Leveraging Blockchain-based protocols in IoT systems

Konstantinos Kolias, Angelos Stavrou
Computer Science Department
George Mason University

Irena Bojanova, Jeff Voas, Tim Grance
Information Technology Laboratory
National Institute of Standards & Technology
Talk Outline

• IoT Scale & Scope
• Crypto Failures in IoT: A Motivating Use Case
• Understanding the IoT Crypto Needs
• Short Blockchain Primer
• Blockchain in IoT: What are the potential use cases?
• Why direct use of Blockchain is not practical for IoT
• Challenge: Design practical Blockchain-based protocols for IoT
Internet of Things Defined

- Kevin Ashton introduced the term Internet of Things (IoT) in 1999
- Network of devices able to configure themselves automatically
- Human is not the center of the system
- **Motivation**: Better understanding of the environment and response to certain events. Machines are doing better in sensing & reporting on conditions
- **Fact**: Applications of traditional Internet are different than the applications of IoT
What the Future Holds

Drivables

Flyables

Scannables

Wearables
The Growth of IoT

THE INTERNET OF THINGS
AN EXPLOSION OF CONNECTED POSSIBILTY

BILLS OF DEVICES

YEAR

2002
1,000,000
Introduction of the first device.

2009
IoT INCEPTION
CIOs and enterprise IT leaders began to realize the potential of IoT.

2010
10.2 BILLION
Shipments of mobile devices surpassed PC shipments for the first time.

2013
11.2 BILLION
A new smartphone is activated every 10 seconds.

2015
14.4 BILLION
First large-scale deployment of connected devices in the smart home.

2016
22.9 BILLION
Several new industries emerged and connected devices never experienced seasonal dips.

2018
34.8 BILLION
The US is at its highest level of connected devices ever.

2019
42.1 BILLION
The internet of things is no longer a novelty among the connected devices.

2020
50.1 BILLION
The internet of things is now an accepted feature on every device.

Each year, the number of connected devices continues to grow at an unprecedented rate.
Sectors of IoT Applications

- **Smart Home**
  - Home automation
  - Energy efficiency
  - Home security

- **Transportation**
  - Road safety
  - Traffic regulation
  - Law enforcement

- **Retail**
  - Automatic payments
  - Efficient cataloguing
  - Shipment tracking

- **Industry**
  - Quality assurance
  - Failure prediction
  - Productivity improvement

- **Healthcare**
  - Condition monitoring
  - Remote treatment
  - Personalized advices
Sensors & Actuators
Connectivity

- **WAN**: 2G, 3G, 4G
- **LAN**: Wi-Fi, 5G
- **PAN**: LoWPAN, Z-WAVE, ANT, Bluetooth

IPv6
Common Security Incidents

- Private Data Collection: 90%
- Insecure Interfaces: 60%
- Unencrypted Communications: 70%
- Weak Requirements: 80%
Top 10 Vulnerabilities (OWASP)

- **Insecure Web Interfaces**
  - Default accounts, XSS, SQL injection

- **Inefficient Authentication/Authorization**
  - Weak passwords, no two-factor authentication

- **Insecure Network Services**
  - Ports open, use of UPnP, DoS attacks

- **Lack of Transport Encryption**
  - No use of TLS, misconfigured TLS, custom encryption

- **Private Data**
  - Unnecessary private information collected

- **Insecure Cloud Interfaces**
  - Default accounts, no lockout

- **Inefficient Mobile Interfaces**
  - Weak passwords, no two-factor authentication

- **Insufficient Security Configurability**
  - Ports open, use of UPnP, DoS attacks

- **Insecure Software/Firmware**
  - Old device firmware, unprotected device updates

- **Poor Physical Security**
  - Exposed USB ports, administrative accounts
Use Case: Home Automation

Typical Use Case:

• Sensors and other devices connected to a Home Network
• Devices communicate directly to an “aggregator” gateway
What Can Go Wrong?

- Attacker introduces a soft-AP or Sensors with the same characteristics
- Custom Crypto
  - Because we can do faster
- No Authentication
- No Encryption
Why Can Go Wrong?

• **Badly Designed System**
  - Platform that cannot handle encryption (SSL/TLS)
  - Cannot communicate securely with standard servers

• **Badly Implemented Crypto**
  - **Example**: Implement “Custom” TLS for “faster” operation
  - **Challenge**: Make TLS lighter but maintain compatibility
  - **Method**: Remove the “heaviest” operations
    - First contender: verification of server certificate
  - **Result**: Minimalistic hardware can support TLS
  - **Gain**: Use of even cheaper hardware
    - Caveat: possible security holes
How Can Go Wrong?

Protocol Hacked at DefCon 2015
• Connects to google calendar to show notes on screen
• Supports SSL/TLS but does not validate server certificates
• Unleash MiM attack
• Steal User’s Credentials

Custom Crypto Implementation not a solution
Why Use Custom Crypto?

RSA 1024 Runtime Overhead:

<table>
<thead>
<tr>
<th>Device</th>
<th>Clock Speed</th>
<th>Implementation Type</th>
<th>RSA 1024 ms*</th>
<th>CPU Overhead</th>
<th>Notes</th>
</tr>
</thead>
<tbody>
<tr>
<td>Arduino UNO</td>
<td>16Mhz AVR</td>
<td></td>
<td>12596 ms*</td>
<td>8504 ms#</td>
<td></td>
</tr>
<tr>
<td>Arduino Leonardo</td>
<td>16Mhz AVR</td>
<td></td>
<td>12682 ms*</td>
<td>8563 ms#</td>
<td></td>
</tr>
<tr>
<td>Arduino Mega</td>
<td>16Mhz AVR</td>
<td></td>
<td>12596 ms*</td>
<td>8504 ms#</td>
<td></td>
</tr>
<tr>
<td>Arduino Due</td>
<td>84Mhz ARM</td>
<td></td>
<td>1032 ms*</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Arduino Yún</td>
<td>16Mhz AVR + 400Mhz MIPS</td>
<td></td>
<td>707 ms*</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Intel Galileo</td>
<td>400Mhz x86</td>
<td></td>
<td>192 ms*</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

* these numbers are based on a 100% C implementation
# these numbers are based on mixed C/AVR assembly implementation

Some of the traditional Crypto is too “expensive” for embedded devices.

https://evothings.com/is-it-possible-to-secure-micro-controllers-used-within-iot/
# Survey of Crypto Support in IoT

<table>
<thead>
<tr>
<th>Brand</th>
<th>Name</th>
<th>CPU</th>
<th>Frequency</th>
<th>Sram</th>
<th>Flash</th>
<th>Crypto Acceleration</th>
<th>Energy</th>
<th>Public Key Encryption</th>
</tr>
</thead>
<tbody>
<tr>
<td>Belkin</td>
<td>WeMo Switch</td>
<td>Ralink RT5350F (MIPS)</td>
<td>360 MHz</td>
<td>32MiB</td>
<td>16MiB</td>
<td>No</td>
<td>Wall socket</td>
<td>Yes</td>
</tr>
<tr>
<td>Samsung</td>
<td>Smarthings Hub</td>
<td>PIC32MX695F-512H 32 Bit</td>
<td>80Mhz</td>
<td>128KB</td>
<td>512K</td>
<td>No</td>
<td>Wall socket/Battery</td>
<td>Yes</td>
</tr>
<tr>
<td>Nest</td>
<td>Thermostat</td>
<td>Texas Instruments AM3703CUS Sitara (ARM Cortex A8 )</td>
<td>1Ghz</td>
<td>512Mb</td>
<td>2Gb</td>
<td>Yes</td>
<td>Wall socket</td>
<td>Yes</td>
</tr>
<tr>
<td>LIFX</td>
<td>Color 1000</td>
<td>Kinetis K22 (ARM Cortex-M4)</td>
<td>120Mhz</td>
<td>128KB</td>
<td>512K</td>
<td>No</td>
<td>Wall socket</td>
<td>No</td>
</tr>
<tr>
<td>Amazon</td>
<td>Echo</td>
<td>Texas Instruments DM3725CUS100 (ARM Cortex A8 )</td>
<td>1Ghz</td>
<td>256MB</td>
<td>4GB</td>
<td>Yes</td>
<td>Wall socket</td>
<td>Yes</td>
</tr>
<tr>
<td>Philips</td>
<td>Hue Lights</td>
<td>ST Microelectronics STM32F217VE (ARM Cortex-M3)</td>
<td>120Mhz</td>
<td>128KB</td>
<td>1MB</td>
<td>Yes</td>
<td>Wall socket</td>
<td>Yes</td>
</tr>
<tr>
<td>Philips</td>
<td>Hue Lights (Bulb)</td>
<td>STM32F100RBT6B (ARM Cortex-M3)</td>
<td>24Mhz</td>
<td>8KB</td>
<td>128KB</td>
<td>No</td>
<td>Wall socket</td>
<td>No</td>
</tr>
<tr>
<td>Nest</td>
<td>Smoke/Carbon Alarm</td>
<td>Freescale SCK60DN512VL10 custom Kinetis K60 (ARM Cortex M4) + Freescale SCKL162128V (ARM Cortex M0)</td>
<td>100 Mhz+48Mhz</td>
<td>128KB</td>
<td>512K</td>
<td>Yes</td>
<td>Wall socket/Battery</td>
<td>Yes</td>
</tr>
<tr>
<td>Pebble</td>
<td>Time</td>
<td>ST Micro STM32F439ZG (ARM Cortex M4)</td>
<td>180 Mhz</td>
<td>256KB</td>
<td>2MB</td>
<td>Yes</td>
<td>Battery</td>
<td>No</td>
</tr>
<tr>
<td>Fitbit</td>
<td>Surge</td>
<td>Silicon Labs EFM32 Giant Gecko (ARM Cortex-M3) EFM32GG395F1024</td>
<td>48Mhz</td>
<td>128KB</td>
<td>1MB</td>
<td>Yes</td>
<td>Battery</td>
<td>No</td>
</tr>
<tr>
<td>Fitbit</td>
<td>One</td>
<td>STMicroelectronics 32L151C6 Ultra Low Power (ARM Cortex M3)</td>
<td>32Mhz</td>
<td>16KB</td>
<td>128KB</td>
<td>No</td>
<td>Battery</td>
<td>No</td>
</tr>
</tbody>
</table>
What do we really need?

• IoT System Operational Requirements (Empirical)
  • Dynamic but verifiable group membership
  • Authentication & Data integrity
  • Secure against single-node (or small sub-set of nodes) key leakage
  • Lightweight operations in terms of resources
  • Encryption is a plus but not firm requirement
  • Capable of handling sensor “sleep/power-off” periods
  • Handle resource diversity and data of sensors and aggregators
Potential Solutions

• Lightweight Cryptography
  • Security/Cost/Performance trade-off a challenge
  • Size of key material cannot be lowered
  • New crypto designs are promising but not standardized or adopted

• Can we use existing crypto blocks to meet the requirements?
  • Microcontroller devices support cryptographic hashing
  • Support for AES 256 is pervasive but still expensive
  • Can we leverage Blockchain-based protocols?
Blockchain Primer

Public Distributed Verifiable Cryptographic Ledger

- Public
  - All participants gain access to “read”
- Distributed
  - Peer-to-Peer Data Communication, Fully Decentralized
- Cryptographic
  - Digitally signed transactions, proof-of-work limits rate of input
- Ledger
  - Verifiable Transactional Database
Blockchain Primer

[Diagram showing the process of signing and verifying transactions in a blockchain, with key components such as public and private keys, hashes, and signatures.]
Blockchain Primer

Blockchain Blocks

- Sequences of signed and verified transactions
- Published and distributed globally
- Magic number, Size
- Header
  - Hash of previous block (chain)
  - Merkle root hash of block
  - Timestamp
  - Target, nonce (mining)
- Number and list of transactions
Blockchain Primer

Longest Proof-of-Work Chain

Block Header
- Prev Hash
- Nonce
- Merkle Root

Block Header
- Prev Hash
- Nonce
- Merkle Root

Block Header
- Prev Hash
- Nonce
- Merkle Root

Hash01
Hash23

Merkle Branch for Tx3

Hash2
Hash3

Tx3

Bitcoin: A Peer-to-Peer Electronic Cash System, Satoshi Nakamoto
Is Blockchain Directly Applicable in IoT?

Desirable Properties

- Distributed protocol with verifiable transaction history
- Dynamic membership multi-party signatures

Undesirable Properties

- Requires proof of “work”
- Requires PKI
- Size of the Ledger an issue for “small” devices
- Anonymous (unverifiable) Join/Leave operations
What can we do?

Eliminate undesirable properties

• Requires proof of “work”
  Requires proof of earlier participation using history

• Requires PKI
  Hash-based signatures (or other Merkle-tree schemes)

• Size of the Ledger an issue for “small” devices
  Prune and Compress Ledger. Maintain only device-relevant transaction ledger when device is too resource constrained

• Anonymous (unverifiable) Join/Leave operations
  Group signatures using pre-shared group Key(s)
Hash-Chain

One-time hash passwords (Lamport 1981):

- Client generates iteratively a list of hash values (in reverse order of index).

\[
\begin{align*}
z_\ell & \leftarrow \{0, 1\}^n \\
z_i & \leftarrow h(z_{i+1}) \quad \text{for } i \in \{\ell - 1, \ell - 2, \ldots, 0\}
\end{align*}
\]

- \(z_0 = h(z_1) = h(h(z_2)) = \ldots\) is the “public key”
- Keys are revealed in opposite order, starting from \(z_1\)
- Verification of \(z_i\): starting from \(z_i\) verify, if \(z_0\) is indeed \(i\)-th hash
- Keys can be used only once!
Hash-Chain: PreImage Path

Lamport's one-time-password scheme has either

- $O(\ell)$ storage (whole chain retained) or
- $O(\ell)$ preimage generation time (only $z_\ell$ retained).

Both extremes are not exactly efficient.

Naive optimization: mark few elements with "pebbles", retain values and use as starting points. If $N$ pebbles are evenly distributed then the worst case is $O(\ell/N)$ hash calculations per key.

Jakobsson (2002): traversal algorithm which amortizes $h()$ calculations. $O(\log \ell)$ memory and $O(\log \ell)$ hashing steps to output a key (preimage).

Pebbles are placed at positions $2^j, j = 1..[\log \ell]$; preimages are extracted from left. If a pebble is reached it jumps next to another, and leftover calculations at each step are used to move it gradually into position between neighbors.
Typical IoT Aggregation Networks

Aggregator

Sensor
Sensor
Sensor

Aggregator

Sensor
Sensor

Aggregator

Sensor
Sensor
Sensor
Sensor
Sensor
Sensor
Blockchain-based Protocol for IoT?

We suggest a Blockchain-based protocol that uses the following blocks:

\[ x_{top} = h(x_{12}|x_{34}) \]

\[ x_{12} = h(x_1|x_2) \]
\[ x_{34} = h(x_3|x_4) \]

\[ x_i = H(Data \ || \ K_G \ || \ H(z_i)^n), H(z_i)^{n-1} \]

*H = Hash, K_G = group Key, z_i = sensor i "public key"*
Blockchain-based Protocol for IoT?

We suggest a Blockchain-based protocol that uses the following blocks:
Does the Scheme Meet the Requirements?

- **IoT System Operational Requirements (Empirical)**
  - Dynamic but verifiable group membership
  - Secure against single-node (or small sub-set of nodes) key leakage
    - Only Aggregators can add nodes by issuing a group Key
    - Can be done using Symmetric Encryption or a Hash Chain
    - Node is verified both by group key AND by participation history
  - To add a node, an adversary will have to:
    a) Compromise the group key
    b) Issue an “add node” transaction
    c) Add a sensor node
  - Shape of the tree shows “additions” and “removals” of nodes over time
Does the Scheme Meet the Requirements?

• IoT System Operational Requirements (Empirical)
  • Authentication & Transaction integrity
    • Nodes and transactions are authenticated using the group key and the node Lamport signatures
    • A node uses his Lamport public key to validate inserted DATA, transmits DATA to aggregator(s)
  • Lightweight operations in terms of resources
    • Operations can be lightweight for sensors. Aggregators have more resources
  • Encryption is a plus but not firm requirement
    • No need for encryption
Does the Scheme Meet the Requirements?

- IoT System Operational Requirements (Empirical)
  - Capable of handling sensor “sleep/power-off” periods
    - Nodes can re-authenticate using their knowledge of historical transactions proving their membership specific historical transactions using predecessors for Lamport Signatures

\[ x_i = H(Data \| K_G \| H(z_i)^n), k, H(z_i)^{n-k} \]

*where n – k is smaller than the last signature from i*

- Handle resource diversity and data of sensors and aggregators
  - Different nodes store different portions of the ledger
  - Aggregators fully, others partial
Conclusions

• IoT Scale, Vendors, Technologies increase exponentially
• IoT Devices will always have diverse capabilities & Resources
• Use of Cryptography is done without clear understanding of the implications
• No Current Standards for Lightweight cryptography

• Blockchain inspired protocols combined with new cryptographic primitives might be the path forward
Questions?