Low-Energy Authentication with Selective Privacy for Heterogeneous IoT Devices in Smart-Farms

Steph Rudd, Hamish Cunningham
The University of Sheffield
Sheffield, United Kingdom
(lia08sr, h.cunningham) @sheffield.ac.uk

***Abstract*—This paper presents a payload-based security model for authenticating constrained and heterogeneous constrained devices in a smart-farm application. The domain ideally operates with optional internet and no constant power supply, using battery-powered microcontrollers for the monitoring and control of aquaponics systems. The motivation was to reduce energy consumption for extended battery life, whilst enabling a robust, decentralised security design. Results demonstrated an annual saving of up to 96% device battery life when compared to recommended TLS implementation, whilst maintaining equally high security standards.**

I. Introduction

The motivations for smart-farms are manifold, including a lower carbon footprint than traditional farming, autonomy, and environmental control. Smart-farming sees benefits of yield and non-indigenous crop varieties with the use of recycled resources without manual labour. Specifically in the domain of aquaponics [1]–[4], natural fertilisers are cycled around a flood-and-drain-system, demonstrating a self-sustaining crop growth environment from the symbiotic relationship between plants and fish. Smart-farming offers many applications to increase productivity in settings such as domestic [5], industrial [6], and with livestock [7]. However, the successes of smart-farming rely heavily on Internet of Things (IoT) infrastructure. For sensors and actuators to operate correctly and continuously, downtime and interruptions have to be as close as possible to zero, and this is where security takes the stage. Security infrastructure can be draining on energy resources and leave networks vulnerable to a single point of failure, but security in general ensures reliable communications between pairs of devices. Now, established security practice has revolved around Transport Layer Security (TLS), utilising a negotiation process known as a 'handshake' to create a client and server relationship. TLS is strong and universally accessible to both constrained and traditional devices and their protocols - but the energy consumption and centralisation are problems. This research is focused on how to address the issues surrounding TLS and PKI [8]. Such issues include reducing the high energy consumption of TLS client and server authentication, and removing the central vulnerabilities of the model, whilst providing security for any communications protocol the network will use. The aim of this research is to remove the barriers of integrating security into constrained networks used for smart-farming, the aims of which are:

1) Privacy; protecting private data whilst allowing selectivity of privacy.
2) Certificates; removing X.509 certificates without detriment to security.
3) Session timeout; extending the connections between entities to reduce energy use.
4) Individual protocol security; remove protocol-specific security infrastructure.
5) Energy consumption; reducing time and power used in authentication.

Aquaponics is used as the smart-farming domain by which to test our security application, exemplifying the benefits of the design to a Wireless Sensor Network (WSN). The aquaponics network is formed of a Raspberry Pi microcomputer acting as a central data server (or, in IoT terms, a gateway) between a series of battery-powered ESP32 microcontrollers. Each ESP32, or edge device, operates a system composed of a fish tank and multiple plant beds, using sensors and actuators for measuring and impacting temperature, humidity, flood and drain, light, and pH level. This symbiotic relationship between plants and fish represents distribution of a natural fertiliser from the fish to nourish plant growth, and a smart-farm which can largely operate autonomously and remotely. The protocols used will include Bluetooth Low Energy (BLE), Long-Range Wireless Area Network (LoRaWAN), and HyperText Transfer Protocol (HTTP), or WiFi. Wherever possible, BLE or LoRaWAN should be used, as WiFi will quickly exhaust a battery-powered microcontroller. This paper presents a study into the energy consumption of authentication models for the ESP32:

1) Analysis of the existing authentication processes available to the ESP32.
2) Design of a decentralised authentication process with structural differences to existing practice.
3) Testing of the differences between energy consumptions of the two models.

II. Problem Background

The background problem is presented in three parts; expectations, centralisation, and application appropriateness.

A. Expectations

The biggest problem of securing constrained devices is the unrealistic expectations of what a microcontroller should be

ISSN 2305-7254

able to process, compared with a perception of adequate security. Now, since TLS traditionally secures machines supplied with constant power, high-bandwidth telecoms and excellent processing capabilities, it has developed in parallel with those provisions. However, it is more appropriate to compare microcontroller capabilities with machines typical of the 90's, and their applications to suit. Constrained devices are equipped with the protocols and power supplies to support the operations of a smart-farm environment - the ESP32 provides suitably small packet sizes and BLE to communicate them. Therefore security considerations should not limit choices between a power and processing trade-off, but how TLS can be adapted to function effectively within an environment reflecting 'obsolete technologies' [9].

B. Centralisation

TLS employs X.509 certificates to bind keys with identities. The management of these certificates in undertaken by a Public Key Infrastructure (PKI), typically consisting of five parts; Certificate Authority (CA), to store and issue X.509 certificates, Registration Authority (RA), to verify entity identity, a central directory to securely store and index keys, a management system for access and distribution, and a certificate policy to state procedures and rules. This is a cumbersome and delicate series of operations criticised for codependency and vulnerabilities due to the centralised data stores [10]. PKI is also storage-intensive, populating lists for both current and revoked data, causing a scalability issue for IoT. The CA of every website is hosted on its server, available to view as current and sufficient by site visitors. Revoked certificates have displayed as current in some browsers, and have been identified as a security flaw [11] - further disproving the value of the X.509 system. Key escrow for certificates also undermines privacy [12], and has been described as placing "Keys under doormats" [13]. Official bodies are granted access to TLS communications for law enforcement using PKI facilities [14], with escrow hosted on servers and retaining key ownership. Although it is possible to operate TLS without certificates [15], there is no alternative fulfillment for verifying the identity of the server by which you, as the client, are contacting - and therefore without employing TLS there is no assurance of security. In summary, even with the heavy implementation and energy consumptions of encryption for privacy, there are still privacy issues present. In the case of IoT applications where the majority of data readings are benign, such a false perception of privacy only serves to drain battery life - in return for a questionable infrastructure.

C. Application-appropriate security

Applying TLS to any protocol comes with the same caveats as traditional internet regulations. Although DTLS [16], MQTTS [17], BLE-TLS [18], and µTLS [19], are lightweight IoT applications, the restrictive and impractical stipulations are still the same. In addition to the X.509 infrastructure, this includes expensive privacy enforcement, and a library for each protocol. Confidentiality, integrity and message payload authentication are enforced by the secure channel TLS creates between the client and server. In many ways this is positive. Confidentiality guards against eavesdropping, integrity ensures non-tampering, and authentication proves the message is from the intended origin - but privacy is not always necessary. Using TLS for any protocol has this same issue. Following authentication, TLS uses a shared symmetric key to encrypt and decrypt data transferred between the two devices. This key is used by a cipher, most commonly the Advanced Encryption Standard (AES), enforcing unnecessary protection over insensitive data. In addition, even the smallest recommended AES key strength, 128-bits, cannot currently be broken in less than around a million years. A final thought towards the threat landscape of the application is use of protocols for communications between the edge devices and the server or central gateway [20] - beyond the 100 meter range of BLE, it is inaccessible. Compared against the global accessibility of traditional internet communications, the risk likelihood is very low. For a decentralised or even distributed heterogeneous network of IoT devices, a scheme providing a 1:M, or M:N relationships between devices and their respective keys [21], would have been ideal. However, the mbedtls library native to the ESP32 does not currently provide such freedom, and so we must proceed this study with cryptographic primitives based on a 1:1 client-server relationship.

III. Related Work

Self-signed certificates and certificate-less security are considered as related work, followed by literature references towards energy consumption testing.

A. Self-signed certificates

There are two ways by which to authenticate the client and server devices in the TLS connection using X.509 certificates. The first and most traditional way is to send the device's Public-key and a Certificate Signing Request (CSR), to a trusted CA, and the second is for the client (ESP32), to act as a CA by generating a root certificate itself and signing it using a Digital Signature Algorithm (DSA); providing verifiable ownership and thus, authenticity. Self-signed certificates have not typically been effective on browsers, since they come with a warning sign that the certificate cannot be trusted due to potential interception between client and server. However, the IoT application will not use a browser for aquaponics readings, and all edge devices will have a trusting relationship with the server since they are part of the same smart-farm. This is very different from interrogating an unknown web server possibly on the other side of the world. Using self-signed certificates, and the DSA, removes many problems of centralised TLS and PKI security.

B. Certificate-less security

On the note of using self-signed certificates, certificate-less key escrow schemes [22], have demonstrated good operation in IoT using the DSA included in the certificate, but with a partial private key to perform the signature. This model

is positive towards an alternative solution for TLS, since the devices are capable of creating key pairs and signatures locally, thus enabling the edge devices to accept responsibility for all secure data exchanges. Outsourcing cryptographic processing to edge devices contributes towards scalability, robustness, and decentralisation, with the additional benefit of energy efficiency - the ESP32 hosts dedicated hardware acceleration for four cryptography functions. Blockchain has demonstrated success in numerous decentralised security applications including cryptocurrencies [23], [24], and IoT [25], [26]. Blockchain is a type of Distributed Ledger Technology (DLT), comparable with a simple bookkeeping ledger where the weighting of profit and expenses is agreed between two parties, and the verification then becomes a block of data added to an ever increasing chain [27]. Applications have been scalable and robust alternatives to the centralised PKI, used in industrial processes [28], farming marketplaces [29] and harvesting [30]. There are two obvious problems with blockchain. Firstly the use of dedicated nodes to hold the smart contracts or consensus mechanism [31], and the full chain [32] - and secondly the continuous mining of the chain for verified transactions, as this quickly exhausts battery life [33], [34], creates complexity, and subsequent security issues [35]. The chain could be stored on the single Pi server, but that creates a similar type of vulnerability to PKI unless there were backup chains beyond the network, or additional nodes. Perhaps a blockchain application for authorisation would be more appropriate than authentication. An authorisation chain would be considerably shorter, since not every day does a device join the network - but it does require key reset. Since additional device authorisation is comparatively seldom, so is mining, transaction 'block' storage, and network scaling.

C. Measuring energy consumption

Energy consumption measurement methods are included as related work specifically for the ESP32 as very low values. Microseconds and milliwatts are the scales of time and power respectively, that ESP-based sketches produce - the differences between security attributes are extremely small. Nevertheless, the impact of an effective security implementation towards battery longevity is years compared against an inappropriate implementation. With reference to [36], [37], the oscilloscope has been illustrated as an effective way of measuring very small voltage levels. With the aid of a shunt resistor of a known value, the oscilloscope can be used to determine current, and subsequently power can be calculated. Using a probe of suitable amplification, units in milli-sizes can be ascertained, and the oscilloscope has been used for suitable accuracy for testing the ESP32 [38]. Fortunately, the ESP32 already has access to timing functions in the native Arduino library micros() [39]). Using time and power, energy can be calculated in millijoules (mJ), to demonstrate battery life differences in the results.

IV. Design and Testing

This section presents the device authentication functions and testing framework; key generation, signing and verification, key exchange methods, and data exchanges. The aim is to discover the most energy-efficient authentication method for subsequent data exchanges.

A. Key Generation

Authentication keys are required for both client and server, and to be kept secret at least in part. These can be two very different asymmetric keys, or they can be a symmetric shared key built from a contribution of data from both client and server. Either way, authentication keys are very different to session keys for encrypting and decrypting readings. The session key is invoked following authentication. The authentication key types available [40], for this study are :

1) Rivest, Shamir and Adleman (RSA); creators of an asymmetric key pair used in Public Key Cryptography (PKC), where the device's publicly-available key is used to lock a message, and upon receipt, the private key is used to unlock the message. RSA has been used as a long-term standard due to its strength, but is notoriously heavy and the key bit-sizes are very large. The ESP32 hosts dedicated hardware for RSA acceleration. Energy testing of RSA key pair generation will demonstrate the energy consumptions of key bit-sizes 2048, 3072, 4096, 7680, and 15,360. 1024 is considered insecure, and deprecated.
2) Elliptic Curve Cryptography (ECC); is a lightweight approach to RSA, based on the algebraic structure of elliptic curves over finite fields - allowing the key sizes to be around a third of RSA sizes, but with equal security. There are 13 curves available to the mbedtls library for call on the ESP32, but two have been deprecated due to insufficient strength. Energy testing of the ECC key pair generation will demonstrate the energy consumptions of curves SECP224R1, SECP256R1, SECP384R1, SECP521R1, SECP224K1, SECP256K1, BP256R1, BP384R1, BP512R1, CURVE448, and CURVE25519. SECP192R1 and SEC192K1 are the approximate equivalents of RSA-1024 and are therefore considered unsafe.

B. Signing and Verification

The Digital Signature Algorithm (DSA), was described earlier as the only valuable component of the traditional X.509 certificate. Using this signature, both server and client devices can authenticate their own key pair immediately after generation. This means that instead of using a certificate and associated infrastructure to prove the origin of a public-key, it can be undertaken in conjunction with generation using one of two methods:

1) RSA sign and verify; the message is signed with an RSA computation of the sender's private-key, and is verified by the recipient by using the RSA computation of the sender's public-key.

2) ECC sign and verify; the message is signed with an ECC computation of the sender's private-key, and is verified by the recipient by using the ECC computation of the sender's public-key.

Energy testing of sign and verify functions for energy consumptions can be recorded at the key generation stage, since the key sizes and curves will influence time and power. Therefore, key generation, sign and verify functions will be tested together.

C. Key Exchange Methods

The key exchange method is how to set up a secure channel in a hostile environment using either an asymmetrical method, where one device does the majority of the processing, or mutual authentication, where contributions are equal between client and server device. Key exchange can be undertaken using:

1) RSA-PKC; RSA-Public Key Cryptography is an asymmetric method in which the session key (typically a 128-bit AES key for IoT), is encrypted using the RSA private-key, and decrypted on the recipient side using the sender's public-key. The sender would have to generate the RSA key pair first, and sign the message (AES key), in addition to encrypting it, to demonstrate authenticity as a unique and genuine sender. Fortunately, the authentication of the client is already satisfied with it generating and sending the session key to the server, and only the server needs to prove its identity. Energy testing will consist of RSA encryption and decryption firstly for an AES128-bit key, and secondly for keys AES128, IV (also 128-bit), and HMAC (256-bit), to represent the additional work for the design. It also includes one signature and verification to represent interrogation of the server, using the lightest RSA key pair.
2) DHM; Diffie-Hellman-Merkle is the mutual authentication key exchange for both client and server to contribute equal work under the key generation of RSA 2048 and above. Energy testing of DHM requires the RSA key list as with key pair generation, sign and verify. As with RSA-PKC, this shall be performed firstly with generating an AES128, and secondly with the AES128, IV, and HMAC to represent the new design.
3) EC-PKC; Elliptic Curve Public Key Cryptography would have been the lightweight equivalent to RSA-PKC by encrypting and decrypting the AES session key using the elliptic curves. Unfortunately, elliptic curve encryption and decryption is not part of the standard TLS ciphersuite (REF RFC 2948 TLS), and so it has not been ported to mbedtls for use on the ESP32. This cannot be tested.
4) ECDHE; or Elliptic Curve Diffie-Hellman Ephemeral, is a key exchange method in which both client and server create a secure channel by contributing equal work. Each party generates and keeps their own secret whilst exchanging a factor of that number to derive the same shared secret key. This is mutual authentication, and requires signing and verification for both parties. Energy testing is undertaken by recording the group of 11 suitable curves under the ECDHE protocol, with a sign and verify function to represent each party, based on the lightest curve drawn from the previous result. As previously, this shall be performed firstly with generating an AES128, and secondly with the AES128, IV, and HMAC to represent the new design.

D. Data Exchanges

Data exchanges are not part of authentication, but are important to exemplify the energy-saving differences between normal TLS implementation and the design - they represent sensor readings or system instructions. A structural difference with the keys allows subsequent selectivity with the security attributes of confidentiality (privacy), integrity, and availability. When using key exchange methods in PKC, the asymmetric key generation takes place on the edge device first, and then the key is encrypted with an asymmetric key, to be decrypted by the server, and used for data exchanges. When using mutual key exchange methods, DHM or ECDHE, a shared key is invoked on the premise of either RSA or elliptic curve computation - both asymmetric. Following this initial shared secret session, the asymmetric key can then be generated for data exchanges. Either way round, the symmetric key(s) must be generated at some point. In addition to this, the design makes use of the IV and HMAC as keys in their own right. Below are details of the data exchange attributes for testing.

1) AES; 128 bit-strength of 16 characters when implementation is defined as a byte per character for simplicity. This is the shared secret key required for encrypting and decrypting messages, and both authenticated devices will store the same key until it is reset. Traditionally the AES key represents confidentiality, or message privacy, although many AES modes incorporate integrity and authentication functions as well, which will also require testing:
   a) AES128 in GCM mode; Galois Counter Mode is a complex approach weighing heavy on the processing abilities of the ESP32, but including integrity and authentication functions as part of an all-in-one. This all-in-one is known as Authenticated Encryption with Added Data (AEAD), and simplifies the implementation process. This is quickly becoming the de facto AES mode standard for TLS, and shall demonstrate the energy consumption of a strong, simple, AEAD mode.
   b) AES128 in CCM; Counter with Cipher Block Chaining Message Authentication Code, or CBC-MAC, is the equivalent of AES128 in CBC mode with AEAD properties. It will be tested as three individual functions that can be used together or separately, for fulfillment of the security attributes confidentiality, integrity, and authentication.
2) IV; Initialization Vector is required for CBC mode to function. The IV is unique and can offer additional

hardening if the same one is never used twice. In the design, it is one of three session keys. In TLS, it is refreshed regularly, as it is part of the session timeout and key reset of TLS sessions. The IV will provide a new shared session key for the design of 16 characters or bytes.

3) HMAC; Hash-keyed Message Authentication Code is the authentication function which can be implemented as a key in its own right for proving the message has come from its claimed origin. The HMAC also contains the SHA integrity component, and therefore proves non-tampering. SHA256 will be used for testing, and so this additional session key is a 32-character HMAC-SHA256.
4) Data exchange server to client; messages sent from the Raspberry Pi to the ESP32 will most likely be long, infrequent, and sensitive. They may contain private instructions, device properties, and identification of living subjects for example. For these reasons, 256 characters will be used as the maximum amount of a server to client set, and the data will be fully protected, with encryption, integrity, and authentication functions.
5) Data exchange client to server; messages sent from the ESP32 to the Raspberry Pi will most likely be short and insensitive readings for temperature, humidity, light and pH levels. 16 characters can provide abbreviated reading content and they do not need protection beyond integrity and authentication. For these reasons, the design will demonstrate energy-saving capabilities of selective AEAD, resulting from the structural changes in the authentication process; the two additional keys, IV and HMAC.

V. Implementation and Results

This section presents details on measuring the entropy and key exchange protocols, time and power sampling to calculate energy consumption, the six sets of results, and how the security objectives have been satisfied.

A. Measuring entropy and key exchange protocols

When undertaking key generation, the ESP32 will employ the Random Number Generator (RNG), hosted as part of its dedicated hardware-accelerated cryptography block. This generator is called by 'ctr_drbg' in the mbedtls library - a deterministic random number generator function. Measuring time consumption of such a process is difficult, as entropy is inherently inconsistent - but indicative of invoked randomness. For this reason, a sample of ten readings for each protocol were recorded, from which an average time could be deduced as an example of 'normal' consumption time.

B. Time and power sampling

TLS and the design implementation can be undertaken very accurately by calculating energy (measured in Joules); by time (measured in microseconds, μS), multiplied by power (measured in milliwatts, mW). Time can be printed to the serial port, and power can be sampled using an oscilloscope. The ESP IoT Development Framework (ESP-IDF), employed the Arduino library as a component. This allowed high control of the ESP board through the IDF's 'menu-config', in order to switch off the WatchDog Timer (WDT). Menuconfig → Component Config → Common ESP related → Disable Interrupt WatchDog, Disable Initialise Task Watchdog Timer on Startup. This was important to ensure that the board did not enter 'boot loop', or interruption, during testing in high repetition cycles - preventing the board from completing each task. The micros() function from the Arduino library returned the timing of any part of the sketch. Micros() was therefore used to measure the context invocation, key generation, signing, verification, and freeing of contexts without also recording the time used for variables and serial port printing. The resulting microseconds were printed to the serial port and recorded, with the exception of the first result of any set, which includes board initialisation and thus counts as an anomaly. Repetitions of 100 loops per task were performed to eradicate errors of single units, but produce results within a reasonable time - loops of 1000 proved too high an expectation of the ESP32, particularly in RSA - causing the 'guru meditation' [41], or stack overflow error (despite disabling the WDT), so all experiments undertook 100 loops to give a fair test. With the curves requiring more entropy (bigger key strengths), even loops of 10 took considerable time. Measuring power required the use of an oscilloscope, to measure the voltage of the board with a given resistance, for calculating current. Where voltage (measured in microvolts, μV), (Ohms, Ω, provides current, represented by the letter I (measured in Amperes, mA), power can be derived; where voltage multiplied by current gives power. A 5V power bank was used to ensure voltage was consistent, and current was measured across the circuit to provide results with a clear difference.

C. Results sets

As discussed in the design section, key pair generation with sign and verify were tested together, and so the following experiments were undertaken to then compare energy consumptions of authentication and data exchanges between TLS and the design:

1) Key pair generation, sign and verify
   a) RSA; 5 key strengths.
   b) ECDSA; 11 curves.
2) Key exchange
   a) RSA-PKC; for one key (TLS), and three keys (design).
   b) ECDHE; for one key (TLS), and three keys (design).
3) Data exchange
   a) From server to client; fully protected GCM and CCM, typical of TLS.
   b) From client to server; selective CCM and excluding privacy, typical of the design.
4) Energy consumption comparison over a 24 hour period

a) Readings; 16 character readings comparing TLS and the design.
b) Instructions; 256 character instructions comparing TLS and the design.

The results sets are compared in order to demonstrate the lightest applications of security with the same standards as TLS, but with lightweight configurations:

TABLE I. 1A) RSA KEY PAIR GENERATION, SIGN AND VERIFY

RSA key	**2048**	**3072**	**4096**	**7680**	**15,360**
Joules	1.390	4.664	11.532	113.350	Error
Millijoules (mJ)	1390	4664	11,532	113,350	Error

TABLE II. 1B) ECDSA KEY PAIR GENERATION, SIGN AND VERIFY

ECDSA curve	**SECP224R1**	**SECP256R1**	**SECP384R1**
Joules	0.160	0.231	0.316
Millijoules (mJ)	160	231	316
ECDSA curve	**SECP521R1**	**SECP224K1**	**SECP256K1**
Joules	0.526	0.246	0.285
Millijoules (mJ)	526	246	285
ECDSA curve	**BP256R1**	**BP384R1**	**BP512R1**
Joules	2.557	4.811	9.769
Millijoules (mJ)	2557	4811	9769
ECDSA curve	**Curve448**	**Curve25519**	-
Joules	Error	0.057	-
Millijoules (mJ)	Error	57	-

RSA demonstrated a phenomenal difference compared with ECDSA - the lightest RSA energy consumption was around 24 times that of elliptic curve mutual authentication, despite the dedicated RSA hardware. It was therefore pointless to carry down RSA into a key exchange comparable with ECDHE (TABLE II). Curve 448 demonstrated no entropy at all, repeated the same key many times, and so was considered an error.

Since RSA-PKC (TABLE III), was considerably simpler in process, it may have consumed less energy than a mutual authentication process.

TABLE III. 2A) RSA-PKC KEY EXCHANGE

RSA-PKC	**RSA in 2048-bit strength PKC with ECDSA for AES 128-bit key**	**RSA in 2048-bit strength PKC with ECDSA for three keys**
Joules	1.450	1.622
Millijoules (mJ)	1450	1622

TABLE IV. 2B) ECDHE KEY EXCHANGE

ECDHE	**ECDHE in Curve 25519 with ECDSA for AES 128-bit key**	**ECDHE in Curve 25519 for three keys**
Joules	0.283	0.284
Millijoules (mJ)	283	284

Even with the simpler process, PKC employing RSA consumed around five times more energy than the elliptic curve mutual authentication model (TABLE IV). This would probably have been even lighter if there was an elliptic curve PKC function available to the ESP32. Although the RSA and ECC difference is less in key exchange than key pair generation, it is still a huge drain of battery lifespan in comparison.

TABLE V. 3A) FROM SERVER TO CLIENT

Reading length in character bytes	**AES128 in GCM mode (mJ)**
16	0.0224
64	0.0376
128	0.0579
256	0.0985
Reading length in character bytes	**AES128 in CCM mode with full AEAD (mJ)**
16	0.0224
64	0.0376
128	0.0579
256	0.0985

TABLE VI. 3B) FROM CLIENT TO SERVER

Reading length in character bytes	**AES128 in CCM without privacy (mJ)**
16	0.0189
64	0.0211
128	0.0211
256	0.0168

GCM mode encouraged in TLS showed to use energy at an increased rate when compared to CCM (TABLE V). At smaller data exchanges, GCM was in fact much lighter - but at smaller levels, full protection was not required anyway and so the saving was irrelevant.

TABLE VII. 3A) COMPARISON OF ENERGY OVER A 24-HOUR PERIOD 16 CHARACTER BYTES IN MJ

ECDHE in Curve 25519 with ECDSA for AES 128-bit key with readings sent in GCM	**ECDHE in Curve 25519 with ECDSA for AES 128-bit key with readings sent in CCM**
7075.56	7075.488
ECDHE in Curve 25519 for three keys and readings including privacy, Pi to ESP32	**ECDHE in Curve 25519 for three keys and readings excluding privacy, ESP32 to Pi**
284.4875	284.4725

GCM and CCM employed by TLS show such a drastic difference because of the full authentication process undertaken on every reading compared with selective AEAD (TABLE VI). If one reading is sent every hour, and each reading requires device authentication before it can exchange data, the process is going to show the energy consumption of authentication 24 times. With the design, the authentication process is undertaken every 24 hours. Between full protection and removing privacy within the design, there is a difference

of 0.015 millijoules. This does not sound much - it is around 5.475 millijoules a year. However, comparing the GCM of TLS over a year, at 2,582,579 millijoules, against not using privacy on the basis of this design, at 103,832 millijoules a year, the design shows a decrease of annual usage by around 96% (TABLE VII and TABLE VIII).

TABLE VIII. 3B) COMPARISON OF ENERGY OVER A 24-HOUR PERIOD 256 CHARACTER BYTES IN MJ

ECDHE in Curve 25519 with ECDSA for AES 128-bit key with instructions sent in GCM	**ECDHE in Curve 25519 with ECDSA for AES 128-bit key with instructions sent in CCM**
7077.4625	7075.448
ECDHE in Curve 25519 for three keys and instructions including privacy, Pi to ESP32	**ECDHE in Curve 25519 for three keys and instructions excluding privacy, ESP32 to Pi**
284.4475	284.42

A similar situation applies to instruction set sizes, at 256 bytes. Over a year, following this example, TLS using GCM mode shows an energy consumption of 2,583,274 millijoules, and the design gives a consumption of 103,813. In this scenario, the design also illustrates a saving of 96%. If authentication was undertaken every week or month compared to TLS authenticating every hour, this would increase to 100's, and 1,000's of percent in energy consumption saving - potentially changing the feasibility of security as part of a very constrained project.

D. Satisfying the security objectives

The security objectives were introduced at the beginning of this paper, highlighting the issues which could arise whilst integrating TLS into an aquaponics application domain. From the results, these security objectives can now be discussed.

1) Privacy concerns: The privacy provided by TLS is arguably the opposite of what it should be for an IoT smart-farm. Where TLS employs PKI, key escrow allows the unauthorised browsing of data exchanges by authorities who wish to view it, such as the government. This may not be preferable as the 'snoopers charter' created quite a disturbance [42]. In contrast, benign data such as temperature and pH levels which need not be protected are under confidentiality enforcement as part of the TLS ciphersuite model. To rectify this unsuitability for the aquaponics setup, the design contained no key escrow, and proposed the separation of security attributes at the authentication stage to permit subsequent selectivity. As a result, insensitive data will not use unnecessary battery power, and unauthorised observers will be disallowed as part of the decentralised model made possible by edge device processing.

2) certificates: Removing certificates from the infrastructure enables authentication to operate without a centralised architecture. This is both more robust than having a single point of failure, and more scalable. As with certificate-less schemes, signing and verification can be used for proving origin without the full certificate. Asymmetric ECC cryptography was used for sign and verify functions, utilising ECDSA as an effective replacement for a full X.509.

3) Session timeout: With reference to [43], the table below illustrates a comparison of key size and strength:

TABLE IX. KEY STRENGTH COMPARISON

AES bit key length	**RSA bit key length**	**ECC bit key length**
112	2048	224
128	3072	256
128-192	4096	256-384
192	7680	384
256	15,360	512

Since 2015, NIST has recommended an RSA key size of 2048-bit size, or the ECC equivalent of 224-bit [44]. At the time of writing, an RSA key of this size, or the equivalent ECC key, would take an estimated 300 trillion years to break by a classical computer, and in around eight hours using a quantum computer. Quantum computers are not expected for a few years, and even when commonplace, re-authenticating every eight hours rather than every few minutes would be advantageous to the smart-farm design. In addition, utilising a protocol that did not allow world-wide access such as the internet would further reduce the threat landscape. The results concluded that by using the ECC Curve 25519 rather than its RSA 3072 equivalent, that key generation, sign and verify functions alone can show the ECC as a 98.7% saving. Applied to authentication once every 24 hours rather than once a reading, these energy savings demonstrate a 96% difference when compared against modern TLS recommendations. Particularly with the persistent client and server relationships of the smart-farm, resetting device authentication over periods of days, weeks or even months has proven to show 100s or 1,000s of percent.

4) Individual protocol security: The ESP32 has demonstrated its abilities to process cryptographic primitives and protocols very effectively on-board, and so security can be distributed to the edge devices within the smart-farm application. Apart from the benefits of a decentralised model, it also negates the need to employ TLS ciphersuites - and therefore the rules of ciphersuites can be ignored. BLE, LoRaWAN, MQTT, and any number of other protocols can be secured by payload, and also selectively. The design encourages selective privacy by separating AEAD functions available to the mbedtls library, enabling further energy consumption reductions on any protocol, without a library for each.

5) Testing for the lightest energy consumption: By testing all the adequately secure ECC and RSA key sizes, the results have shown that Curve 25519 in application of ECDHE, is the lightest available authentication process available to the ESP32. In addition, providing three session keys during authentication instead of the traditional AES key alone can provide enormous energy savings, which can be critical for smart-farm lifespan when relying on constrained devices.

VI. Conclusion and further work

This study concludes that a design with structural differences to TLS, but employing the same security standards as recommended by NIST and FIPS, can provide energy consumption savings of around 96% on a basis of one reading an hour, every hour for a full day. This study has ascertained positive results for resolving concerns pertaining to TLS:

1) Privacy concerns; removing key escrow but enabling privacy for readings has hardened overall security and saved energy on unnecessary protection.
2) Certificates; removing X.509 certificates has enabled a decentralised model with less overhead and less structural setup.
3) Session timeout; reducing frequency of session timeout has proven great energy savings, less processing, and no less protection given the strength of keys.
4) Individual protocol security; ciphersuite regulations and individual libraries for each protocol are no longer an issue since data can be secured by payload.
5) Lowest energy consumption; studying the available authentication mechanisms and applying IV, integrity, and availability keys to the session setup has hardened security whilst permitting selective privacy - further contributing to the energy saving of 96%.

Further work could include Elliptic Curve Public Key Cryptography (EC-PKC), where the session keys AES, IV and HMAC are generated at source and then encrypted using the lighter elliptic curve alternative to RSA. Unfortunately, EC-PKC is unavailable to the mbedtls library since it is not part of the TLS suite, yet RSA-PKC is available - despite the high energy consumption. EC-PKC using Curve 25519 could be an even lighter alternative to ECDHE. In addition, cryptocurrency technologies are also utilising Zero-Knowledge Succinct Non-interactive ARgument of Knowledge (ZK-SNARKs) [45], or the 'secret cave' protocol [46], as a keyless authentication mechanism currently used in Z-Cash [47]. Cryptocurrencies in general and blockchain applications such as IOTA are exemplifying lightweight capabilities suitable for IoT - even if only for the authentication mechanisms and not to store the blockchain itself.

References

[1] J. E. Rakocy, "Integrating hydroponic plant production with recirculating system aquaculture: Some factors to consider," *Recirculating Aquaculture, Roanoke, Virginia, Virginia-Tech*, 1998.

[2] W. A. Lennard, "Aquaponics research at RMIT university, melbourne australia," *Aquaponics Journal*, vol. 35, pp. 18–24, 2004.

[3] H. Cunningham and B. Kotzen, "Meet the sustainable vegetables that thrive on a diet of fish poo," *The Conversation*, Nov. 2015.

[4] S. Goddek, B. Delaide, U. Mankasingh, K. Ragnarsdottir, H. Jijakli, and R. Thorarinsdottir, "Challenges of sustainable and commercial aquaponics," *Sustain. Sci. Pract. Policy*, vol. 7, no. 4, pp. 4199–4224, Apr. 2015.

[5] B. Rieger, G. Demeneghi Ludke, K. Kumykov, R. G. Chatani, and R. I. Hoxha, "Designing an aquaponic greenhouse for an urban food security initiative," *Worcester Med. News*, 2015.

[6] A. Yousuf, *Microalgae Cultivation for Biofuels Production*. Academic Press, Nov. 2019.

[7] T. Y. Kyaw and A. K. Ng, "Smart aquaponics system for urban farming," *Energy Procedia*, vol. 143, pp. 342–347, Dec. 2017.

[8] D. Díaz-Sánchez, A. Marín-Lopez, F. Almenarez, P. Arias, and R. S. Sherratt, "TLS/PKI challenges and certificate pinning techniques for IoT and M2M secure communications," *IEEE Communications Surveys Tutorials*, pp. 1–1, 2019.

[9] M. Y. Afanasev, A. A. Krylova, S. A. Shorokhov, Y. V. Fedosov, and A. S. Sidorenko, "A design of cyber-physical production system prototype based on an ethereum private network," in *2018 22nd Conference of Open Innovations Association (FRUCT)*, May 2018, pp. 3–11.

[10] D. Li, W. Peng, W. Deng, and F. Gai, "A Blockchain-Based authentication and security mechanism for IoT," in *2018 27th International Conference on Computer Communication and Networks (ICCCN)*, Jul. 2018, pp. 1–6.

[11] J. Larisch, D. Choffnes, D. Levin, B. M. Maggs, A. Mislove, and C. Wilson, "CRLite: A scalable system for pushing all TLS revocations to all browsers," in *2017 IEEE Symposium on Security and Privacy (SP)*, May 2017, pp. 539–556.

[12] T. Meskanen, V. Niemi, and J. Kuusijäarvi, "Privacy-Preserving peer discovery for group management in p2p networks," in *2020 27th Conference of Open Innovations Association (FRUCT)*, Sep. 2020, pp. 150–156.

[13] H. Abelson, R. Anderson, S. M. Bellovin, J. Benaloh, M. Blaze, W. w. Diffie, J. Gilmore, M. Green, S. Landau, P. G. Neumann, R. L. Rivest, J. I. Schiller, B. Schneier, M. A. Specter, and D. J. Weitzner, "Keys under doormats," pp. 24–26, 2015.

[14] D. Shaw, "Hundreds arrested as crime chat network cracked," *BBC*, Jul. 2020.

[15] "Ieft rfc5246," https://datatracker.ietf.org/doc/html/rfc5246, accessed: 2021-6-23.

[16] T. Kothmayr, C. Schmitt, W. Hu, M. Brünig, and G. Carle, "DTLS based security and two-way authentication for the internet of things," *Ad Hoc Networks*, vol. 11, no. 8, pp. 2710–2723, Nov. 2013.

[17] N. Nikolov and O. Nakov, "Research of secure communication of esp32 IoT embedded system to.NET core cloud structure using MQTTS SSL/TLS," 2019.

[18] H. Chiang, J. Hong, K. Kiningham, L. Riliskis, P. Levis, and M. Horowitz, "Tethys: Collecting sensor data without infrastracture or trust," in *2018 IEEE/ACM Third International Conference on Internet-of-Things Design and Implementation (IoTDI)*. ieeexplore.ieee.org, Apr. 2018, pp. 249–254.

[19] "TLS (transport layer security) running on an arduino UNO," https://forum.arduino.cc/t/tls-transport-layer-security-running-on-an-arduino-uno/426291, Dec. 2016, accessed: 2021-7-12.

[20] O. Galinina, K. Mikhaylov, S. Andreev, A. Turlikov, and Y. Koucheryavy, "Smart home gateway system over bluetooth low energy with wireless energy transfer capability," *Eurasip J. Wirel. Commun. Network.*, vol. 2015, no. 1, pp. 1–18, Jun. 2015.

[21] V. Petrov, S. Edelev, M. Komar, and Y. Koucheryavy, "Towards the era of wireless keys: How the IoT can change authentication paradigm," in *2014 IEEE World Forum on Internet of Things (WF-IoT)*, Mar. 2014, pp. 51–56.

[22] M. K. Aditia, S. Paida, F. Altaf, and S. Maity, "Certificate-less public key encryption for secure e-healthcare systems," in *2019 IEEE Conference on Information and Communication Technology*. ieeexplore.ieee.org, Dec. 2019, pp. 1–5.

[23] P. Tarasov and H. Tewari, "Internet voting using zcash," *IACR Cryptol. ePrint Arch.*, vol. 2017, p. 585, 2017.

[24] D. Puthal, N. Malik, S. P. Mohanty, E. Kougianos, and G. Das, "Everything you wanted to know about the blockchain: Its promise, components, processes, and problems," *IEEE Consumer Electronics Magazine*, vol. 7, no. 4, pp. 6–14, Jul. 2018.

[25] D. Fakhri and K. Mutijarsa, "Secure IoT communication using blockchain technology," in *2018 International Symposium on Electronics and Smart Devices (ISESD)*, Oct. 2018, pp. 1–6.

[26] W. F. Silvano and R. Marcelino, "Iota tangle: A cryptocurrency to communicate Internet-of-Things data," *Future Gener. Comput. Syst.*, vol. 112, pp. 307–319, Nov. 2020.

[27] A. Pieroni, N. Scarpato, and L. Felli, "Blockchain and IoT Convergence—A systematic survey on technologies, protocols and security," *NATO Adv. Sci. Inst. Ser. E Appl. Sci.*, vol. 10, no. 19, p. 6749, Sep. 2020.

[28] N. Teslya and I. Ryabchikov, "Blockchain platforms overview for industrial IoT purposes," in *2018 22nd Conference of Open Innovations Association (FRUCT)*, May 2018, pp. 250–256.

[29] G. Leduc, S. Kubler, and J.-P. Georges, "Innovative blockchain-based farming marketplace and smart contract performance evaluation," *J. Clean. Prod.*, vol. 306, p. 127055, Jul. 2021.
[30] T. H. Pranto, A. A. Noman, A. Mahmud, and A. B. Haque, "Blockchain and smart contract for IoT enabled smart agriculture," p. e407, 2021.
[31] B. Cao, Y. Li, L. Zhang, L. Zhang, S. Mumtaz, Z. Zhou, and M. Peng, "When internet of things meets blockchain: Challenges in distributed consensus," *IEEE Netw.*, vol. 33, no. 6, pp. 133–139, Nov. 2019.
[32] M. Gorodnichev, A. Kukharenko, E. Kukharenko, and T. Salutina, "Methods of developing systems based on blockchain," https://www.fruct.org/publications/acm24/files/Gor.pdf, accessed: 2021-5-11.
[33] M. Salimitari, M. Chatterjee, and Y. Fallah, "A survey on consensus methods in blockchain for resource-constrained IoT networks."
[34] D. Puthal, S. P. Mohanty, V. P. Yanambaka, and E. Kougianos, "PoAh: A novel consensus algorithm for fast scalable private blockchain for large-scale IoT frameworks," *CoRR*, Jan. 2020.
[35] T. Krupa, M. Ries, I. Kotuliak, K. Košťál, and R. Bencel, "Security issues of smart contracts in ethereum platforms," in *2021 28th Conference of Open Innovations Association (FRUCT)*, Jan. 2021, pp. 208–214.
[36] R. L. Ponting, "The oscilloscope as a primary measurement tool," *Phys. Teach.*, vol. 29, no. 6, pp. 401–403, Sep. 1991.
[37] R. A. Kjellby, L. R. Cenkeramaddi, A. Frøytlog, B. B. Lozano, J. Soumya, and M. Bhange, "Long-range self-powered IoT devices for agriculture aquaponics based on multi-hop topology," in *2019 IEEE 5th World Forum on Internet of Things (WF-IoT)*, Apr. 2019, pp. 545–549.
[38] A. Maier, A. Sharp, and Y. Vagapov, "Comparative analysis and practical implementation of the ESP32 microcontroller module for the internet of things," in *2017 Internet Technologies and Applications (ITA)*, Sep. 2017, pp. 143–148.
[39] "micros()," https://www.arduino.cc/reference/en/language/functions/time/micros/, accessed: 2021-6-23.
[40] ARM Limited, "ecp.h file reference - API documentation - mbed TLS (previously PolarSSL)," https://tls.mbed.org/api/ecp_8h.html, accessed: 2021-7-12.
[41] "Fatal errors - ESP32 - — ESP-IDF programming guide latest documentation," https://docs.espressif.com/projects/esp-idf/en/latest/esp32/api-guides/fatal-errors.html, accessed: 2021-7-12.
[42] B. Gill, "The wheel of reincarnation turns again: Time for another go at the clipper chip," *Researchgate,[SI], dec*, 2019.
[43] S. Li, "IoT node authentication," in *Securing the Internet of Things*. Syngress Boston, 2017, pp. 69–95.
[44] E. Barker and Q. Dang, "Nist special publication 800-57 part 1, revision 4," *NIST, Tech. Rep*, vol. 16, 2016.
[45] A. M. Pinto, "An introduction to the use of zk-SNARKs in blockchains," in *Mathematical Research for Blockchain Economy*. Cham: Springer International Publishing, 2020, pp. 233–249.
[46] B. Schneier, *Applied Cryptography: Protocols, Algorithms and Source Code in C*. John Wiley & Sons, Mar. 2015.
[47] A. Banerjee, M. Clear, and H. Tewari, "Demystifying the role of zk-SNARKs in zcash," in *2020 IEEE Conference on Application, Information and Network Security (AINS)*, Nov. 2020, pp. 12–19.