Quality-Time-As-An-Advantage Zero-Pre-Configuration Pairing Scheme for seL4 IoT Devices/Networks

Yong Guan

Professor, Department of Electrical and Computer Engineering
Associate Director for Research, Information Assurance Center
Digital Forensics Coordinator, NIST Center of Excellence in Forensic Sciences (CSAFE)
Iowa State University

September 23, 2019

Our Research Foci

- Cyber Attacks and Crimes: A painful side-effect of the innovations of Computer and Internet technologies
  - Almost all physical crimes involve digital evidence
  - Low percentage of cases reported to law enforcement
- Our Research Foci in DF and Security:
  - Build Accountability & Incident Response
  - Security Monitoring & Impact Analysis
  - Human-centric Security Solutions
  - Hardware-assisted Security, OS
- **NIST CoE in Forensic Sciences 2015 - present**

NSF

U.S. Army Research Office

IARPA

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

A Complex "Two/Multiple Player Game - Chess" – Security and Forensics

Observable with complete and precise info

In cyberspace, do we have an observable "chessboard"?

No, more like a "Two or more players' game" with

Incomplete and imprecise info of each other

The Big Picture

- Mobile and Wearable Device, and IoT/RFID technology to become ubiquitous in near future.
- Applications in all aspects of our lives:
  - where we work (industrial platforms, transportation);
  - where we shop (retail stores, restaurants);
  - where we live (home environments, smart health);
  - where we recover (medical facilities);
  - and coming soon to your local funeral parlor.

["We cook your meals, we haul your trash, we connect your calls, we drive your ambulances, we guard you while you sleep." – *Fight Club*, 1999]

- Main drawback: Subject to S&P *attacks*:
  - generated several small-scale protests;
  - could have serious social and legal consequences.

Loss of Privacy – IoT & RFID-enabled Apps

Pairing Problem and Privacy

- IoT/Tags should only respond with sensitive information to, or accept commands from *trusted* parities,.
- To establish trust – some form of pairing is required:
  1. implicit (tag's secret key is distributed to trustworthy readers);
  2. explicit (reader and tag undergo a pairing protocol);

1. Secret keys are *impractical to distribute along supply chain*;
2. Pairing protocols: legitimate readers need *some form of advantage* over malicious readers.

Ideally, zero-configuration!

Literature Survey

Several procedures for key pairing/establishment:

- Public-key infrastructure (using certificates)
- Centralized systems/ key distribution centers (based on symmetric key encryption/ authentication)
- Out-of-band channels
- Superior-quality channels (the "eavesdropper channel"/wiretapping)
- Sources of common randomness (noisy signal from a satellite, network metadata, etc.)

Secure Pairing Using Time Advantage:
From "Adopted Pet (AP)" to Algebraic Protocol (WiSec'18)

- We introduced the AP pairing protocol, suited for supporting zero-configuration systems, in RFIDSec'11.
- AP protocol is automatic, and based on two principles:
  - Legitimate reader has the advantage of time;
  - Cipher weaknesses can be used constructively.

The Adopted Pet protocol was a first step towards a paradigm where authentication and security is based on the legitimate parties mounting successful attacks on each-other's cryptographic protocols, and where the work of anonymous attackers and hackers can serve as the basis for faster authentication and legitimate decryption.

Funded by NSF, published in RFIDSec'11 and ACM WiSec'18

The Adopted Pet (AP) Protocol

- We proposed the novel AP pairing protocol, which:
  - requires no human interaction;
  - is transient;
  - is aimed at commercial-level RFID privacy;
  - tolerates interference and de-synchronizations;
  - demands limited resources.

The main idea:

1. TIME AS A RESOURCE;
2. CONSTRUCTIVELY EXPLOIT CIPHER WEAKNESSES:
   - Tag's trust is earned by spending a *long, quality time* in its presence.
   - Long, uninterrupted time allows legitimate user to mount successful attack on tag's inner (weak) cipher.
   - Attacker cannot be near tag enough time to gain its trust, without being detected.

Time-based Key Establishment

The two legitimate parties' advantage over the potential eavesdropper:

- They can spend long, uninterrupted periods of time ("quality time") with each-other

This type of advantage has some benefits:

1. very natural – this is how animals establish trust;
2. appropriate for certain lightweight applications:
   - IoT/RFID readers spend quality time with tags in the home, retailer's and manufacturer's storage, etc.;
   - Military lightweight wireless sensors spend quality time with each other before deployment;
   - Home-area-network IoT devices spend most of their lives with each other;
3. requires no infrastructure or key pre-distribution;
4. requires no external source of common randomness.

System Model, Threat Model

- *Scenario 1—legitimate pairing:*
  - new tag arrives home; needs to work with smart fridge or wardrobe;
  - reader begins "courting" the tag throughout the night;
  - once reader has enough information, it proves that it learned tag's secret, and pairing is complete.

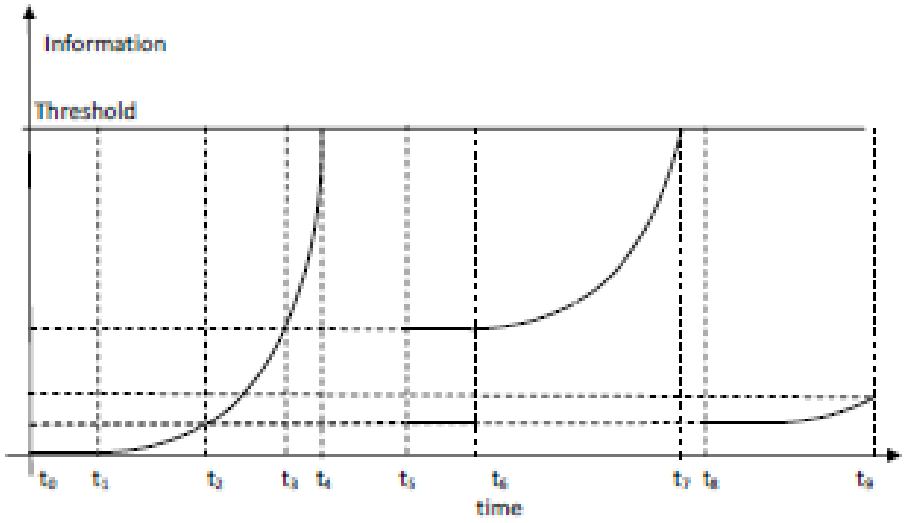
- *Scenario 2—the man on the bus:*
  - tag is carried on the bus, everyday, but only for several hours;
  - attacker rides same bus;
  - attacker's reader tries to pair with tag.

Challenges

- A certain amount of *uninterrupted time* (10 hours) allows legitimate reader to learn tag's secret;
- A larger amount of *interrupted time* (2 hours a day, for many days) does not reveal tag's secret to attacker;
- Passive RFID tags have no internal time reference:
  - tag cannot keep track of how much time it spends with reader;
  - small interruptions are possible even with legitimate reader (e.g., owner takes the trash out);
  - if the tag just counts number of queries from certain reader, "man on the bus" would eventually succeed.

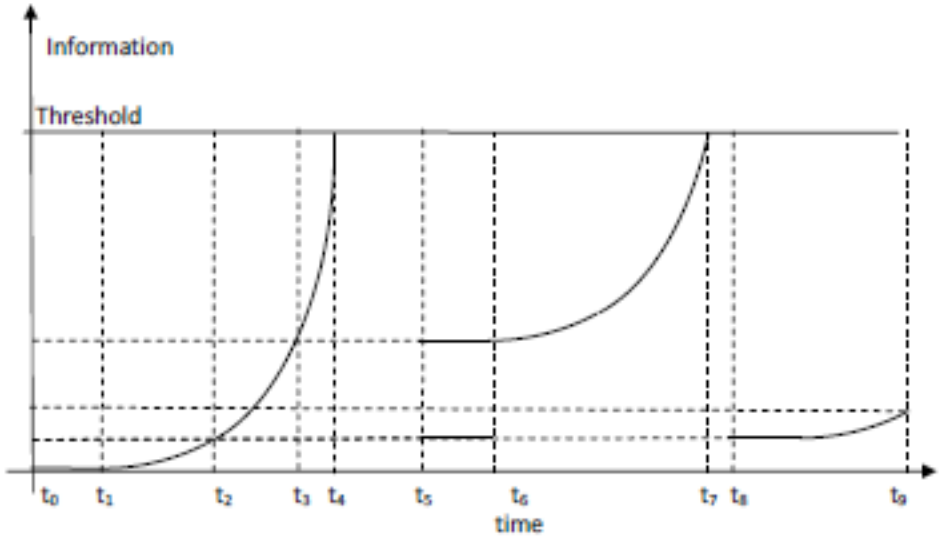
An Ideal Solution

- Tag contains inner secret;
- For every query from untrusted reader, tag responds with *clue*.
- Consecutive clues leak information about tag's secret exponentially over time (but only after time threshold $t1 - t0$).
- If reader and the tag become desynchronized, rate of gathering information returns to its initial value, and starts increasing from there (however, previous information not lost).



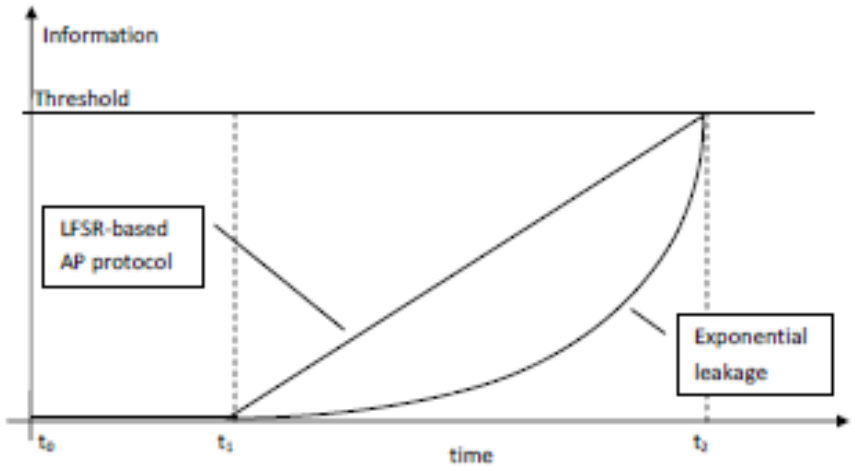
An Ideal Solution – Example

- Reader 1 interacts with tag continuously over [*t*0, *t*4];
- Reader 2 queries tag over distinct uninterrupted intervals [*t*0, *t*3] and [*t*5, *t*7];
- Attacker queries tag during intervals [*t*0, *t*2], [*t*5, *t*6], and [*t*8, *t*9], without reaching the information threshold.



A Practical Solution – AP Protocol

- Tag contains internal LFSR of length *L* and secret characteristic polynomial.
- A *clue*: LFSR is clocked and transmits one bit.
- After 2*L* consecutive clues, internal LFSR structure is known.
- No information leaks before *L* + 1st clue
- Information leaks at a linear rate (first-order approximation of ideal exponential leakage system).



Implementation Solutions

- We proposed and analyzed several types of implementations:
  - The bare LFSR (linear complexity too small);
  - Nonlinear Combination Generator (recommended);
  - Nonlinear Filter Generator (recommended);
  - Shrinking Generator (linear complexity too large).

Security and Functionality Analysis

- How does legitimate reader find secret characteristic polynomial of LFSR?
  - Gather 2*L consecutive* bits of LFSR output;
  - Solve system of *L* linear equations with *L* unknowns;
  - Gaussian elimination takes time $O(L^3)$, or $O(L^4)$ if *k* unknown;
  - Berlekamp-Massey algorithm [Massey,1969] more efficient.

Security and Functionality Analysis

- What if legitimate reader misses a small number *k* of bits in the middle?
  - If *L* + *M* consecutive bits were gathered before the de-synchronization, reader can gather 2*L* − *M* consecutive bits afterwards (a total of *L* lin. equations).
  - Alternatively, replace missed bits by unknowns and build system of *L* + *k* equations, with *L* + *k* unknowns (*equations are now quadratic!*)
  - Alternatively, brute-force the missing bits (works if *k* small).

Security and Functionality Analysis

- Can the attacker follow the same strategies for correcting de-synchronizations?
  - Attacker normally has access to groups of less than L consecutive bits.
  - No efficient method of solving quadratic systems over finite fields (MQ problem is NP-complete [Fraenkel, Yesha, 1979]).
  - Attacker's de-synchronizations are large – brute-forcing the bits does not work.

Security and Functionality Analysis

- Can the attacker synchronize encounters with tag to capture consecutive bits?
  - The period of the LFSR's output is $T = 2L - 1$;
  - For $L = 18,000$ (1 bit per second – throttled response – for 10 hours), all the bits that an attacker can gather in his lifetime belong to same LFSR period.
- Can an attacker verify tag's presence in a certain place (tag tracking) based on incomplete information about tag's secret?
  - This relies on verifying whether a system of (more than) $L + k$ quadratic equations with $L + k$ unknowns admits a solution – as hard as finding the solution.

Adopted Pet II Protocol (Algebraic Design)

How should we implement this, named as QTAB-KEP?

- We chose to implement it as a puzzle – one of the parties (the challenger/ authenticator) keeps transmitting clues that the other party (the prover/ supplicant) gathers to learn the first party's secret (the puzzle solution).

What do we want from the QTAB-KEP?

1. completely automatic,
2. independent of other security protocols,
3. should rely on a single session and be independent of the protocol's starting time (to prevent DoS),
4. should allow a ***customizable information transfer function*** versus the length of uninterrupted time spent listening to clues (to enable graceful degradation or time-based authorization policies),
5. robust to interference causing a few missed clues, or a few erroneous clues.

QTAB-KEP design

Design implications:

- 1 (completely automatic) and 2 (independent of other protocols) imply that the protocol has to be absolute-time and place independent.
- 3 (single session, independent of starting time) implies an evolving secret.
- 5 (robustness) implies some form of redundancy – error-correction coding. Note: this can also deal with (some) maliciously-injected clues.

Basic QTAB-KEP instantiation

Slightly-modified Shamir's secret sharing scheme:

- Take a finite field F = Z/pZ, with large prime p.
- Publish n fixed points $(a_1, a_2 \ldots a_n) \in F$.
- Choose $n - k - 1$ random shares $c_{i-n+1,i} = f_{1,i}$, $c_{i-n+2,i} = f_{2,1}$,..., $c_{i-k-1,i} = f_{n-k-1,i}$ from F.
- Set $a_0 = 0$ and $f_{0,i} = s_i$, and now, with access to $n - k$ fixed points, we compute the unique polynomial

  $f_i(z)$ of degree $n - k - 1$, that goes through all these points, i.e. $f_i(a_j) = f_{j,i}$ for $j = 0, 1, 2, \ldots, n-k-1$. This is done using Lagrange interpolation.
- The remaining k+1 shares are produced as follows: $c_{i-k+1,i} = f_i(a_{n-k})$,..., $c_{i,i} = f_i(a_n)$

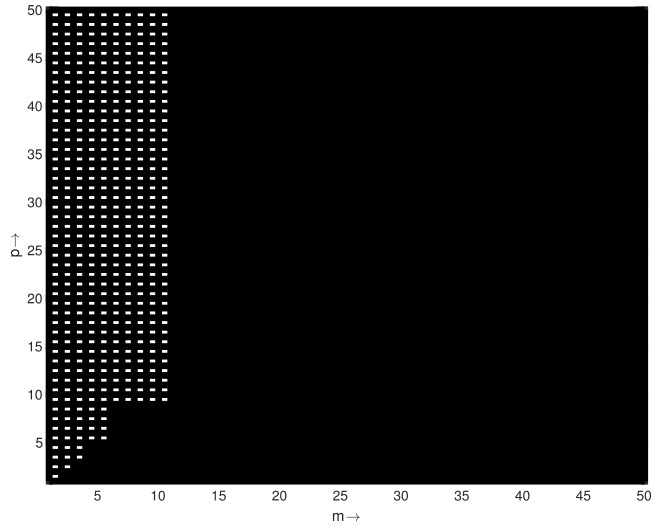
Basic QTAB-KEP instantiation with error correction

Slightly-modified Shamir's secret sharing scheme in Reed-Solomon (canonical) code form:

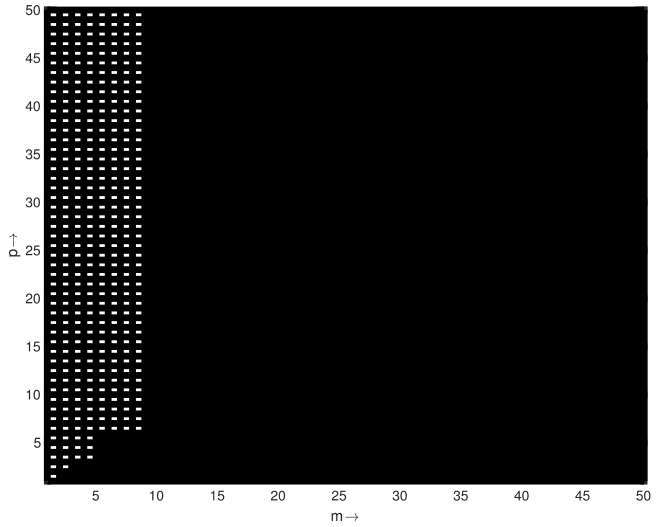
- Choose $n-k-1$ random coefficients $f_{1,i},...f_{n-k-1,i}$ from F.
- Pick and publish a random element $a_0 \in F$ and assign the secret to be $s_i = f_i(a_0)$.
- Compute $f_i(z)=f_{n-k-1,i}z^{n-k-1}+ ...+f_{1,i}z+f_{0,i}$, where $f_{0,i} = s_i-(f_{n-k-1,i}a_0^{n-k-1}+ ... +f_{1,i}a_0)$.
- Now $f_i(a_0)=s_i$, and any subset of $n-k-1$ coefficients of $f_i(z)$ leaks no information about s_i .
- Pick and publish a random primitive element b of the field F, and construct the generator polynomial $g(z) = (z - b)(z - b^2) \ldots (z - b^k)$ of degree k.
- The first $n - k$ shares of the secret are the coefficients of $f_i(z)$, and the following k shares are the coefficients of the remainder polynomial obtained by dividing $z^k f(z)$ by $g(z)$.

Parameter choice and feasible regions

Choose n and k according to the desired application. Then m and p may be anywhere in the feasible region. A straightforward choice: $m = n - k - 1$ and $p = k + 1$.



n=19, k=8



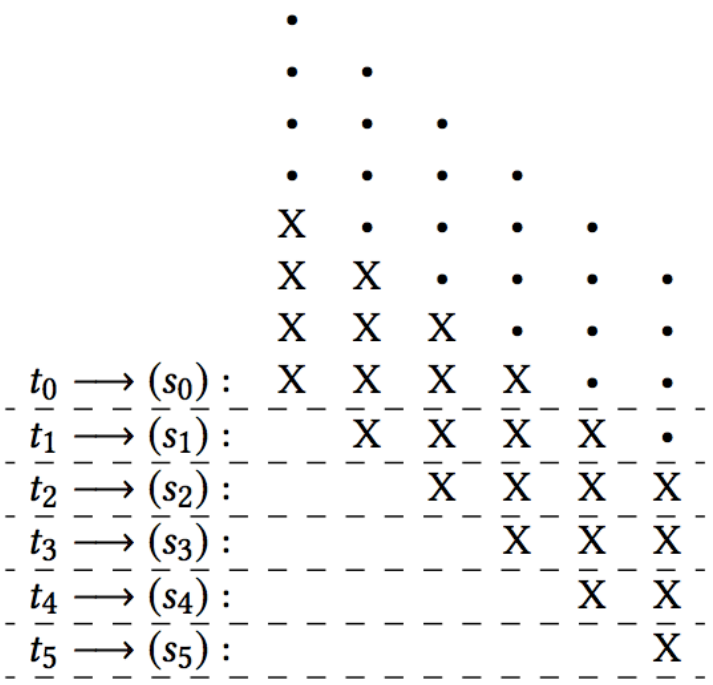
n=14, k=5

Achieving start-time independence

A multiplexed QTAB scheme.

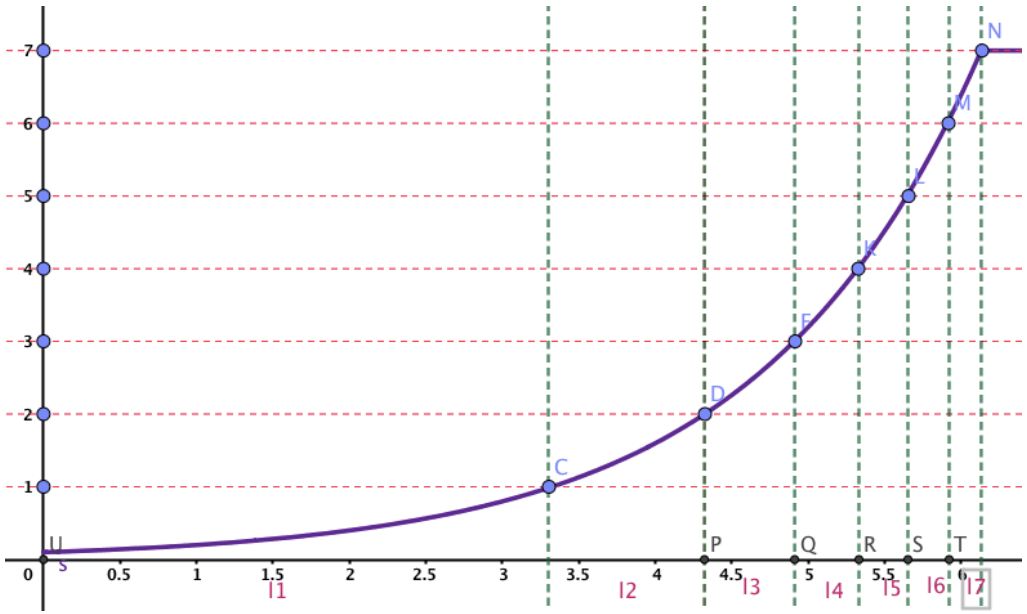
- Since first (n-k-1) clues are random, we can batch them.

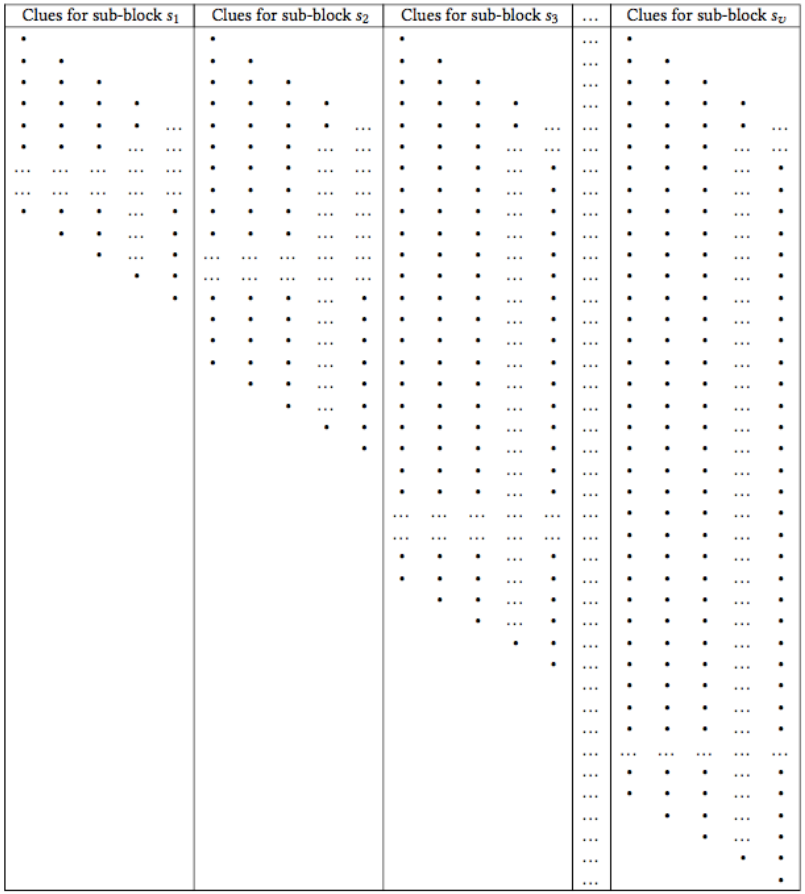
- The width of the multiplexed scheme is now (k+1) instead of n.



Controlling information transfer

The **extended** QTAB-KEP.





Security considerations of extended QTAB-KEP

Lemma 1.

Consider an extended QTAB-KEP consisting of v parallel basic QTAB-KEPs, indexed by $j \in \{1, 2, \ldots, v\}$.

Assume that the basic QTAB-KEP j is (n_j,k_j)-robust and (m_j,p_j)-secure, and that the parameters (n_j,k_j) were chosen such that $(n_1 - k_1) = z_1$, $(n_2 - k_2) = z_1 + z_2$,and soon, until $(n_v - k_v) = z_1 + z_2 + \ldots + z_v$, for some arbitrary positive integers $z_1,z_2,\ldots,z_v$.

Denote:

$n_{max} = \max\{n_1,n_2,\ldots,n_v\}$, $k_{min} = \min\{k_1,k_2,\ldots,k_v\}$, $p_{max} = \max\{p_1,p_2,\ldots,p_v\}$, $m_{min} = \min\{m_1,m_2,\ldots,m_v\}$.

Then the extended QTAB-KEP is (n_{max},k_{min})-robust and (m_{min},p_{max})- secure.

Security considerations of extended QTAB-KEP

Definition 3. δ-Robustness:

A QTAB-KEP is said to be (δ,n)- robust if a legitimate party who listens to the clue-issuer for the duration of at least n consecutive clues (out of which some clues may be received incorrectly), can recover the secret key with probability larger than $1 - \delta$.

Lemma 2.

Consider an extended QTAB-KEP consisting of v parallel basic QTAB-KEPs, indexed by $j \in \{1, 2, \ldots, v\}$.

Assume that the basic QTAB-KEP j is (δ,n_j)-robust and (m_j,p_j)-secure, and that the parameters (n_j,k_j) were chosen such that $(n_1-k_1)=z_1$, $(n_2 -k_2) = z_1 +z_2$, and so on, until $(n_v -k_v) = z_1+z_2 + \ldots + z_v$, for some arbitrary positive integers $z_1,z_2,\ldots,z_v$.

Denote: $n_{max} = \max\{n_1,n_2,\ldots,n_v\}$, $p_{max} = \max\{p_1,p_2,\ldots,p_v\}$, $m_{min} = \min\{m_1,m_2,\ldots,m_v\}$.

Then the extended QTAB-KEP is $(1 - (1 -\delta)^v, n_{max})$-robust and (m_{min}, p_{max})-secure.

seL4 Implementation (AP1 and AP2)

- Docker System
  - V. 19.03.2

```
ubuntu1804@ubuntu1804-VirtualBox:~$ docker version
Client: Docker Engine - Community
 Version:           19.03.2
 API version:       1.40
 Go version:        go1.12.8
 Git commit:        6a30dfc
 Built:             Thu Aug 29 05:29:11 2019
 OS/Arch:           linux/amd64
 Experimental:      false

Server: Docker Engine - Community
 Engine:
  Version:          19.03.2
  API version:      1.40 (minimum version 1.12)
  Go version:       go1.12.8
  Git commit:       6a30dfc
  Built:            Thu Aug 29 05:27:45 2019
  OS/Arch:          linux/amd64
  Experimental:     false
 containerd:
  Version:          1.2.6
  GitCommit:        894b81a4b802e4eb2a91d1ce216b8817763c29fb
 runc:
  Version:          1.0.0-rc8
  GitCommit:        425e105d5a03fabd737a126ad93d62a9eeede87f
 docker-init:
  Version:          0.18.0
  GitCommit:        fec3683
```

Build Dependencies

- Install Docker on local OS
- Get a running build environment from SEL4PROJ repository
- Map a particular directory into the container and create a bash alias to make it easier to restart

Setup Environment

- Create a bash alias "container"
  - echo $'alias container=\'make -C /<path>/<to>/seL4-CAmkES-L4v-dockerfiles user HOST_DIR=$(pwd)\" >> ~/.bashrc
- Load the seL4 environment by using the bash alias
- Create a directory for the project
- Initialize the build directory with an exercise. Enter the proper platform in --plat parameter
- Enter the build directory
- Build the exercise by executing the command **ninja**
- Once successfully executed, an image is created in the Image folder

seL4 Implementation (Cont.)

- Loading an Image
  - Two Possible ways
    - Using QEMU (Simulator for seL4)
      - Just type ./simulate
    - Other Platforms
      - Create the image with proper value in the parameter --plat
      - Load the created image in the SD card boot directory
      - Load the SD card in the platform and connect it to the computer with USB to TTL cable
  - seL4 Version: 9.0.1

Protocol Initialization

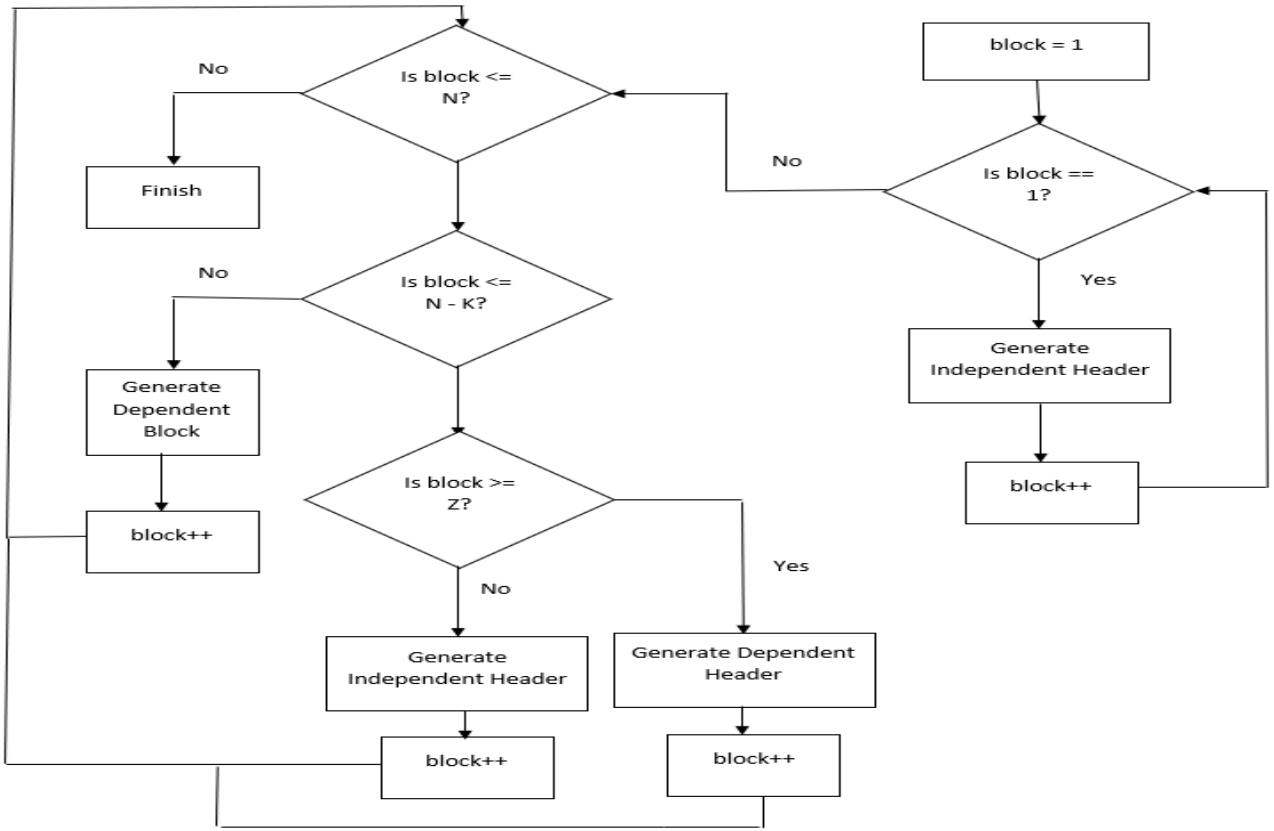
- Alice is sending the clue
- Bob is receiving the clue
- N = Number of total blocks
- K = Number of dependent blocks
- n = Number of total clues per block
- k = Number of dependent clues per block
- Z – 1 = Number of blocks an attacker can listen to consecutively

Alice

Generate Independent Header ≈ Generate Clue → Compute Tail → Randomly Assign Header

Generate Dependent Header ≈ Ger C

Generate Dependent Block ≈ Gath for ((N –

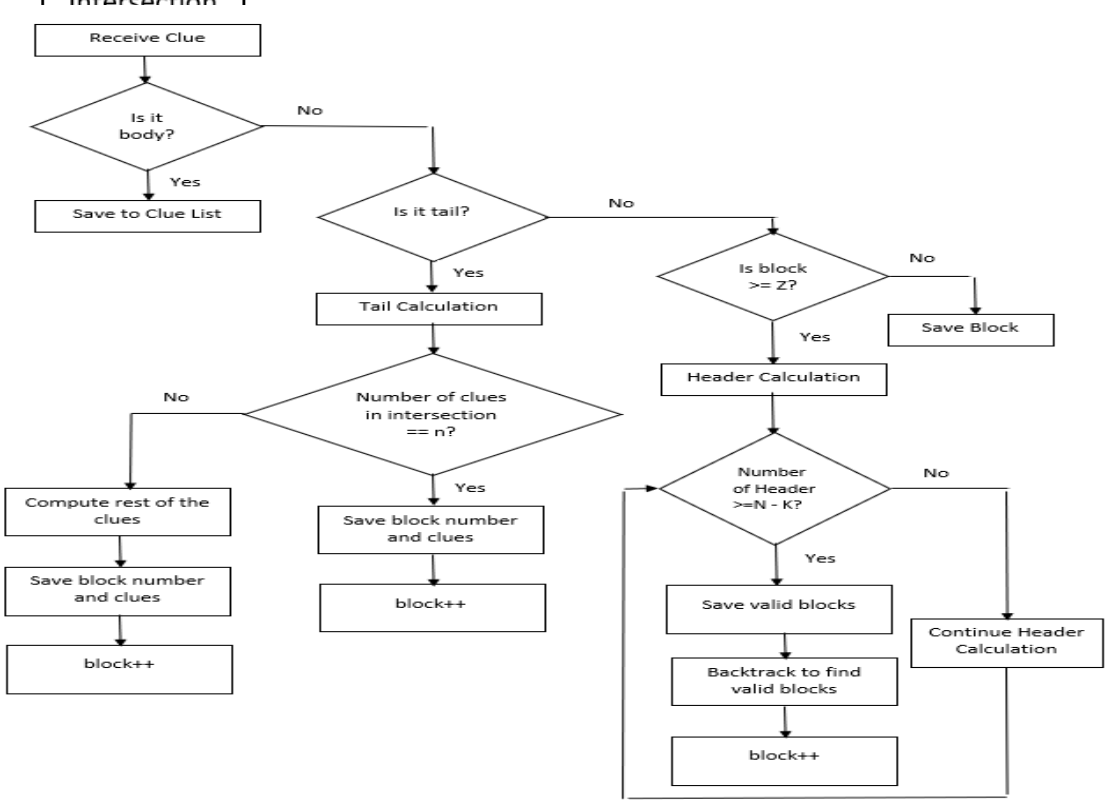


Bob

Tail Calculation ≈ Compute Hash for each clue → Find Intersection

Header Calculation ≈ Compute Header → Find Intersection

Continue Header Calculation ≈ Gather another 7 blocks



Simulation on Linux Platform

- Alice

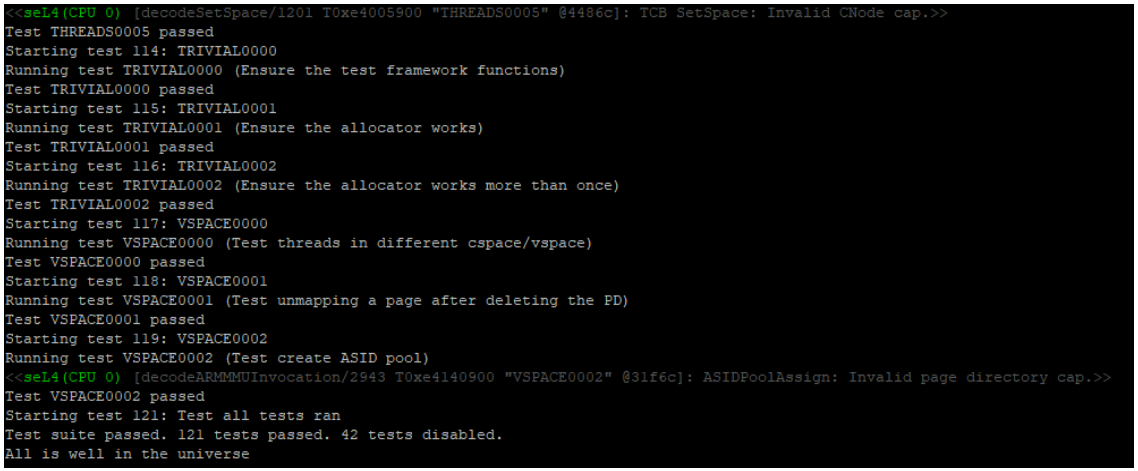
```
ELF-loading userland images from boot modules:
size=0x32f000 v_entry=0x408dea v_start=0x400000 v_end=0x72f000 p_start=0xc0d000 p_end=0xf3c000
Moving loaded userland images to final location: from=0xc0d000 to=0xa16000 size=0x32f000
Starting node #0 with APIC ID 0
Mapping kernel window is done
Booting all finished, dropped to user space
Sender process ID is: 3
The secret is: 1f653785cc44d75b0f8284452628c7393584e3f0fc6980c2bf90bddcdb59f329
Sending Clue: 204
Sending Clue: 178
Sending Clue: 192
Sending Clue: 94
Sending Clue: 1512
Sending Clue: 156
Sending Clue: 119
Sending Clue: 181
Sending Clue: 109
Sending Clue: 1373
Sending Clue: 99
Sending Clue: 29
Sending Clue: 80
Sending Clue: 76
Sending Clue: 701
Sending Clue: 90
Sending Clue: 169
Sending Clue: 74
Sending Clue: 140
Sending Clue: 1210
All clues have been sent!
```

- Bob

```
ELF-loading userland images from boot modules:
size=0x331000 v_entry=0x408dea v_start=0x400000 v_end=0x731000 p_start=0xc0e000 p_end=0xf3f000
Moving loaded userland images to final location: from=0xc0e000 to=0xa16000 size=0x331000
Starting node #0 with APIC ID 0
Mapping kernel window is done
Booting all finished, dropped to user space
Receiver process ID : 4
Recovered Clue: 204
Recovered Clue: 178
Recovered Clue: 192
Recovered Clue: 94
Recovered Clue: 1512
Recovered Clue: 156
Recovered Clue: 119
Recovered Clue: 181
Recovered Clue: 109
Recovered Clue: 1373
Recovered Clue: 99
Recovered Clue: 29
Recovered Clue: 80
Recovered Clue: 76
Recovered Clue: 701
Recovered Clue: 90
Recovered Clue: 169
Recovered Clue: 74
Recovered Clue: 140
Recovered Clue: 1210
Reconstructed Secret is: 1f653785cc44d75b0f8284452628c7393584e3f0fc6980c2bf90bddcdb59f329
```

Implementation on Raspberry Pi

- Which Raspberry Pi
  - Raspberry Pi 3B+
- What we did
  - Load the seL4 test image to RPi3B+ using SD card
  - The test output looks fine



Implementation on Raspberry Pi

- Modified the code to include rpi3

```
# Define some meta options
set(valid_arm_platform "am335x;sabre;kzm;exynos5410;exynos5422;tx1;tx2;zynq7000;rpi3")
set(valid_x86_platform "ia32;x86_64")
set(valid_riscv_platform "spike")
set(valid_platforms "${valid_x86_platform};${valid_arm_platform};${valid_riscv_platform}")
set_property(CACHE PLATFORM PROPERTY STRINGS "${valid_platforms}")
list(FIND valid_platforms "${PLATFORM}" index)
if("${index}" STREQUAL "-1")
    message(FATAL_ERROR "Invalid PLATFORM selected: \"${PLATFORM}\"
Valid platforms are: \"${valid_platforms}\"")
endif()
```

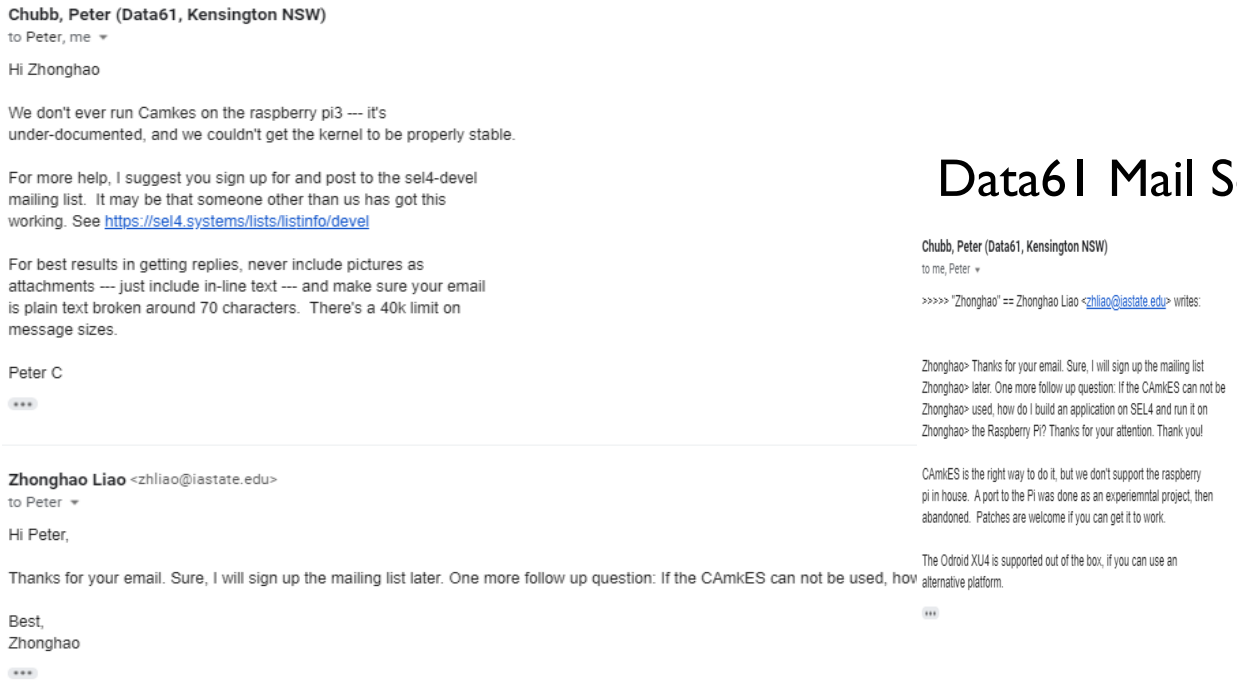
- Now image can be built but can't be simulated
- Even Rpi doesn't recognize the image

```
U-Boot> fatload mmc 0 0x10000000 capdl-loader-image-arm-bcm2837
** Reading file would overwrite reserved memory **
U-Boot> fatload mmc 0 0x00100000 capdl-loader-image-arm-bcm2837
2110528 bytes read in 94 ms (21.4 MiB/s)
U-Boot> bootelf 0x00100000
## No elf image at address 0x00100000
U-Boot>
```

Implementation on Raspberry Pi

- Issues we faced
  - How to build the seL4 based project for RPi and make the image?
  - How to simulate RPi image using qemu?
  - How to build a project on seL4 that can communicate between two RPis?
  - How to get the benchmarking tool work? Is there any sample code or tutorial available? The existing tutorial page is out of date because the seL4 has been moved from Kconfig build system to a CMake-based build system. If

Data61 Mail Screenshot



Chubb, Peter (Data61, Kensington NSW)
to Peter, me

Hi Zhonghao

We don't ever run Camkes on the raspberry pi3 --- it's under-documented, and we couldn't get the kernel to be properly stable.

For more help, I suggest you sign up for and post to the sel4-devel mailing list. It may be that someone other than us has got this working. See https://sel4.systems/lists/listinfo/devel

For best results in getting replies, never include pictures as attachments --- just include in-line text --- and make sure your email is plain text broken around 70 characters. There's a 40k limit on message sizes.

Peter C

Zhonghao Liao <zhliao@iastate.edu>
to Peter

Hi Peter,

Thanks for your email. Sure, I will sign up the mailing list later. One more follow up question: If the CAmkES can not be used, how

Best,
Zhonghao

Chubb, Peter (Data61, Kensington NSW)
to me, Peter

>>>>> "Zhonghao" == Zhonghao Liao <zhliao@iastate.edu> writes:

Zhonghao> Thanks for your email. Sure, I will sign up the mailing list
Zhonghao> later. One more follow up question: If the CAmkES can not be
Zhonghao> used, how do I build an application on SEL4 and run it on
Zhonghao> the Raspberry Pi? Thanks for your attention. Thank you!

CAmkES is the right way to do it, but we don't support the raspberry pi in house. A port to the Pi was done as an experiemntal project, then abandoned. Patches are welcome if you can get it to work.

The Odroid XU4 is supported out of the box, if you can use an alternative platform.

Summary and Future Works

- We proposed the novel Quality-time-as-an-advantage AP pairing protocols, which:
  - require no human interaction;
  - transient;
  - tolerate interference and de-synchronizations;
  - demands limited resources.

Work with seL4:

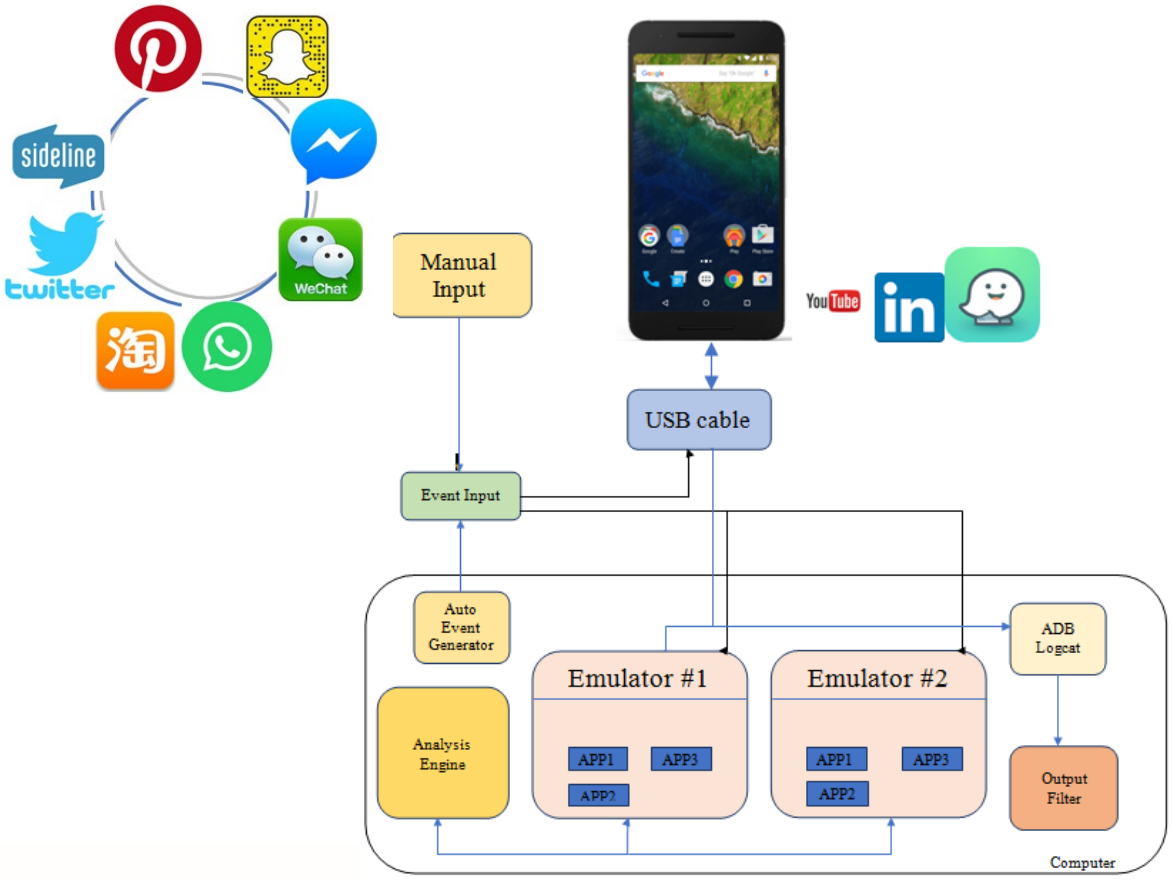
1. Implementation on seL4 helps further the understanding of OS security.
2. Develop course module and project assignments around seL4: Adding new driver, network stack, and apps.
3. Explore other IoT OS platforms, e.g., QNX, Fuchsia, Zephyr, …

Thanks

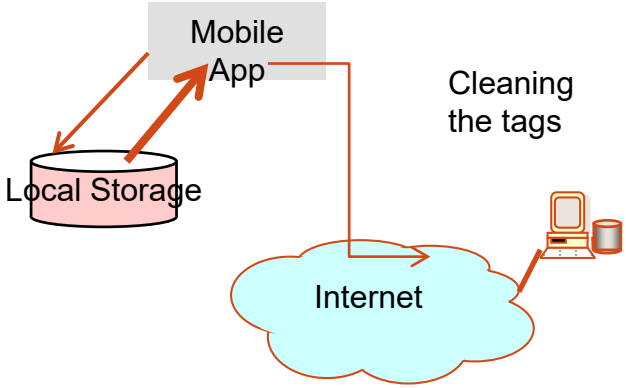
Q&A

Yong Guan
guan@iastate.edu
Iowa State University

Mobile App Forensic Analysis

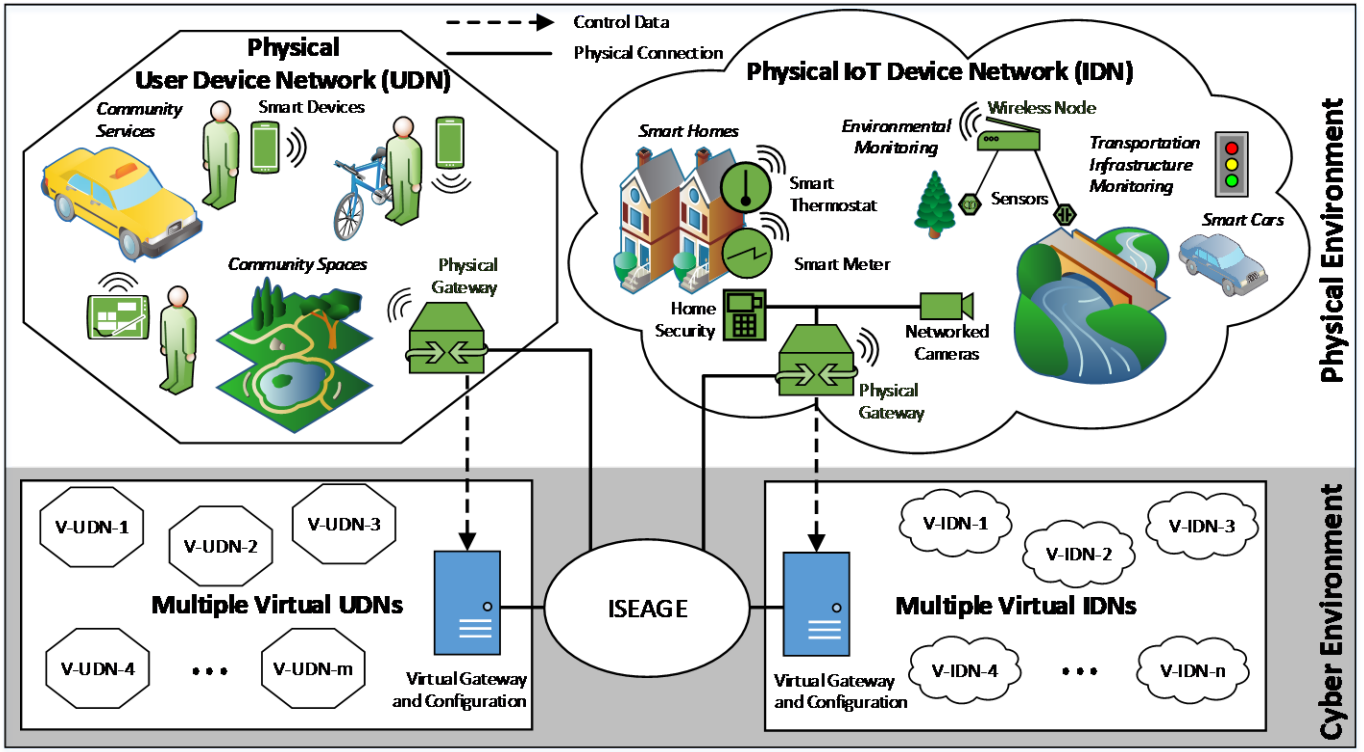


- Native Code Analysis
  - Obfuscator - 加固
- Third-party Library
- Implicit Flow, Indirection
- Inter-component Communication
- Reflection
- Dynamic loaded Library



csafe
Center for Statistics and Applications in Forensic Evidence

IoTE^3 – Cyber-Physical Integration Security



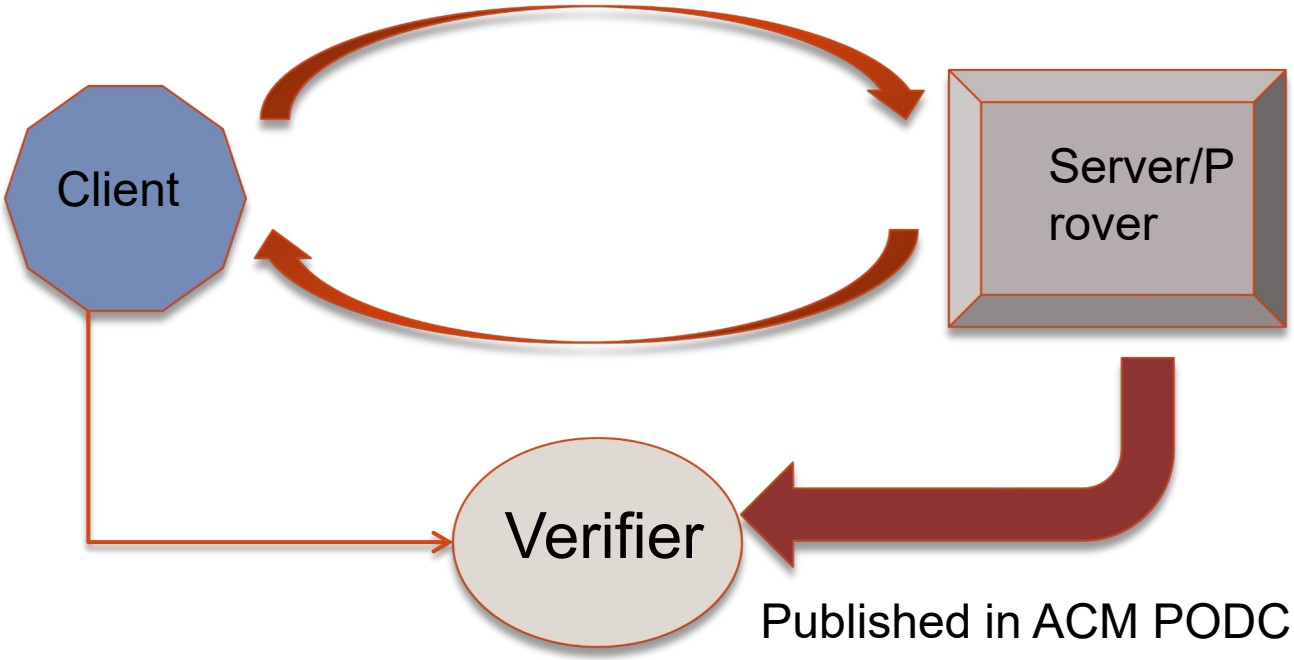
- Physical environment: physical IoT networks, smart factories, etc.
- Cyber environment: virtual IoT networks – driven by traces and traffic collected from the physical networks.
- Cyber-physical integration: multiple physical and virtual IoT networks are connected with each other via ISEAGE as the communication backbone.

Funded by NSF

Verifiable Computation via Independent Verification Outsourcing

Forensic auditing via independent 3rd party verification

Crypto Protocol Designs



Published in ACM PODC, ESORICS, AsiaCCS, CNS