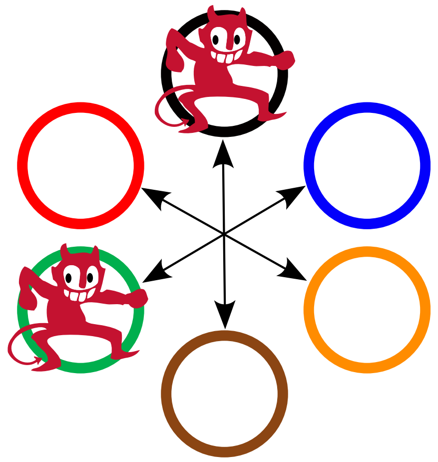
Threshold approach (high level idea)

Use redundancy & diversity to mitigate the compromise of some (up to a *threshold* number of) components (a.k.a. nodes)



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

Threshold approach (high level idea)

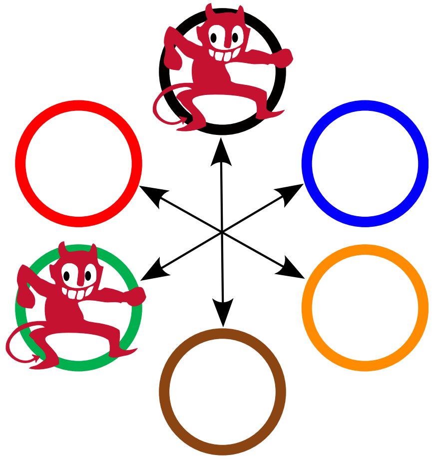


The red dancing devil is from clker.com/clipart-13643.html

Use redundancy & diversity to mitigate the compromise of some (up to a *threshold* number of) components (a.k.a. nodes)

The intuitive aim: improve security vs. a non-threshold scheme

Threshold approach (high level idea)



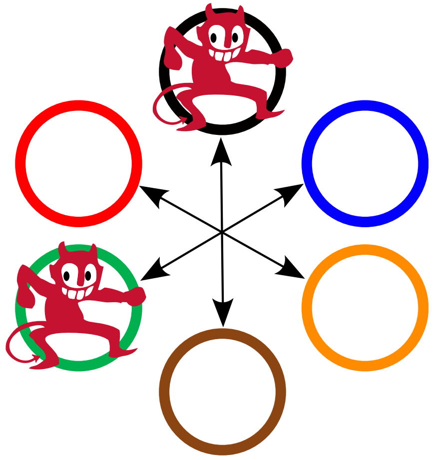
The red dancing devil is from
clker.com/clipart-13643.html

Use redundancy & diversity to mitigate the compromise of some (up to a *threshold* number of) components (a.k.a. nodes)

The intuitive aim: improve security
vs. a non-threshold scheme
(depends on adversarial model)

clker.com/clipart-10778.html

Threshold approach (high level idea)



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

Use redundancy & diversity to mitigate the compromise of some (up to a *threshold* number of) components (a.k.a. nodes)

The intuitive aim: improve security
vs. a non-threshold scheme
(depends on adversarial model)

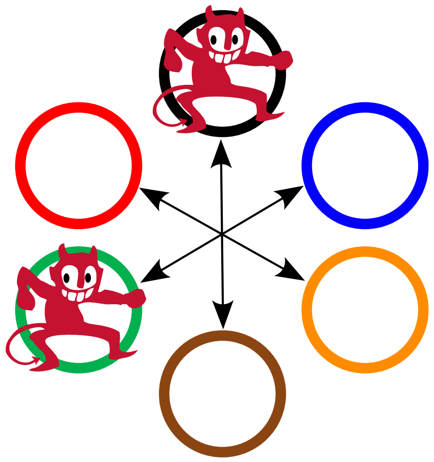
clker.com/clipart-10778.html

Image adapted from:
openclipart.org/detail/283392

Our current step: devise initial questions for discussion towards standardization and validation of threshold-cryptography* related schemes.

Threshold approach (high level idea)

Use redundancy & diversity to mitigate the compromise of some (up to a *threshold* number of) components (a.k.a. nodes)



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

The intuitive aim: improve security vs. a non-threshold scheme (depends on adversarial model)

clker.com/clipart-10778.html

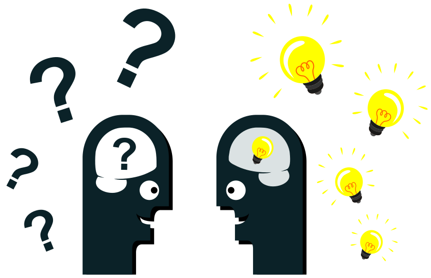


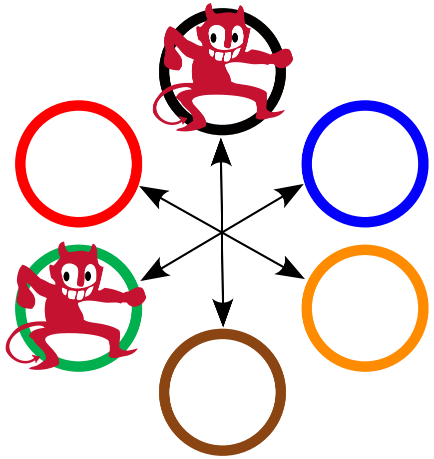
Image adapted from: openclipart.org/detail/283392

Our current step: devise initial questions for discussion towards standardization and validation of threshold-cryptography* related schemes.

* We may use **“threshold cryptography”** as a shorthand for threshold approaches applied to crypto primitives, and **“threshold <primitive> scheme”** for specific constructions.

Threshold approach (high level idea)

Use redundancy & diversity to mitigate the compromise of some (up to a *threshold* number of) components (a.k.a. nodes)



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

The intuitive aim: improve security vs. a non-threshold scheme (depends on adversarial model)

clker.com/clipart-10778.html

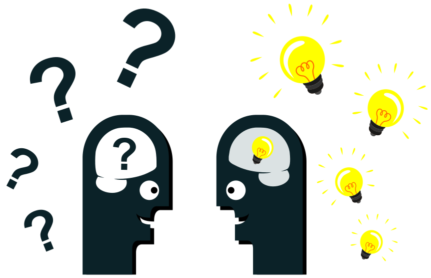


Image adapted from: openclipart.org/detail/283392

Our current step: devise initial questions for discussion towards standardization and validation of threshold-cryptography* related schemes.

Several related research areas: threshold cryptography; secure multi-party computation; intrusion-tolerant protocols; fault-tolerant and side-channel-resistant circuits; leakage models; secret-sharing schemes (possible arbitrary access structures); …

* We may use **"threshold cryptography"** as a shorthand for threshold approaches applied to crypto primitives, and **"threshold <primitive> scheme"** for specific constructions.

Illustrative example(s)

Illustrative example(s)

<u>3-out-of-3 encryption</u>

3-out-of-3 nodes needed to produce a ciphertext; key is secret if at least 1 component does not leak.

Illustrative example(s)

3-out-of-3 encryption

3-out-of-3 nodes needed to produce a ciphertext; key is secret if at least 1 component does not leak.

Does a threshold scheme provide better security than a non-threshold one?

Illustrative example(s)

<u>3-out-of-3 encryption</u>

3-out-of-3 nodes needed to produce a ciphertext; key is secret if at least 1 component does not leak.

Does a threshold scheme provide better security than a non-threshold one?

- Are there common failure modes (e.g., is breaking 1 equivalent to breaking 3)?
- Fault tolerance: two cannot break secrecy, but can one alone break integrity!?
- Is plaintext secrecy affected? (how does the client send it: whole or shared?)
- May the implementation bring new security problems?

Illustrative example(s)

<u>3-out-of-3 encryption</u>

3-out-of-3 nodes needed to produce a ciphertext; key is secret if at least 1 component does not leak.

<u>2-out-of-3 signature</u>

2 nodes needed to produce a signature; correct and key secret if at least 2 nodes are correct and do not leak.

Does a threshold scheme provide better security than a non-threshold one?

- Are there common failure modes (e.g., is breaking 1 equivalent to breaking 3)?
- Fault tolerance: two cannot break secrecy, but can one alone break integrity!?
- Is plaintext secrecy affected? (how does the client send it: whole or shared?)
- May the implementation bring new security problems?
- Even if independent failure mode: can breaking 2 out of 3 be easier than 1 out of 1?

Illustrative example(s)

3-out-of-3 encryption

3-out-of-3 nodes needed to produce a ciphertext; key is secret if at least 1 component does not leak.

2-out-of-3 signature

2 nodes needed to produce a signature; correct and key secret if at least 2 nodes are correct and do not leak.

Does a threshold scheme provide better security than a non-threshold one?

- Are there common failure modes (e.g., is breaking 1 equivalent to breaking 3)?
- Fault tolerance: two cannot break secrecy, but can one alone break integrity!?
- Is plaintext secrecy affected? (how does the client send it: whole or shared?)
- May the implementation bring new security problems?
- Even if independent failure mode: can breaking 2 out of 3 be easier than 1 out of 1?

“*k*-out-of-*n*” is not a sufficient characterization
to enable a comprehensive security assertion.

Image sources:
clker.com/clipart-*.html,
for * in {encryption, 3712}

One metric of security: reliability

One metric of security: reliability

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

One metric of security: reliability

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

A possible model: nodes fail independently, with constant rate probability.

Example: ETTF per node under attack is 1 month

(ETTF = Expected time to failure)

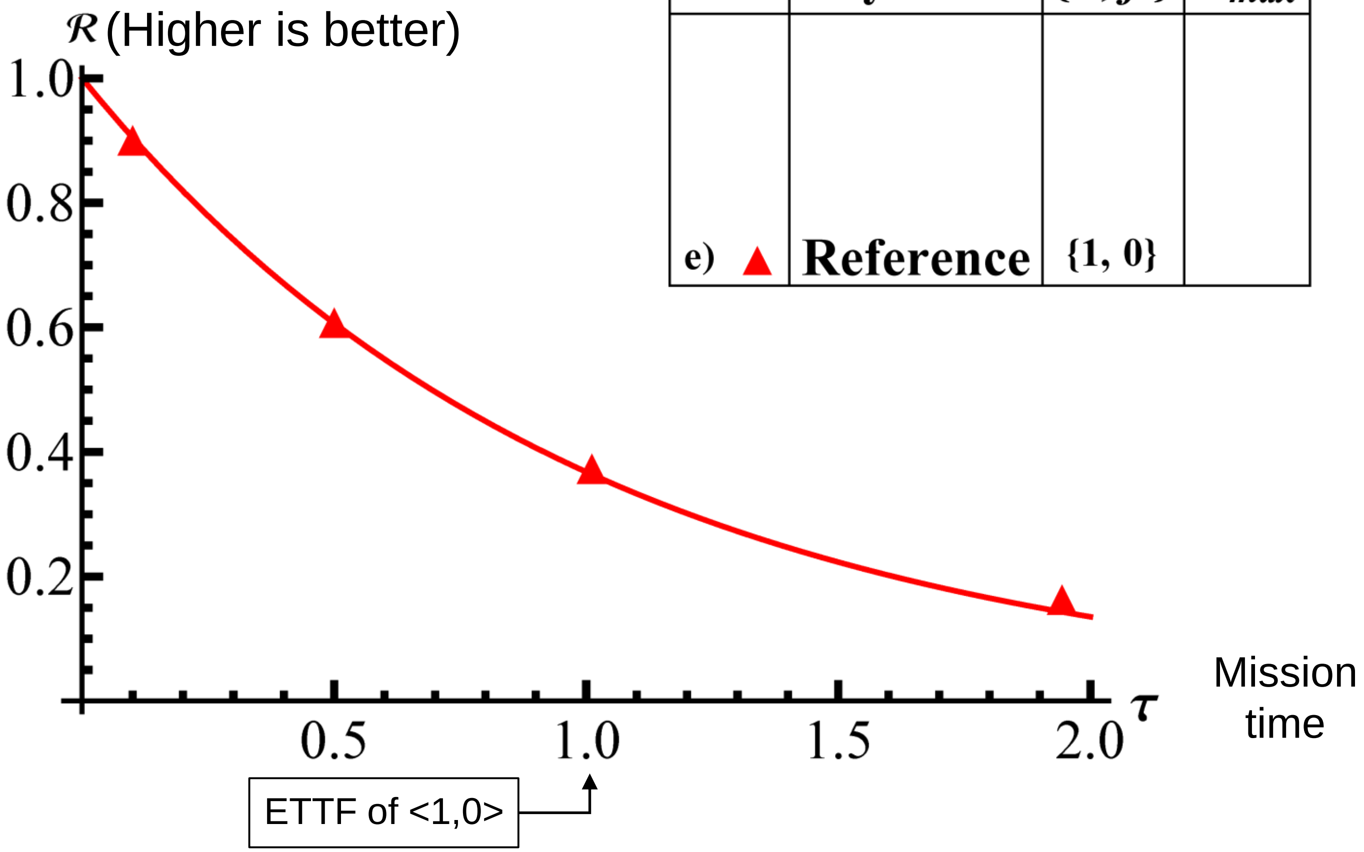
One metric of security: reliability

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

A possible model: nodes fail independently, with constant rate probability.

Example: ETTF per node under attack is 1 month

(ETTF = Expected time to failure)



Id	System	$\{n, f\}$	τ_{max}
e) ▲	**Reference**	{1, 0}	

n nodes; tolerating up to f faulty nodes.

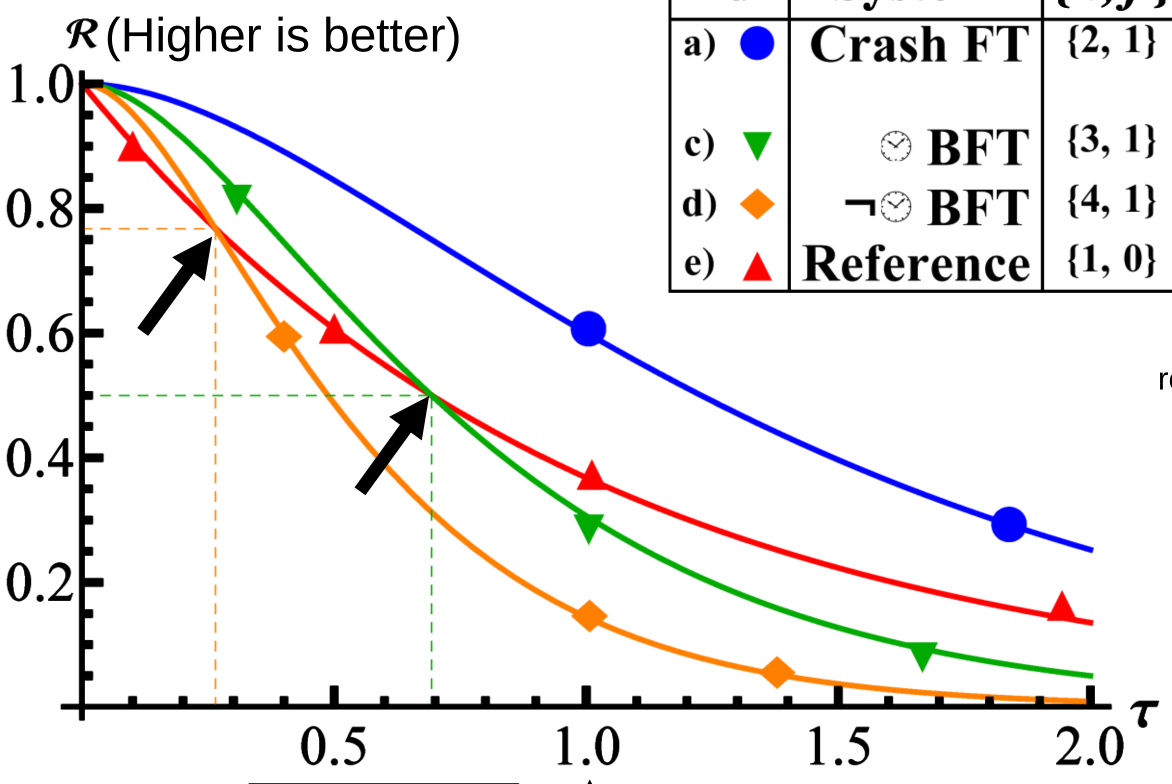
One metric of security: reliability

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

A possible model: nodes fail independently, with constant rate probability.

Example: ETTF per node under attack is 1 month

(ETTF = Expected time to failure)



Mission time

ETTF of <1,0>

Id	System	{n, f}	τ_{max}
a) ●	Crash FT	{2, 1}	∞
c) ▼	⏲ BFT	{3, 1}	0.693
d) ◆	¬⏲ BFT	{4, 1}	0.264
e) ▲	Reference	{1, 0}	

n nodes; tolerating up to f faulty nodes.

BFT (Byzantine fault tolerance) often requires n ≥ 3f+1 (or ≥ 2f+1 with special assumptions) to tolerate f faults.

One metric of security: reliability

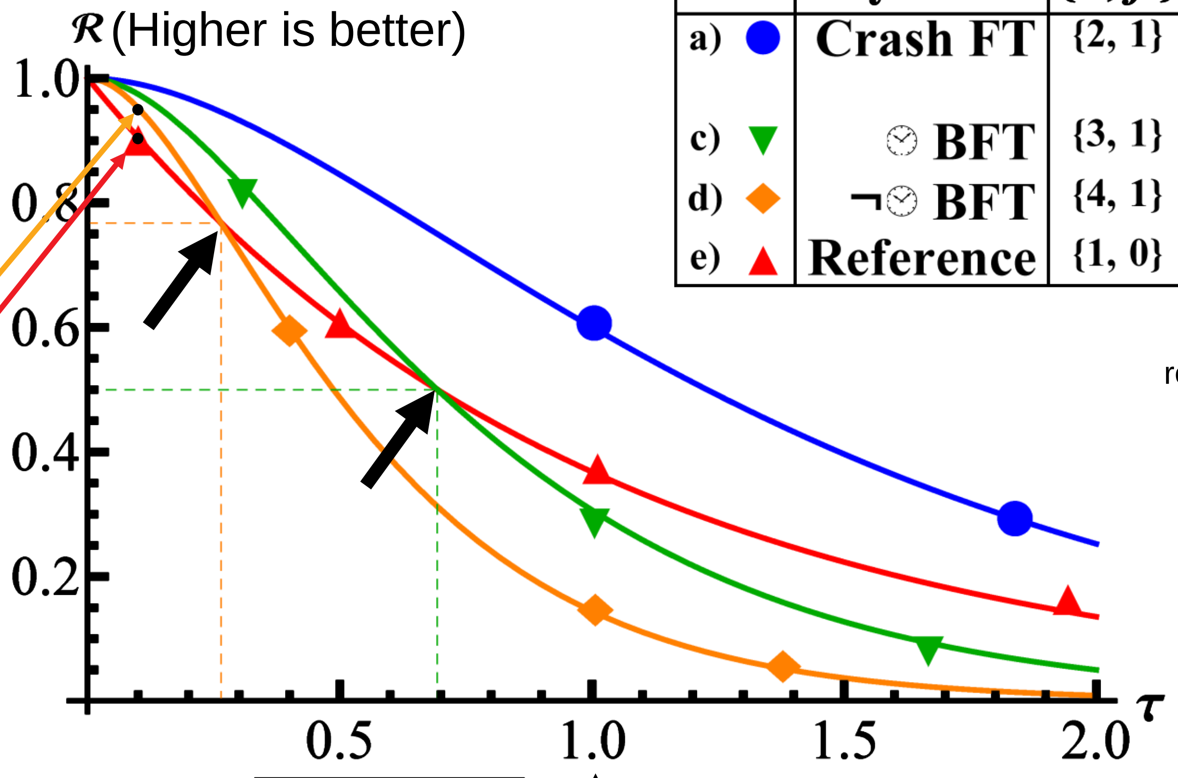
Probability that a security property (e.g., secrecy or integrity) never fails during a mission time

A possible model: nodes fail independently, with constant rate probability.

Example: ETTF per node under attack is 1 month

(ETTF = Expected time to failure)

For 3 days:
(Better) $R_{4,1}(0.1) \approx$ **0.95**
$R_{1,0}(0.1) \approx$ **0.91**



ETTF of <1,0>

Mission time

Id	System	{n, f}	τ_{max}
a) ●	Crash FT	{2, 1}	∞
c) ▼	⏲ BFT	{3, 1}	0.693
d) ◆	¬⏲ BFT	{4, 1}	0.264
e) ▲	Reference	{1, 0}	

n nodes; tolerating up to f faulty nodes.

BFT (Byzantine fault tolerance) often requires n ≥ 3f+1 (or ≥ 2f+1 with special assumptions) to tolerate f faults.

One metric of security: reliability

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

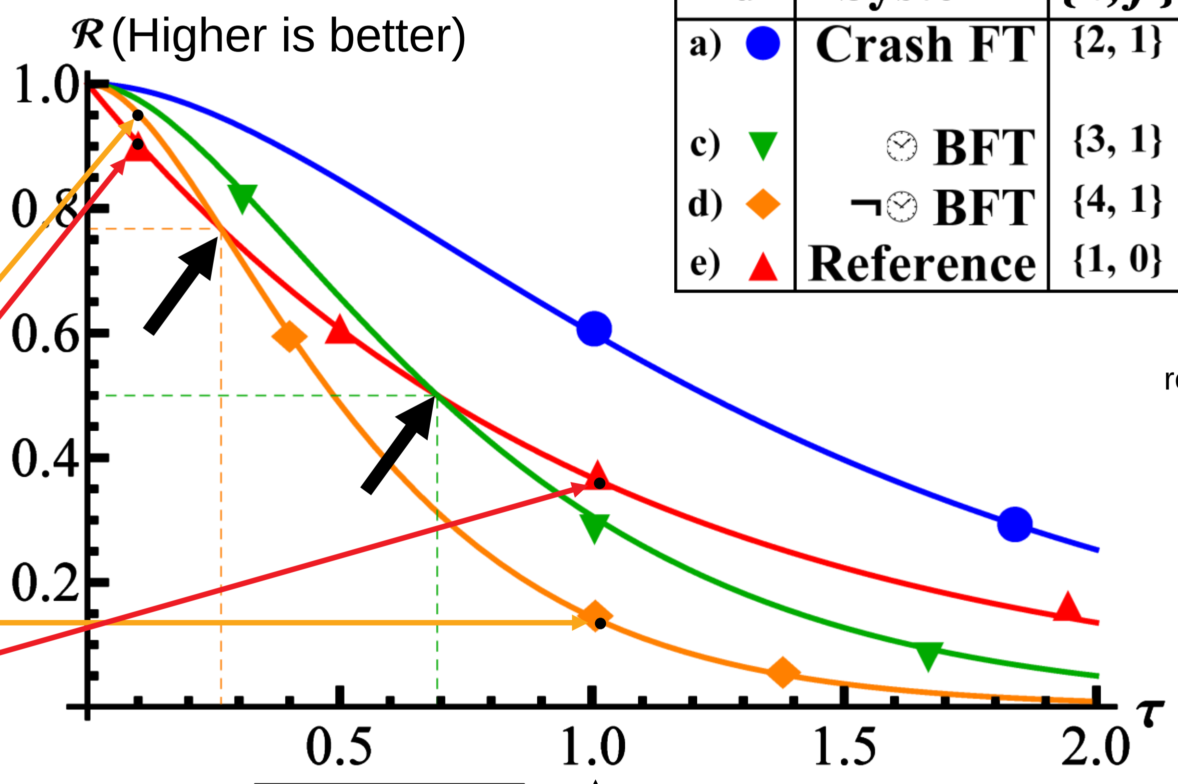
A possible model: nodes fail independently, with constant rate probability.

Example: ETTF per node under attack is 1 month

(ETTF = Expected time to failure)

For 3 days:
(Better) $R_{4,1}(0.1) \approx$ **0.95**
$R_{1,0}(0.1) \approx$ **0.91**

For 30 days:
$R_{4,1}(1.0) \approx$ **0.14**
(Better) $R_{1,0}(1.0) \approx$ **0.37**



Mission time

ETTF of <1,0>

Id	System	{n, f}	τ_{max}
a) ●	Crash FT	{2, 1}	∞
c) ▼	⏲ BFT	{3, 1}	0.693
d) ◆	¬⏲ BFT	{4, 1}	0.264
e) ▲	Reference	{1, 0}	

n nodes; tolerating up to f faulty nodes.

BFT (Byzantine fault tolerance) often requires n ≥ 3f+1 (or ≥ 2f+1 with special assumptions) to tolerate f faults.

Graphic source: https://doi.org/10.1007/s13173-012-0062-x

One metric of security: reliability

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

A possible model: nodes fail independently, with constant rate probability.

Example: ETTF per node under attack is 1 month

(ETTF = Expected time to failure)

For 3 days:

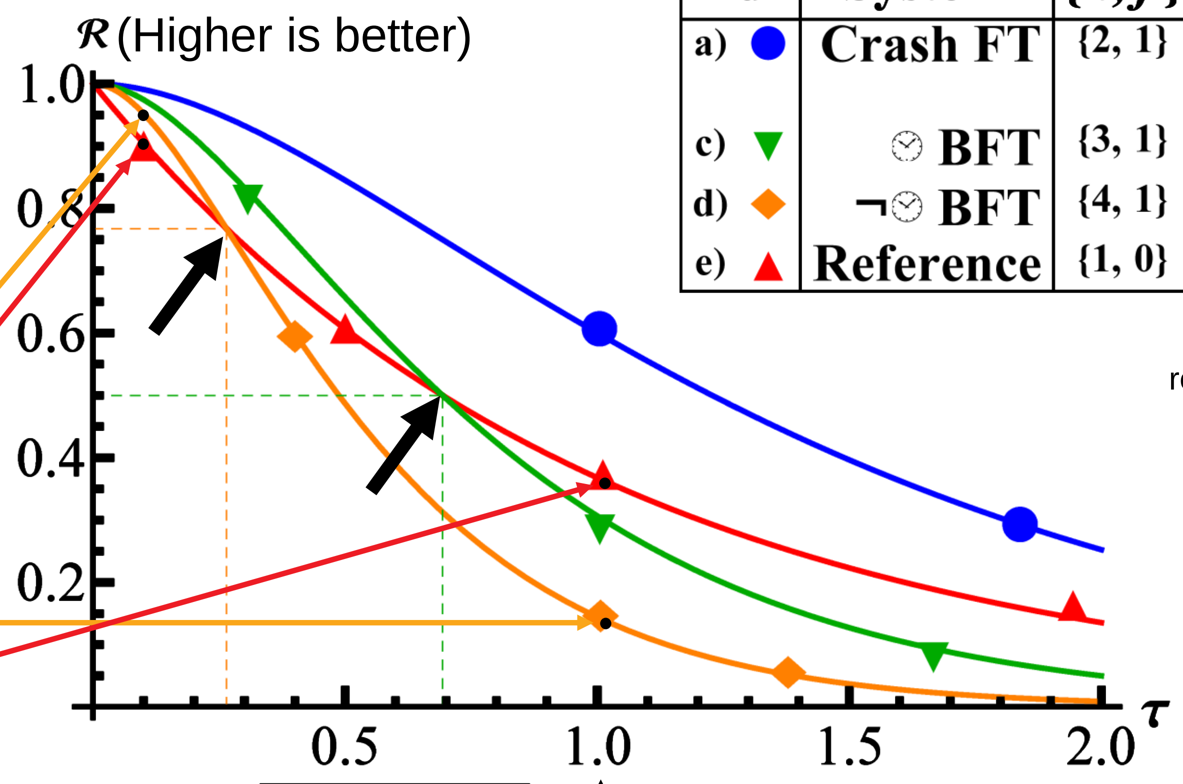
(Better) $R_{4,1}(0.1) \approx$ **0.95**

$R_{1,0}(0.1) \approx$ **0.91**

For 30 days:

$R_{4,1}(1.0) \approx$ **0.14**

(Better) $R_{1,0}(1.0) \approx$ **0.37**



ETTF of <1,0>

Id	System	$\{n, f\}$	τ_{max}
a) ●	Crash FT	{2, 1}	∞
c) ▼	⏲ BFT	{3, 1}	0.693
d) ◆	¬⏲ BFT	{4, 1}	0.264
e) ▲	Reference	{1, 0}	

n nodes; tolerating up to f faulty nodes.

BFT (Byzantine fault tolerance) often requires n ≥ 3f+1 (or ≥ 2f+1 with special assumptions) to tolerate f faults.

Mission time

Graphic source: https://doi.org/10.1007/s13173-012-0062-x

Reliability can be degraded when increasing the threshold (f), even if nodes fail independently.

Improve reliability with rejuvenations

Improve reliability with rejuvenations

Recover nodes: compromised state $\rightarrow$ healthy state

- Examples: replace device, patch vulnerability, update or reset a state, …
- Rejuvenations attenuate (but do not remove) the reliability degradation of long mission time

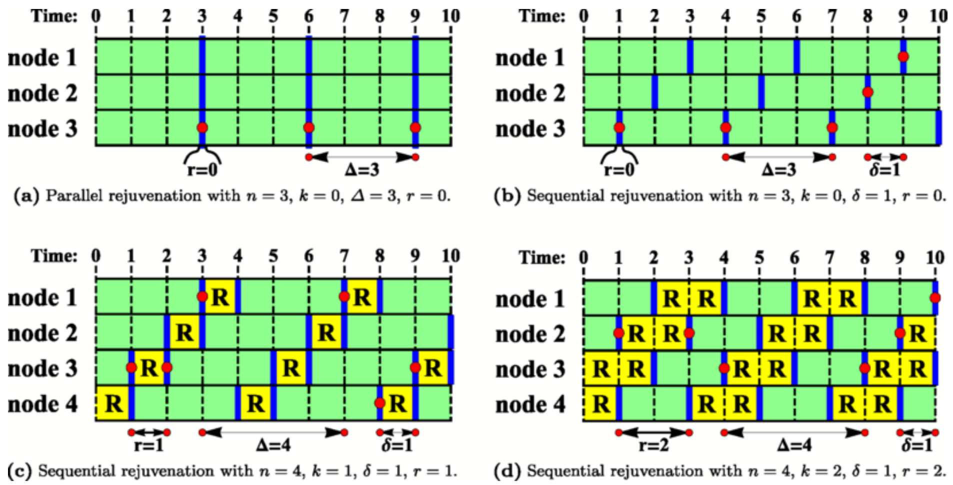
Improve reliability with rejuvenations

Recover nodes: compromised state → healthy state

- Examples: replace device, patch vulnerability, update or reset a state, …
- Rejuvenations attenuate (but do not remove) the reliability degradation of long mission time

Rejuvenation modes:

- **parallel vs. sequentially**
- **online vs. offline**



Graphic source: https://doi.org/10.1007/s13173-012-0062-x

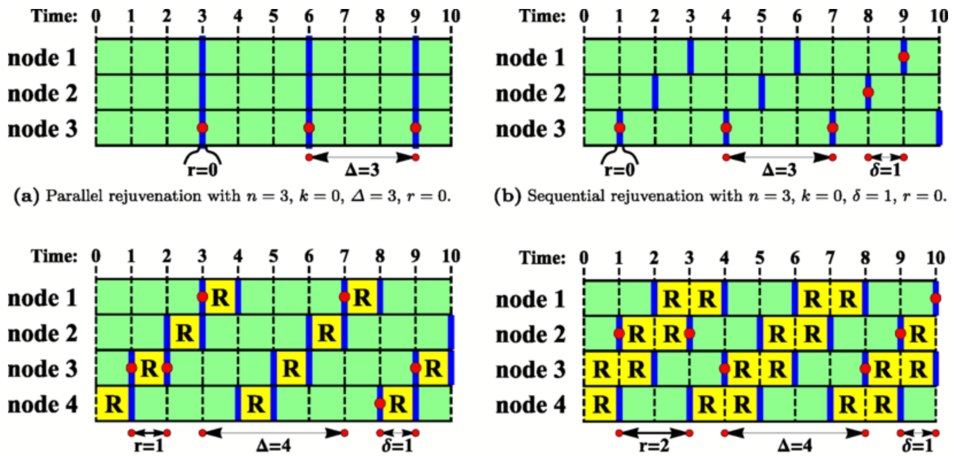
Improve reliability with rejuvenations

Recover nodes: compromised state → healthy state

- Examples: replace device, patch vulnerability, update or reset a state, …
- Rejuvenations attenuate (but do not remove) the reliability degradation of long mission time

Rejuvenation modes:

- **parallel vs. sequentially**
- **online vs. offline**
- **reactively (if detected intrusion) vs. proactively (for stealth scenario; which frequency?)**



(a) Parallel rejuvenation with $n = 3$, $k = 0$, $\Delta = 3$, $r = 0$.
(b) Sequential rejuvenation with $n = 3$, $k = 0$, $\delta = 1$, $r = 0$.
(c) Sequential rejuvenation with $n = 4$, $k = 1$, $\delta = 1$, $r = 1$.
(d) Sequential rejuvenation with $n = 4$, $k = 2$, $\delta = 1$, $r = 2$.

Graphic source: https://doi.org/10.1007/s13173-012-0062-x

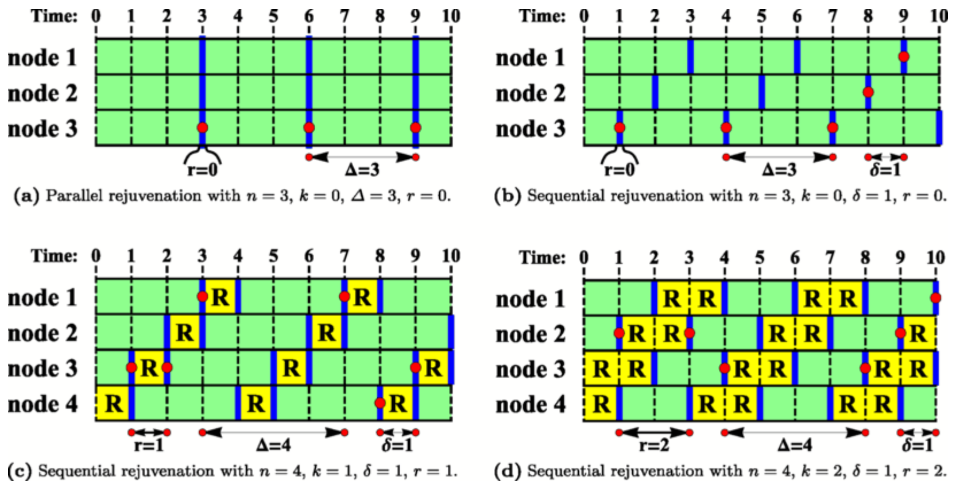
Improve reliability with rejuvenations

Recover nodes: compromised state → healthy state

- Examples: replace device, patch vulnerability, update or reset a state, …
- Rejuvenations attenuate (but do not remove) the reliability degradation of long mission time

Rejuvenation modes:

- **parallel vs. sequentially**
- **online vs. offline**
- **reactively (if detected intrusion) vs. proactively (for stealth scenario; which frequency?)**



(a) Parallel rejuvenation with $n = 3$, $k = 0$, $\Delta = 3$, $r = 0$.
(b) Sequential rejuvenation with $n = 3$, $k = 0$, $\delta = 1$, $r = 0$.
(c) Sequential rejuvenation with $n = 4$, $k = 1$, $\delta = 1$, $r = 1$.
(d) Sequential rejuvenation with $n = 4$, $k = 2$, $\delta = 1$, $r = 2$.

Graphic source: https://doi.org/10.1007/s13173-012-0062-x

Effects:

- may add cost, implementation complexity, new (?) vulnerabilities
- sequential rejuvenations may allow a mobile attacker to persist
- parallel offline rejuvenation may imply period of unavailable service
- increases **availability** (another metric: % secure time), even for ∞ mission time

Another model

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 needs exponential

number of traces for high-order Differential Power Analysis)

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 needs exponential

number of traces for high-order Differential Power Analysis)

Case scenario: encryption circuit with n-out-of-n threshold implementation (design based on secret-sharing & SMPC).

Key remains secret while attacker does not find the bits in *n* wires.

(but attacker cannot directly probe the wires)

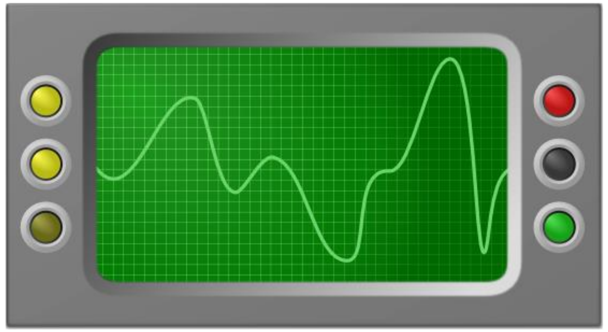


Image source: openclipart.org/detail/172330

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 needs exponential number of traces for high-order Differential Power Analysis)

Case scenario: encryption circuit with n-out-of-n threshold implementation (design based on secret-sharing & SMPC).
Key remains secret while attacker does not find the bits in *n* wires.
(but attacker cannot directly probe the wires)

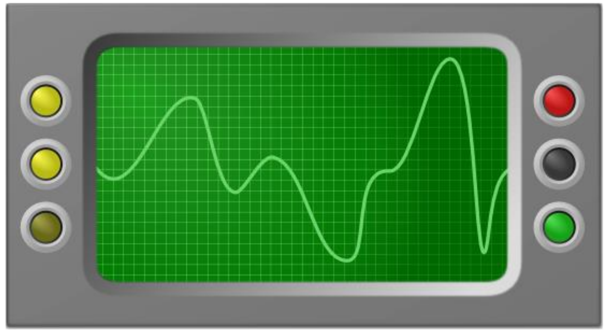


Image source: openclipart.org/detail/172330

Challenge questions:

- which models are realistic / match state-of-the-art attacks?
- what are concrete parameters (e.g., *n*) that make a real attack infeasible?
- what is the exploitation-complexity for other attacks? …

Outline

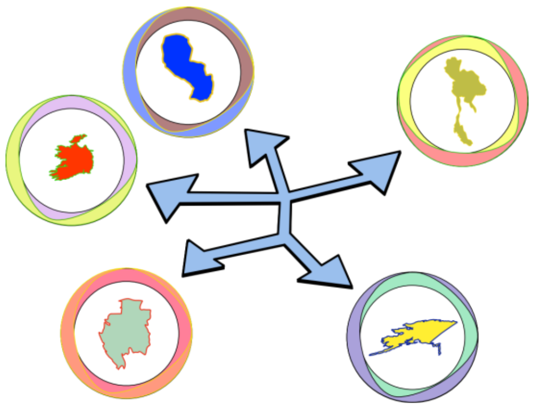
1. Introduction: we need reliable crypto
2. Validating a crypto module (the CMVP at NIST)
3. The threshold approach
4. Characterizing a threshold scheme
5. The threshold validation challenge
6. Concluding remarks

Characterizing features (1–2)

Characterizing features (1–2)

1. Kinds of threshold

- Need *k*-out-of-*n* good ones (or tolerate up to *f*-out-of-*n* bad ones) for which values *k* and *f* ? for which security properties?

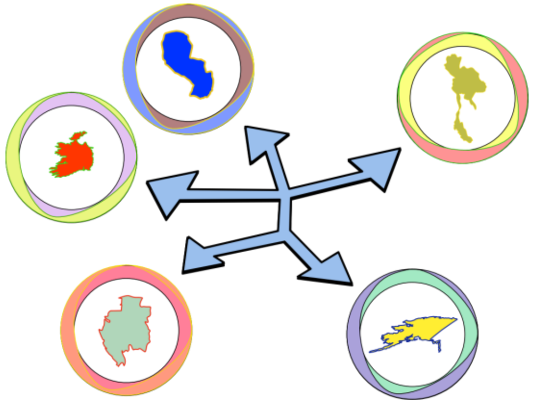


openclipart.org/detail/71491

Characterizing features (1–2)

<u>1. Kinds of threshold</u>

- Need *k*-out-of-*n* good ones (or tolerate up to *f*-out-of-*n* bad ones) for which values *k* and *f* ? for which security properties?
- Levels of *diversity* (e.g., location, software, shares) vs. *non-diversity* across the n components (common vulnerabilities?)

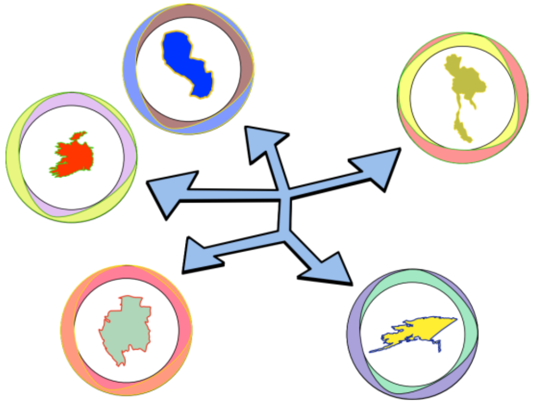


openclipart.org/detail/71491

Characterizing features (1–2)

1. Kinds of threshold

- Need *k*-out-of-*n* good ones (or tolerate up to *f*-out-of-*n* bad ones) for which values *k* and *f* ? for which security properties?
- Levels of *diversity* (e.g., location, software, shares) vs. *non-diversity* across the n components (common vulnerabilities?)



openclipart.org/detail/71491

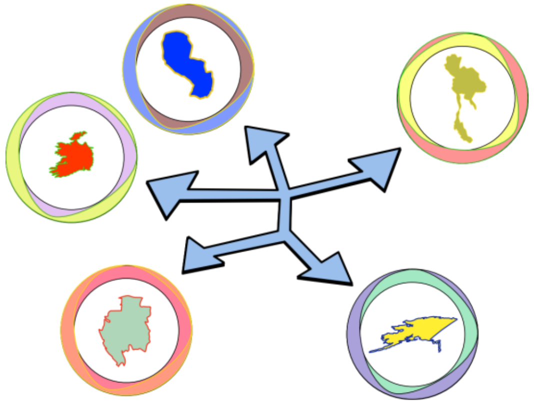
2. Communication interfaces

openclipart.org/detail/190624

Characterizing features (1–2)

1. Kinds of threshold

- Need *k*-out-of-*n* good ones (or tolerate up to *f*-out-of-*n* bad ones) for which values *k* and *f* ? for which security properties?
- Levels of *diversity* (e.g., location, software, shares) vs. *non-diversity* across the n components (common vulnerabilities?)



openclipart.org/detail/71491

2. Communication interfaces

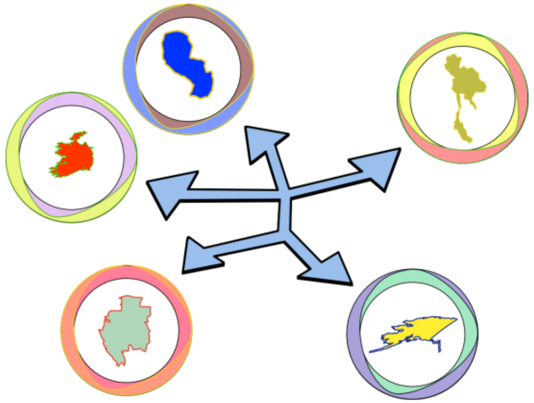
- Client ↔ crypto module: proxy? primary node? shares? (is client aware of threshold scheme?)

openclipart.org/detail/190624

Characterizing features (1–2)

1. Kinds of threshold

- Need *k*-out-of-*n* good ones (or tolerate up to *f*-out-of-*n* bad ones) for which values *k* and *f* ? for which security properties?
- Levels of *diversity* (e.g., location, software, shares) vs. *non-diversity* across the n components (common vulnerabilities?)



openclipart.org/detail/71491

2. Communication interfaces

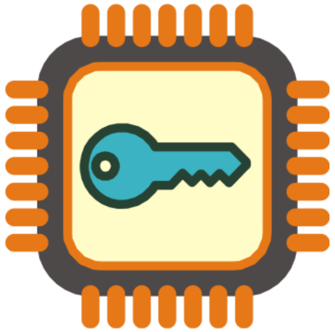
- Client ↔ crypto module: proxy? primary node? shares? (is client aware of threshold scheme?)
- Inter-node: structure (e.g., star vs. clique)? channel protection?

openclipart.org/detail/190624

Characterizing features (3–4)

Characterizing features (3–4)

3. Executing platform

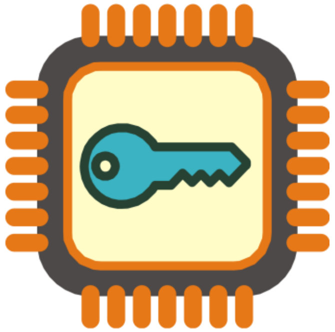


openclipart.org/detail/101407/

Characterizing features (3–4)

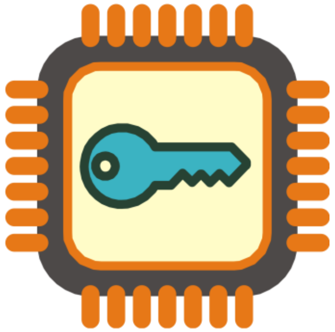
3. Executing platform

- Single (multi-chip) device vs. multi-party (e.g., multiple computers)



openclipart.org/detail/101407/

Characterizing features (3–4)


openclipart.org/detail/101407/

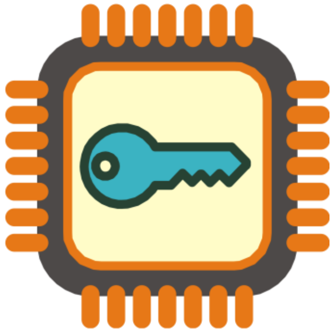
3. Executing platform

- Single (multi-chip) device vs. multi-party (e.g., multiple computers)
- Software vs. hardware

Characterizing features (3–4)

3. Executing platform

- Single (multi-chip) device vs. multi-party (e.g., multiple computers)
- Software vs. hardware
- Need additional machinery? (trusted global clock, proxy, RNG, combiner)

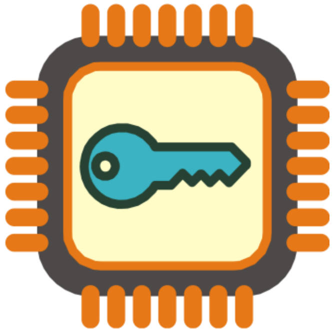


openclipart.org/detail/101407/

Characterizing features (3–4)

3. Executing platform

- Single (multi-chip) device vs. multi-party (e.g., multiple computers)
- Software vs. hardware
- Need additional machinery? (trusted global clock, proxy, RNG, combiner)

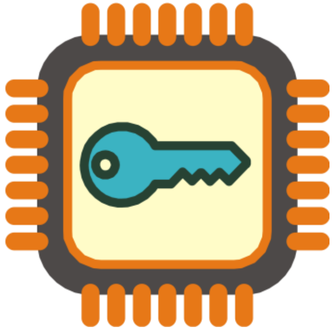


openclipart.org/detail/101407/

4. Setup (bootstrap) and rejuvenation / recovery

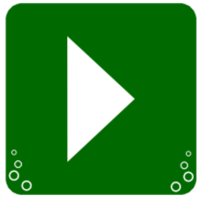
Characterizing features (3–4)

3. Executing platform


openclipart.org/detail/101407/

- Single (multi-chip) device vs. multi-party (e.g., multiple computers)
- Software vs. hardware
- Need additional machinery? (trusted global clock, proxy, RNG, combiner)

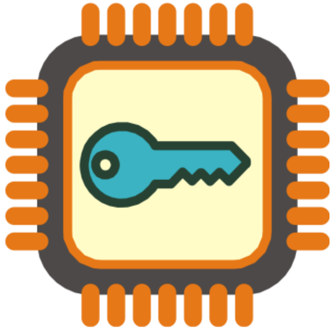
4. Setup (bootstrap) and rejuvenation / recovery



- How to bootstrap: dealer vs. secure multi-party initialization of secret shares

Characterizing features (3–4)

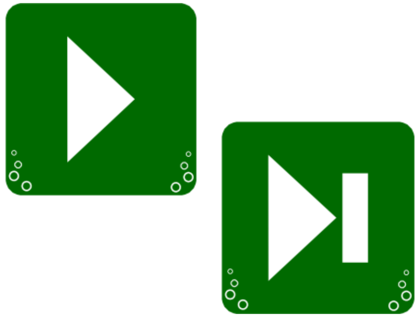
3. Executing platform



openclipart.org/detail/101407/

- Single (multi-chip) device vs. multi-party (e.g., multiple computers)
- Software vs. hardware
- Need additional machinery? (trusted global clock, proxy, RNG, combiner)

4. Setup (bootstrap) and rejuvenation / recovery



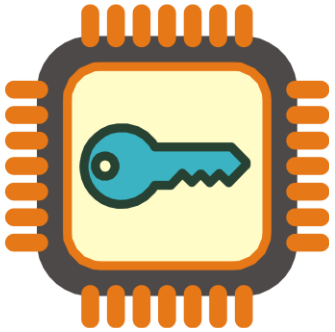
openclipart.org/detail/*
* in {161401, 161389}

- How to bootstrap: dealer vs. secure multi-party initialization of secret shares
- Rejuvenation modes? (reactive vs. proactive, parallel vs. sequential)

Characterizing features (3–4)

3. Executing platform

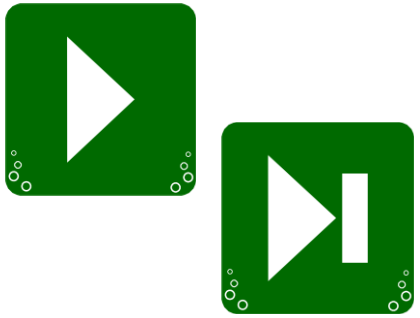
- Single (multi-chip) device vs. multi-party (e.g., multiple computers)
- Software vs. hardware
- Need additional machinery? (trusted global clock, proxy, RNG, combiner)



openclipart.org/detail/101407/

4. Setup (bootstrap) and rejuvenation / recovery

- How to bootstrap: dealer vs. secure multi-party initialization of secret shares
- Rejuvenation modes? (reactive vs. proactive, parallel vs. sequential)
- Diversity: offline pre-computation vs. on-the-fly vs. limited set



openclipart.org/detail/*
* in {161401, 161389}

Additional considerations

Additional considerations

- **Performance.** How (in)efficient is the threshold vs. non-threshold version?

openclipart.org/detail/291407

Additional considerations

- **Performance.** How (in)efficient is the threshold vs. non-threshold version?
- **Operational pros & cons.** Isolated patch may become trivial in multi-party setting; can testing components become more difficult?

openclipart.org/detail/291407


openclipart.org/detail/22712

Additional considerations

- **Performance.** How (in)efficient is the threshold vs. non-threshold version?
- **Operational pros & cons.** Isolated patch may become trivial in multi-party setting; can testing components become more difficult?
- **Application context.** Should it affect security requirements? E.g., signature may ignore concerns of integrity, if app layer verifies correctness. Encryption more difficult?

openclipart.org/detail/291407


openclipart.org/detail/22712


openclipart.org/detail/263691, 281637

Additional considerations

- **Performance.** How (in)efficient is the threshold vs. non-threshold version?
- **Operational pros & cons.** Isolated patch may become trivial in multi-party setting; can testing components become more difficult?
- **Application context.** Should it affect security requirements? E.g., signature may ignore concerns of integrity, if app layer verifies correctness. Encryption more difficult?
- **Conceived attack types.** - active vs. passive; - static vs. adaptive; - stealth vs. detected - invasive (physical) vs. non-invasive; - side-channel vs. comm. interfaces; - parallel vs. sequential (wrt attacking nodes); …

openclipart.org/detail/291407


openclipart.org/detail/22712


openclipart.org/detail/263691, 281637


clker.com/clipart-10778.html

Additional considerations

- **Performance.** How (in)efficient is the threshold vs. non-threshold version?
- **Operational pros & cons.** Isolated patch may become trivial in multi-party setting; can testing components become more difficult?
- **Application context.** Should it affect security requirements? E.g., signature may ignore concerns of integrity, if app layer verifies correctness. Encryption more difficult?
- **Conceived attack types.** - active vs. passive; - static vs. adaptive; - stealth vs. detected - invasive (physical) vs. non-invasive; - side-channel vs. comm. interfaces; - parallel vs. sequential (wrt attacking nodes); …

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

openclipart.org/detail/291407


openclipart.org/detail/22712


openclipart.org/detail/263691, 281637


clker.com/clipart-10778.html

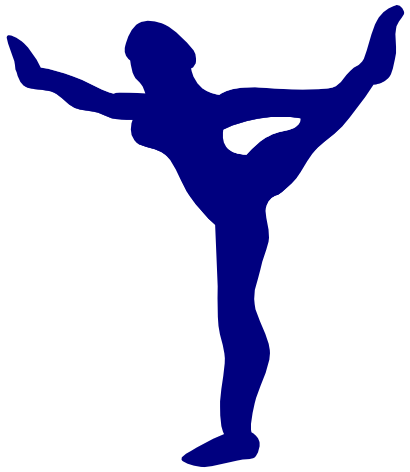
Outline

1. Introduction: we need reliable crypto
2. Validating a crypto module (the CMVP at NIST)
3. The threshold approach
4. Characterizing a threshold scheme
5. The threshold validation challenge
6. Concluding remarks

A main question: flexibility?

A main question: flexibility?

What flexibility of features & parameters should a threshold-scheme standard allow?



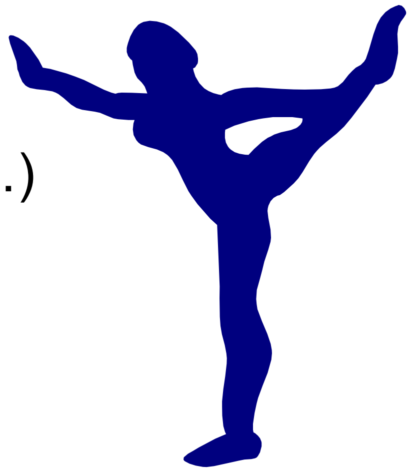
clker.com/clipart-stretching-navy.html

A main question: flexibility?

What flexibility of features & parameters should a threshold-scheme standard allow?

What should then be delimited at validation phase

(e.g., validated only for n ≥ 2f+1; particular hardware; shares initialized with SMPC, …)



clker.com/clipart-stretching-navy.html

A main question: flexibility?

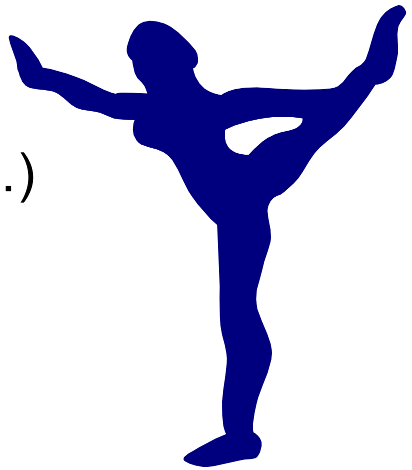
What flexibility of features & parameters should a threshold-scheme standard allow?

What should then be delimited at validation phase

(e.g., validated only for $n \geq 2f+1$; particular hardware; shares initialized with SMPC, …)

What may remain flexible for deployment?

(e.g., f ; how to (re-)initialize shares: dealer vs. SMPC?, …)



clker.com/clipart-stretching-navy.html

A main question: flexibility?

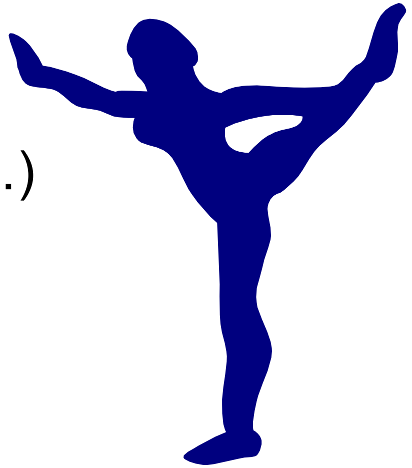
What flexibility of features & parameters should a threshold-scheme standard allow?

What should then be delimited at validation phase

(e.g., validated only for $n \geq 2f+1$; particular hardware; shares initialized with SMPC, …)

What may remain flexible for deployment?

(e.g., *f* ; how to (re-)initialize shares: dealer vs. SMPC?, …)



clker.com/clipart-stretching-navy.html

What can be directly validated vs. what must rely on vendor assertion / deployment?

- How to be sure that good randomness will be used?
- How to validate schedule of rejuvenations, ensure appropriate diversity? …

A main question: flexibility?

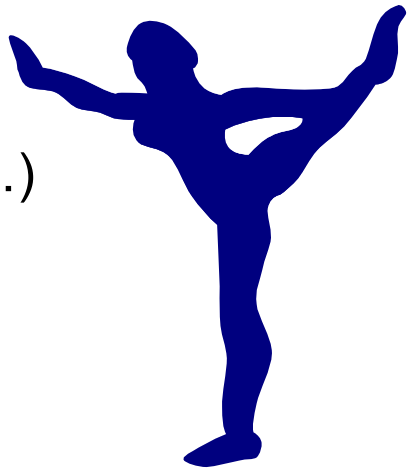
What flexibility of features & parameters should a threshold-scheme standard allow?

What should then be delimited at validation phase

(e.g., validated only for n ≥ 2f+1; particular hardware; shares initialized with SMPC, …)

What may remain flexible for deployment?

(e.g., *f* ; how to (re-)initialize shares: dealer vs. SMPC?, …)



clker.com/clipart-stretching-navy.html

What can be directly validated vs. what must rely on vendor assertion / deployment?

- How to be sure that good randomness will be used?
- How to validate schedule of rejuvenations, ensure appropriate diversity? …

Answers may to a certain extent depend on what can be assessed by test & validation procedures (some of which to develop)!

Other questions about validation

Other questions about validation

Checklist: should the validation level (= a set of security assertions) contain a checklist of attack scenarios and security properties?

- pairs <attack, security property> for which is the scheme is considered okay
- expected / adequate parameters for different conceived attacks

Other questions about validation

Checklist: should the validation level (= a set of security assertions) contain a checklist of attack scenarios and security properties?

- pairs <attack, security property> for which is the scheme is considered okay
- expected / adequate parameters for different conceived attacks

pixabay.com/en/checklist-clipboard-questionnaire-1622517/

Adaptation: How much should validation procedures / levels / assertions adapt to threshold features and/or application context?

- Validation already currently have differences between platforms (software / hardware)
- Standards of crypto primitives are usually independent of platform, but there may be a foreseen deployment (e.g., light-weight crypto)

Other questions about validation

Checklist: should the validation level (= a set of security assertions) contain a checklist of attack scenarios and security properties?

- pairs <attack, security property> for which is the scheme is considered okay
- expected / adequate parameters for different conceived attacks

pixabay.com/en/checklist-clipboard-questionnaire-1622517/

Adaptation: How much should validation procedures / levels / assertions adapt to threshold features and/or application context?

- Validation already currently have differences between platforms (software / hardware)
- Standards of crypto primitives are usually independent of platform, but there may be a foreseen deployment (e.g., light-weight crypto)

Base primitives: Should some base primitives be independently standardized / validated?

- Could some base primitives (e.g., secret sharing, oblivious transfer, commitments) be useful for the validation of a complex threshold scheme with flexible parameters? (composability argument)

What test/validation procedures do develop?

What test/validation procedures do develop?

- That implementation is consistent with described features & parameters

clker.com/clipart-8522.html

What test/validation procedures do develop?

clker.com/clipart-8522.html

- That implementation is consistent with described features & parameters
- Develop test suites (including automated ones):
  - Validate functional properties of expected threshold protocol
  - Removing *n*-*k* components does not affect output (e.g., use input/output test vectors)
  - Exercise erroneous behavior by up to *f* components
  - Interfere with communication channels (inter-node and client to module)
  - Exercise rejuvenation of components
  - ...

What test/validation procedures do develop?

clker.com/clipart-8522.html

- That implementation is consistent with described features & parameters
- Develop test suites (including automated ones):
    - Validate functional properties of expected threshold protocol
    - Removing *n*-*k* components does not affect output (e.g., use input/output test vectors)
    - Exercise erroneous behavior by up to *f* components
    - Interfere with communication channels (inter-node and client to module)
    - Exercise rejuvenation of components
    - ...
- Other generic tests: code analysis, side-channel resistance, …

What test/validation procedures do develop?

clker.com/clipart-8522.html

- That implementation is consistent with described features & parameters
- Develop test suites (including automated ones):
  - Validate functional properties of expected threshold protocol
  - Removing *n*-*k* components does not affect output (e.g., use input/output test vectors)
  - Exercise erroneous behavior by up to *f* components
  - Interfere with communication channels (inter-node and client to module)
  - Exercise rejuvenation of components
  - ...
- Other generic tests: code analysis, side-channel resistance, …

(Can we try to predict likely types of bugs when implementing a threshold scheme?)

Outline

1. Introduction: we need reliable crypto
2. Validating a crypto module (the CMVP at NIST)
3. The threshold approach
4. Characterizing a threshold scheme
5. The threshold validation challenge
6. Concluding remarks

In summary

In summary

Is a threshold scheme more secure than a non-threshold one? It depends!

In summary

Is a threshold scheme more secure than a non-threshold one? It depends!

To assess security effects, we should characterize:

- Features of the threshold scheme
- Adversarial model: goals, capabilities, vectors
- Different effects (improve vs. degrade) on different security properties of interest
- New complexity from threshold approach (e.g., likely bugs, vulnerable extra components), …

In summary

Is a threshold scheme more secure than a non-threshold one? It depends!

To assess security effects, we should characterize:

- Features of the threshold scheme
- Adversarial model: goals, capabilities, vectors
- Different effects (improve vs. degrade) on different security properties of interest
- New complexity from threshold approach (e.g., likely bugs, vulnerable extra components), …

Standardizing a chosen scheme also entails:

- Deciding what remains flexible up to validation and/or deployment phases
- Develop test procedures and security assertions for validation

Next steps

Next steps

- We are working on a report about the subject
  - It's about positioning a set of questions — what/how should we look at?
  - To raise awareness, we published a short article “Psst, can you keep a secret?”, to appear in IEEE Computer (Jan 2018)

clker.com/clipart-4281.html

Next steps

- We are working on a report about the subject
  - It's about positioning a set of questions — what/how should we look at?
  - To raise awareness, we published a short article “Psst, can you keep a secret?”, to appear in IEEE Computer (Jan 2018)
- The report will be published for public comments
  - We'll be looking for your feedback

clker.com/clipart-4281.html

Next steps

- We are working on a report about the subject
  - It's about positioning a set of questions — what/how should we look at?
  - To raise awareness, we published a short article “Psst, can you keep a secret?”, to appear in IEEE Computer (Jan 2018)
- The report will be published for public comments
  - We'll be looking for your feedback
- A constructive process may then consider concrete proposals and a procedure for running the standardization effort

clker.com/clipart-4281.html

Next steps

- We are working on a report about the subject
  - It's about positioning a set of questions — what/how should we look at?
  - To raise awareness, we published a short article “Psst, can you keep a secret?”, to appear in IEEE Computer (Jan 2018)
- The report will be published for public comments
  - We'll be looking for your feedback
- A constructive process may then consider concrete proposals and a procedure for running the standardization effort
- The end goal:

  - standardize threshold schemes for cryptographic primitives

  - develop guidelines for validation

  - promote good practices of deployment

clker.com/clipart-4281.html

Thank you for your attention

“Sizing up the threshold”

Presented at the 2nd Theory of Implementation Security Workshop

January 09, 2018 (Zurich, Switzerland)