NTNU – Trondheim
Norwegian University of
Science and Technology

Cryptoprocessing on the Arduino

Protecting user data using affordable
microcontroller development kits

Stig Tore Johannesen

Master of Science in Informatics
Submission date: August 2014
Supervisor: Guttorm Sindre, IDI
Co-supervisor: Danilo Gligoroski, ITEM

Norwegian University of Science and Technology
Department of Computer and Information Science

Acknowledgements

I would like to thank my thesis advisor, Guttorm Sindre, for all his help during the writing process.
I would also like to thank my co-advisor, Danilo Gligoroski, for getting me started on the technical aspect of the report.
Thanks are also owed to everyone who helped me by reading through and commenting on the structure and language of this report.

Abstract

There is a growing trend of data breaches, which this report looks into. The breaches often turn out to have, at their core, an element of either poor security management, or outdated or incorrectly applied security procedures. With this in mind, an affordable off-the-shelf microcontroller development kit is suggested as a way to lessen the impact of data theft during data breaches. Utilising an Arduino Due this report looks at the performance available for cryptoprocessing and key storage, showing that while it is not a viable solution for encrypting large amounts of data, it is however suitable for securely encrypting limited data sets, such as customer data.

Table of Contents

Chapter 1

Introduction

There is a growing trend of large database and user information leaks. This, combined with non-standard, or deficient security methodologies relating to storage and transmission of the data, have led to a lot of user data, e.g. email addresses, credit card information, and passwords, being compromised.

There has been a lot of data breaches (According to datalossdb.org there were 1651 reported incidents in 2012, and 1465 in 2013) over the last few years, quite a few of which have hit mainstream news [5, 6, 10, 11, 21, 22]. According to [34, 35] this trend is increasing. While there has been a noticeable drop in the number of incidents in 2013 compared to 2012, the number of exposed records has increased to over twice that of any other years, with the Adobe Systems data breach contributing greatly. Hacking is by far the biggest contributor in terms of exposed records, fraud/social engineering, as well as stolen or lost equipment also represents a large portion of the reported incidents[14].

The Ponemon Institute has since 2009 published annual studies on the cost of data breaches. While their analysis explicitly excludes the largest and most published data breaches, what they call mega leaks, their data tries to estimate the cost of the average data breach. Their 2013 study [32] reports the costs as varying between 42 and 199 USD per record, depending on the nationality of the affected company. The lower end of their calculated cost spectrum is occupied by India and Brazil, while the top 5 are the United Kingdom, Australia, France, the USA and Germany. For most western corporations it would be reasonable to assume that a breach would cost between 124 and 199 USD per record, which would include the cost brackets of the breaches in the all countries analysed by the Ponemon Institute, except for India and Brazil. Another notable variation is the per-record cost by industry, where healthcare, financial and pharmaceutical institutions are the most expensive, while public services and retail the least expensive. The study also shows that, with few exceptions, the biggest contributors to the breaches are malicious attacks, and

that this root cause is also the most costly.

While the long term effects on the financial status of the affected companies has not been catastrophic so far, the negative publicity and the direct financial damages to the companies has been far from trivial.

As can be seen from the sources above, data at rest has been consistently shown to be at risk of theft. By encrypting the data, and applying proper key management schemes it is possible to greatly reduce the value of the stolen data to the attacker. While cryptography using known standards, and up-to-date cryptographic algorithms, and verified libraries is reasonably secure, the problem of key management is considered hard[44]. There are however hardware solutions, called Hardware Security Modules (HSM'), that both implement cryptography in a verifiable way, but also store and manage the cryptographic key(s) in such a way as to make them unreachable for an attacker. Which means the data that is stored at rest can be considered secure as long as the HSM is secure. HSM' are however prohibitively expensive for everyone but larger corporations, with prices in the tens of thousands of dollars (Examples being the Sun Crypto Accelerator 6000 at USD 9 950, and the AEP Keyper 9720 starting at USD 16 150).

1.1 Goal

This report will shed light on how vulnerable data-at-rest is to theft through looking at recent data breaches, and illuminate the massive costs of these types of security incidents. This report will also seek to identify key vulnerabilities and poor security practices where the information is available. Then working from the assumption that data-at-rest might be leaked, the goal of this report will be to suggest and evaluate a way of providing key management, especially key storage, and basic cryptographic functions using affordable off-the-shelf hardware in accordance with up to date best practices in data security.

Specifically this report seeks to answer the following research questions:

- Research Question 1: What are the current best practices for securing data stored at rest?
- Research Question 2: Why is it that despite the above best practices data still leaks? Are the practices too abstract, outdated, weak, or are they simply not being implemented properly?
- Research Question 3: What are the major strengths of the available methods in secure data storage (i.e. hashing, cryptography, key management), and what are their most important weaknesses?
- Research Question 4: Could a proof of concept system, utilising affordable off-the-shelf microprocessor kits, alleviate some of the weaknesses of on-server cryptography (most notably key management) without compromising the strenghts?

Chapter 2

Research Methods

There are a lot of methodologies available to provide answers to the questions posed in this report. This section will present the most relevant ones, briefly look at their advantages and challenges, and explain the reasoning behind the methods which have been chosen for this report.

2.1 Possible Methodologies

Two very similar methodologies that are relevant in this context are interviews and questionnaires, both of which provide the same kind of information. Their difference is in how easy they are to plan and execute, how easy it is to analyse and compare the data, and how adaptable they are. While interviews provide the most adaptable and in most cases most detailed information, the subjective nature of the interview will make the data harder to analyse and compare. The questionnaire on the other hand is far easier to analyse and compare, but relies on how well it is designed to provide the information one is trying to gather, and it has next to no adaptability as it needs to stay consistent throughout the study to provide easily analysed data. Their similarity lies in that they both require that the subjects of the study are willing to participate, are able to set aside the time required, and that they are interested in or have incentives to provide correct information.

In the methodology of literature review two very relevant and similar sources are peer reviewed articles, and professional articles (e.g. security reports, yearly analyses). Both of them provide highly reliable information, and are considered the preferred source to base this kind of report upon, when the information is available. While in theory professional articles are not subject to peer review in the same manner as published articles, in practice this point is not really valid provided that the professional articles come from a reputable

source in the relevant field of study. Both of their scopes are however most often limited to a narrow subset of topics within a field, and finding articles on relevant topic is limited by the subjects' willingness and ability to provide relevant information.

Similar to articles is yet another set of sources: news articles, press releases, and "others" (here referring to forum posts, blogs, breach databases). These have to be carefully examined for validity and truthfulness, the latter more so than the two former. But in the context of data breaches these are often the only sources of information that is available and can as such provide valuable information.

A source of information on best practices, expected behaviour, and other expectations in various businesses is standards documents. While they are often vague on specifics, they can be further augmented by looking into related documents such as laws, as well as requirement documents from certain sources such as governments. Together they will often provide valuable insight into best practices in their fields.

When looking at implementations of relevant technologies the proof of concept methodology is an important one when there are few or no available articles on similar systems to investigate. While it does provide a lot of relevant information on how effective a system would be compared to another, and whether it is a viable option, the data validity relies heavily on the implementers skill with the relevant technologies.

2.2 Research Question 1

> What are the current best practices for securing data stored at rest?

For this research question the only truly reliable source of information are standards documents as well as requirements from reputable sources (e.g. government agencies) and laws on data storage.

2.3 Research Question 2

> Why is it that despite the above best practices data still leaks? Are they too abstract, outdated, weak, or are they simply not being implemented properly?

Unfortunately there are few or no incentives for companies to release the details on their data breaches beyond basic notifications that in some areas are required by law. And as openness about breaches can lead to loss of revenue, loss of standing, and open the company up to litigation, reliable information is hard to find. This means that interviews and questionnaires, which would probably provide the best data, are not a viable in the context of this report. There are very few relevant articles on the topic, most likely due to the above

reasons. Some statistically oriented professional articles are available, though few of them go into any single incident in any detail. Because of the difficulties in finding relevant information most of the research for this question will have to rely on other sources, such as news articles, press releases, as well as breach databases and other third party sources.

2.4 Research Question 3

> What are the major strengths of the available methods in secure data storage (i.e. hashing, cryptography, key management), and what are their most important weaknesses?

To answer this question the most natural methodology to use is to review articles and teaching material on the basic methods of the relevant best practices from question 1, with a focus on tendencies discovered when researching question 2. I.E. Q2 will hopefully reveal some core areas in the best practices that are often overlooked, implemented incorrectly, or deficiently.

2.5 Research Question 4

> Could a proof of concept system utilising affordable off-the-shelf microprocessor kits alleviate some of the weaknesses of on-server cryptography (most notably key management) without compromising the strengths?

Looking at cryptography, specifically at how to secure the encryption keys, the question is essentially if an off-the-shelf development kit, like the Arduinos or other micro-controller development kits, are powerful enough to be used. There have been some articles published on using high end microprocessors for this purpose, but none on the extremely low price level and with the easy availability and usability of the Arduino. While there are alternatives such as simulation and estimation, due to the lack of any extensive study on the performance of Arduino boards the most natural approach to answer this question is a proof of concept system. This system can then be compared to high end microprocessors, cryptoprocessors and modern desktop/server systems, where the data is available.

Chapter 3

Background

3.1 Breaches

This section will cover some of the high profile breaches that have occurred in the last few years, and try to, as far as that information is available, name the vulnerabilities or shortcomings that led to the breaches.

Further it will look deeper into the information that is available on the types of breaches that have occurred, and what major consequences they have had on the organisations involved.

It is worth pointing out that this section is not trying to point fingers at specific companies related to the breaches, as the problem of data breaches is endemic, but some cases have received far more publicity than most and thus more details are available on those breaches.

3.1.1 Sony Online Entertainment, 2011

Sony actually suffered from several, apparently, related data breaches on the 16th and 17th of April. The 16th of April breach of Sony Online Entertainment seems to have compromised 24.6 million user accounts consisting of names, addresses, emails, birth dates, gender, phone numbers, login names and hashed passwords, as well as at least 12 700 non-US credit card numbers with expiration dates, and 10 700 direct debit records from Austria, Germany, Netherlands and Spain.

The second breach on the 17th of April, at Sony Corporation, consisted of at least 77 million user entries containing names, addresses, emails, birth dates, PlayStation Network

or Qriocity log-ins and passwords, online IDs for the former services, profile data, and purchase history, and possibly credit card information as well. It is unknown if there is any overlap between this and the 16th of April breach, but given the short amount of time between the breaches the probability is high that they are related to one another.

In response to the data breaches Sony shut down their PlayStation Network and Sony Online services for 24 days, as well as issuing game-time credit to all active subscribers. They also hired an external security company to review their systems. The cause of the breach has not, as of the writing of this report, been published, nor has the stolen data leaked to the public, so it is impossible to tell what kind of security the database and data was protected by. What seems to be the case is that several of the back-end servers for the PlayStation network were running outdated versions of Apache and Linux[9], with the update status of other software on the servers in question as well.

A press release by Sony in 2011 states that the cost of the data breach so far had been 171 million USD[15, 33], which is a lot less than an estimation based on the numbers from Ponemon Institute, but as [15] states the costs and losses are likely to increase both for Sony and any affected customers. The data breach led to at least one class action lawsuit being filed, with Sony offering a settlement estimated to be worth at least 15 million USD[31].

3.1.2 LinkedIn, 2012

On the 5th of June 2012 the popular professional social network LinkedIn was breached by what has been reported as Russian hackers[13], and data on at least 6.4 million of their, at the time, 150 million users was stolen. LinkedIn has not posted details on exactly what data was stolen, but the passwords were posted online shortly after the breach.

The passwords were secured using unsalted SHA-1. Both SHA-1 and unsalted hashes are discouraged. And as such the leak has been one of the biggest assets in recent years, for both hackers and security researchers, as it has provided extensive insight into what passwords people use.

The cost of the leak is, at the writing of this report, impossible to estimate due to the lack of any official statement from LinkedIn, or useful estimates on leaks of this magnitude. At least one lawsuit has been filed, but was dismissed in 2013[12].

3.1.3 Adobe, 2013

Some time previous to the 3rd of October, likely early to mid August, an old customer database backup was stolen from Adobe Systems, followed shortly by the theft of old Adobe source code likely coming from the same server. Initial reports by Adobe puts the number of lost records at 2.9 million, but was later amended to 38 million. A 3.8GB file

was later posted online, containing 152 million entries, confirmed to be from the Adobe data breach. While the number of entries in the leaked file far exceeds the reported number by Adobe, the fact that the majority of the entries belong to adobe or related companies, as well as being single use accounts to get free copies of software, must be taken into account, making the 38 million active account estimate fairly reliable.

The user names, email addresses and password hints were stored in plaintext, while the passwords were encrypted[16]. However the encryption methods used, combined with the plaintext password hints, has proven the security of the methods used to be quite deficient. Encrypting passwords is discouraged in the first place, as hashing provides far more security for the user. The encryption algorithm used was 3DES, which together with the fact that block chaining was not used, and the passwords were unsalted, means that each block of 8 bytes can, in most cases, be easily guessed at with the help of the password hints, and this can be used in breaking other passwords on the system.

The source code that was stolen also helped the hackers in discovering several exploits in Adobe software, most notably ColdFusion, which they used to break into several other systems[24].

3.1.4 Attack Vectors

As mentioned in chapter 2 companies have little to no incentives to disclose more information about data breaches than they are legally required to. This means that finding the exact cause of the data breaches, in most cases, is impossible. But the stolen data will at least help explain where protection is needed.

As seen from the LinkedIn and Adobe breaches the use of outdated or poor data security (old non-chained encryption, not using a salt) is common, and as is confirmed by [42] data at rest is one of the two major targets (the other one being data during processing). The same report[42] also points out that financial motivations is the largest motive for breaches, followed by espionage. Ideological or purely destructive motivations are rarely observed in this context.

While the yearly OWASP Top 10[41] does not deal specifically with data breaches, instead focusing on general web application security, most of the entries on their list do allow user information to be stolen in some way. Of most relevance to this report are injection attacks, which have featured as the top risk in most, if not all, of their top 10 lists, as well as broken authentication and session management, security misconfiguration, sensitive data exposure, and using components with known vulnerabilities.

Another interesting conclusion from [42] is that among the information that has been stolen the largest categories are bank, payment, and credentials. The first two can easily by attributed to financial gains, but stealing credentials, due to poor password management on the users part, is useful to the attacker to get into other systems than the one the information originated from.

While in a lot of attacks, as can be seen from [4], the human is the weak element. Meaning the attacker relies on social engineering, phishing, and other related techniques. This report will however exclude this class of attacks from further discussion, as it is outside its scope.

3.2 Best Practices

This section will cover the best practices in secure data storage, primarily with regards to privacy and confidentiality. At the core of privacy protection as described by [20] and elaborated on by ENISA in [18], is encryption, and as such this will be the focus of this chapter and the majority of this report.

When looking at data security it is useful to keep in mind that while confidentiality may be what most people think about as security, for data security the considerations should also include authenticity, integrity and non-repudiation where applicable.

In a growing number of jurisdictions corporations are required by law to inform their customers of any data breaches that leads to the customers privacy being compromised, there are often exemptions for data that has been securely stored in such a manner as to make the data useless to the attacker. One such example is the EU electronic privacy laws which state that if the data has been rendered ”unintelligible to any person who is not authorised to access it”[18] with the definition that it is unintelligible if

> ”it has been securely encrypted with a standardised algorithm, the key used to decrypt the data has not been compromised in any security breach, and the key used to decrypt the data has been generated so that it cannot be ascertained by available technological means by any person who is not authorised to access the key;”

or

> ”it has been replaced by its hashed value calculated with a standardised cryptographic keyed hash function, the key used to hash the data has not been compromised in any security breach, and the key used to hash the data has been generated in a way that it cannot be ascertained by available technological means by any person who is not authorised to access the key.”

This further highlights the importance of strong data security, as the public backlash from a data breach can constitute a large portion of the cost of the data breach together with liability for insufficient data protection.

The core source of best practices in data security is the ISO/IEC 27000 series standards, most relevant of which is 27002 ”Information technology - Security techniques - Code of practice for information security controls”, the current version of which is ISO/IEC 27002:2013[3]. While this standard does not dictate what exact technology to use for secure storage, chapter 10 on cryptography states under implementation guidance that:

> ”When developing a cryptographic policy the following should be considered:
> b) based on a risk assessment, the required level of protection should be iden-

tified taking into account the type, strength and quality of the encryption algorithm required;
d) the approach to key management, including methods to deal with the protection of cryptographic keys. . . ”

3.2.1 Standards Sources

On the deeper topic of actual algorithms, key sizes, and parameters this report relies on four primary sources: the US National Institute of Standards and Technology (NIST), the US National Security Agency (NSA), the International Organisation for Standardization (ISO), and the European Union Agency for Network and Information Security (ENISA).
Several of the most relevant algorithms and standards originate as NSA ”competitions”, the winners of which are then been published as NIST Federal Information Processing Standards (FIPS). This together with the fact that the US in effect, by being the dominating power in international cooperations where they are involved, sets the format and standards for secure data storage and communication, specifically through their NSA Suite B Cryptography list of approved algorithms for interoperability. As a side note there are two other algorithm suites originating from the NSA, but Suite A contains only classified algorithms, and Suite C is still awaiting approval, so both are, as of the time of writing, not relevant to the topic of this report.
ISO, as the largest international organisation in charge of standardisation, is responsible for publishing and managing a vast number of international standards on a huge number of topics, including computer security best practices and standards. A notable standard is ISO/IEC 19772:2009 which standardises six different encryption modes for authenticated encryption, among them GCM.
ENISA is the EU agency responsible for advising EU member states on topics of security, and is a central agency when it comes to lawmaking, law interpretation and setting expected best practices when it comes to electronic security. As such they can be considered the authority on best practices on data security in most parts of Europe.

3.2.2 Cryptography

In the context of cryptographic stream ciphers, ciphers operating on a stream of data rather than blocks, are, while generally faster than block ciphers, very limited in their ability to provide authentication and integrity controls, and as such their use is inadvisable in this case. Asymmetric ciphers, also known as public key cryptography, are considered far too slow for larger datasets, and are often only used for signing, or for encrypting symmetric keys for transfer between communicating parties. The block cipher however is capable of using modes of operation that enable both authenticity and integrity controls, while still being reasonably fast.

When looking at the sources in unison only one block cipher is recommended for use by all of them, namely AES. NIST lists AES, 3DES and Skipjack as approved algorithms[28],

the NSA only AES[29], and ENISA lists AES and Camellia[17]. When it comes to key length the NSA requires 128-bit for information rated secret, and 256-bit for information rated top secret, while ENISA recommends 128-bit for near term, but 256-bit for long term use.

When it comes to the modes of operation for AES the NSA lists Galois/Counter Mode (GCM) as the preferred mode as it provides strong confidentiality and authentication. If authentication is not required the requirements and preferences are a bit more diverse, but all of them seems to agree that Cipher Block Chaining (CBC) provides sufficient security given a few caveats. Other modes are also approved for certain applications/circumstances, but the above two constitutes what seems to be agreed upon to be the least complicated, while still maintaining a high enough level of security[17, 29].

3.2.3 Hashing

In regards to the choice of hash algorithms, as of the writing of this report SHA-3 has not yet been published, so it really only leaves SHA-2 as the agreed upon set of algorithms, with variations in sizes of 256, 384 or 512 bits (224 is considered deprecated for future use by ENISA), the NSA requirement is 256-bit for information rated secret, while 384 is required for information rated top secret.

3.2.4 Key Management

The last topic for consideration in this section is the issue of key management. More specifically secure key storage, as the creation, deletion, and phasing out of keys is beyond the scope of this report. It is an established fact that anything stored in the memory or on disk on any computer can be accessed by any program running on the system, with or without permission from the system, and as such cannot be considered as a secure place to store cryptographic keys. NIST[27] even goes so far as to completely ignore on-system storage of keys.

> ”Keys may be maintained within a cryptographic module while they are being actively used, or they may be stored externally (provided that proper protection is afforded) and recalled as needed.”

As can be gathered from this the only recommended best practice storage solution for cryptographic keys in active use is a hardware cryptographic module or system. These recommendations or requirements often come in the form of a list of approved vendors or modules. Modules which as mentioned in the introduction chapter, are prohibitively expensive for small and medium-sized enterprises and organisations with limited IT budgets.

3.3 Cryptographic Hashing

This section will cover the topic of hashing algorithms, with a specific focus on SHA-2, as well as attacks and protections that can be applied to hashing.

While this report will not cover hashing in depth beyond this section, it is nevertheless a highly valuable tool in certain contexts. Notably providing extra protection to information you only need to verify that a third party has or knows, but you do not need to know, such as passwords. It can also be an essential tool in providing integrity/authenticity controls to information in an efficient manner.

By using hashing as an example this report aims to provide some insight into the arms race that takes place between attackers and defenders in all areas of computer security. This will be achieved by first looking at how basic hashing can be defeated by using reverse lookup tables, then at how these can be defeated using salting. And lastly at how salting, together with the availability of affordable massively parallel hardware, has led to high-paralell dictionary attacks has become the most common way to attack hashes.

3.3.1 Hashing Basics

In essence hashing is taking a block of data of any (reasonable) variable length and compressing it down into a fixed size fingerprint of that data. This should preferably be done in an efficient manner that does not lead to the fingerprint saying anything about the initial data, yet gives the exact same fingerprint for identical data. In other words it is a one way function.

To be considered a cryptographically strong hashing function there are a number of requirements, some of which have already been stated above, but are repeated here for completeness. As per [40] a Function/Algorithm H is considered a cryptographic hash if it has the following properties:

Variable input size H can be applied to a block of data of any size.

Fixed output size H produces a fixed-length output.

Efficiency H(x) is relatively easy to compute for any given x, making both hardware and software implementations practical.

Preimage resistant (one-way property) For any given hash value h, it is computationally infeasible to find y such that H(y) = h.

Second preimage resistant (weak collision resistant) For any given block x, it is computationally infeasible to find $y \neq x$ with H(y) = H(x).

Collision resistant (strong collision resistant) It is computationally infeasible to find any pair (x, y) such that H(x) = H(y).

Pseudorandomness Output of H meets standard tests for pseudorandomness.

From the description of a hashing algorithm, as a function that takes variably sized data of length n and produces a hash of a fixed length m, it is obvious that, given that n can be larger than m, collisions to exist. However collision resistance only requires that it is infeasible to find them, and some legacy algorithms that are considered broken for strong cryptographic applications can still be used for other purposes if they are still resistant to finding meaningful collisions.

3.3.2 SHA-2

SHA-2 is a family of hashing algorithms designed by the NSA and published in 2001 by NIST. This family consists of six different hash functions with outputs of 224, 256, 384 or 512 bits, the last two of the six are SHA-512/224 and SHA-512/256 which are both variations of SHA-512 with reduced output sizes. The SHA-2 family of hashing algorithms were developed to replace the similar SHA-1.

Following the discovery of a weakness in SHA-1 in 2005 its use was discouraged by NIST, requiring SHA-2 to be used instead in many applications used by federal agencies by 2010.[40]

In essence the SHA-2 algorithm works by splitting the message into 512 or 1024 bit blocks, padding the last block and processing each block separately, then XOR-ing the results of each together to form the finished hash.

SHA-3 is as of the writing of this report in the final stages before official publication, though it is not intended to replace SHA-2, but work as an alternative hashing algorithm based on different operations.

3.3.3 Rainbow tables

Looking at limited length input size fields with limited possible values, for example passwords, one intuitive way to try to get around the one way property would be to simply precompute all possible input values and their hashes, and store these in what is called a reverse lookup table. While this sounds possible in theory even a limited example shows that it is infeasible in terms of storage.

An example: Take a very limited password field, all English lower- and upper-case letters as well as all numbers. That makes 62 possible characters. Limiting the input size to exactly 8 characters gives us a total of 62^8 possible values. This is over 218 trillion possible values, which given a hash output length of 128-bits (16 bytes) stored as hash and data

without any extra characters and without compression yields 4766 tebibytes, obviously far too much data to be stored on any reasonable storage system.

If a reduction function for the hash is used to generate another valid input, which could then be hashed, and so on, this limitation could be reduced. Applied N times this algorithm could in theory, by only storing the last hash together with the initial input, reduce the storage required by a factor of N, given the perfect reduction. This is what is refered to as a precomputed hash chain. To look up a hash in such a reverse lookup table of hash chains would require more processing power, as the lookup would have to include all N possible positions of the original input in the chain. However, such a perfect reduction function does not exist and precomputed hash chains are prone to collisions when hash chains produce the same value, causing the possible lookup values to be incomplete.

This is where rainbow tables come in. Instead of defining a single reduction function a reduction function is defined for each of the N reduction steps, with the requirement that each reduction function have near absolute collision resistance (each output is unique to one input). In this way if two chains produce different values they must by definition be on different steps of their respective chains, and as such will diverge into separate chains again. This does have a slight increase in computational power required as each possible position in the hash chain would have to be computed completely independently of each other, unlike in precomputed hash chains where only a single chain would need to be computed.

3.3.4 Salting

Even with rainbow tables with a very large N the required storage and computational requirement for the table becomes infeasibly large as the input size is increased. Simply doubling the size of the input increases the storage requirement to $\frac{1.387*10^{18}}{N}$ tebibytes.

This illustrates the major weakness of precomputational attacks, large inputs. By appending a long, known, value to the input the computational difficulty of an attack of this type is increased without requiring the user or data provider to provide a longer input. This value is known as a salt. By lengthening the input for the hashing algorithm this way the attacker is left either having to generate an impossibly large reverse lookup table, or having to generate a lookup table for each unique salt. With a single salt used for all the hashed values in a database or similar storage system the latter is possible, but infeasible. With a unique salt, of sufficient length, for each entry in the database both approaches are considered impossible.

3.3.5 Cracking

Applying a salt to a value to be hashed effectively eliminates the dangers posed by precomputed reverse lookup tables. However with the increase in processing power, espe-

cially with the degree of parallelisation available on modern graphical processing units, another possible attack has become not only possible, but highly effective, namely dictionary brute-force attacks. While brute-forcing a random password of any significant length is still far too slow, as shown in the example above with over 280 trillion possible values for a very simple alphanumeric 8 character password, a basic understanding of how people chose and remember passwords can reduce the number of values worth trying to a very manageable number. Almost no one use truly random passwords,due to the difficulty of remembering them[8], and users, more often than not, tend to use the same password for most, or all of their accounts.

In a report by Cazier and Medlin[8] a set of 520 passwords from real users were run against a tool called L0phtCrack 5. Only 4 of the passwords tested held up, while 40% were cracked in under an hour. An article published by Ars Technica in 2013[19] put three experienced password crackers up against a list of 16 449 passwords hashed with the MD5 hashing algorithm. The most successful, in terms of success rate, of the three managed to crack 90% of them in around 20 hours, while the others managed 82% in 2 hours and 62% in around one.

One thing worth noting is that password crackers and modern password cracking tools use dictionaries and word lists compiled and honed by analysing real world data from leaked passwords, meaning that the tools and success rates of cracks will increase as more leaked passwords become available.

3.4 Cryptography

This section will cover, in brief, the Advanced Encryption Standard. Four modes of operation for cryptography will also be covered, namely electronic code book, the two modes from the section on best practices, CBC and GCM, and one final mode of operation that covers a usecase not covered by the others.

3.4.1 Advanced Encryption Standard

AES is a block encryption algorithm using a single 128, 192, or 256 bit key on a 128 bit block. It works by applying 4 elementary transformations on the data in 10, 12 or 14 rounds, depending on the size of the key. Since AES does not use the encryption key directly, but uses a unique key for each round, the first step is to expand the key according to what is known as the Rijndael key schedule. The following operations are then applied for each round, except the first and last, on the data organised in a four by four grid construct: First each byte in the block being processed is substituted according to a lookup table, second the second to fourth rows are shifted, third the columns are mixed, fourth and last the round key is XORed with the block. The first round only applies the last step, while the final round omits the third step.

In terms of security no attack has been found for AES, using the full number of rounds. which is more efficient than 2^{96} for AES-192. Of specific note in this context is the fact that AES, due to a large avalanche effect, is considered immune to related text attacks. The avalance effect in this context referring to the property that a small differences in input leads to large differences in the output. However some side-channel attacks, attacks on implementations that leak information in some way, have been found for several implementations of AES, though most of them have been quickly patched when the attack became known.

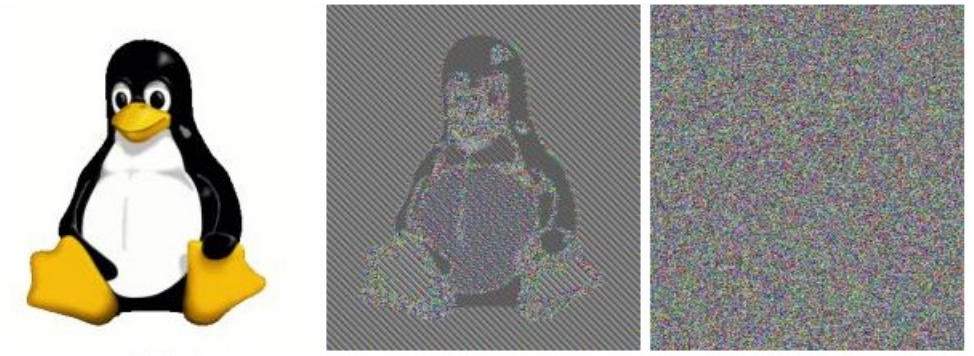
3.4.2 Modes of operation

This section will cover, in brief, four modes of operations for encryption. Two terms of note that are used extensively are Initialisation Vector (IV) and nonce, both are known random and unique values, of different lengths. The IV is a full block long, while the nonce, in this context, is shorter.

Electronic Code Book (ECB)

Electronic Code Book mode is the simplest of all the modes, as it is simply a lack of any additional operations on top of the encryption algorithm. It simply calculates each block

independently of each other block. While this means the mode is highly parallelizable, it has severe security implications. As long as the encryption key and algorithms are safe it is still impossible to directly decrypt the message, but due to a small block size (128 bits) it is possible to see patterns in the output. If you know the input for some of the output it is also possible to know the input of a given output. Figure 3.1 demonstrates this quite well by showing the ouput of ECB and CBC mode next to the original input, with a clearly visible pattern in the output of ECB mode.

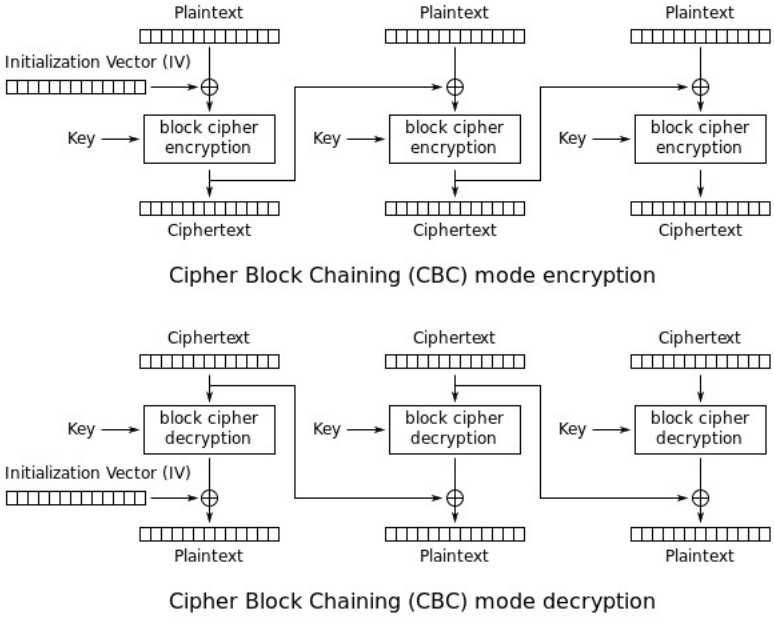


Courtesy of Larry Ewing and Wikimedia Commons

Figure 3.1: Example of output of various modes of operation

Cipher Block Chaining (CBC)

Cipher Block Chaining is among the simpler true encryption modes of operation. It works by XORing the ciphertext of the previous block with the plaintext of the current block when encrypting, and with the result of the decryption operation when decrypting. For the first block an Initialisation Vector is used. Provided the IV is unique this mode is considered secure. It does limit the operations to being serial, since the result of the previous operation needs to be known before the current operation can be processed. Figure 3.2 illustrates CBC.

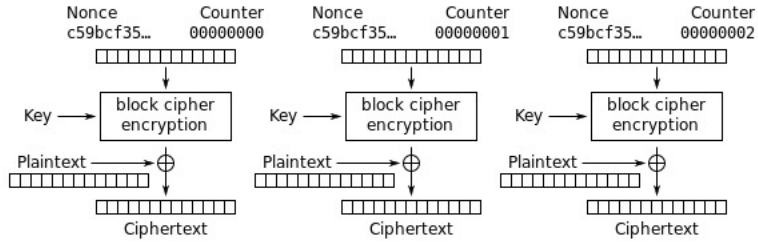


Courtesy of Wikimedia Commons

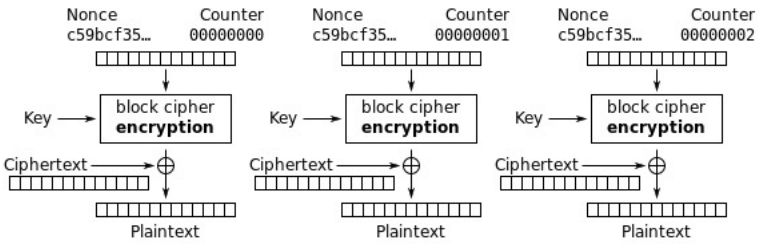
Figure 3.2: CBC Encryption and Decryption

Counter (CTR)

While this mode of operation is not explicitly mentioned by all standards bodies as approved, Galois/Counter Mode (GCM) builds upon this mode of operation and it can therefore be considered a non-authenticating version of GCM. This is also one of the more unusual modes, as it does not actually use the decrypt operation of whatever encryption algorithm it uses. It works by selecting a nonce that is short enough to allow the remaining bits in the encryption block to be used as a counter. An initial value for the counter is then selected (usually 0s), and a positive non-zero integer is selected to increment by (selecting 1 is most common). As is shown in figure 3.3, for each block, the nonce with the counter for that block appended is encrypted, and the plaintext is XORed with the result to encrypt, or the ciphertext with the result to decrypt. This operation eliminates the problem with chaining modes being serial, and as such can be parallelised linearly. Originally this mode was considered weak due to the fact that each input is sequential, but this is now commonly considered a problem with the encryption algorithm if it is weak to related text attacks. This mode does have one weakness that is relevant in the context of this report. That is that access to encrypt and decrypt information cannot be separated, as the two operations are indistinguishable.



Counter (CTR) mode encryption



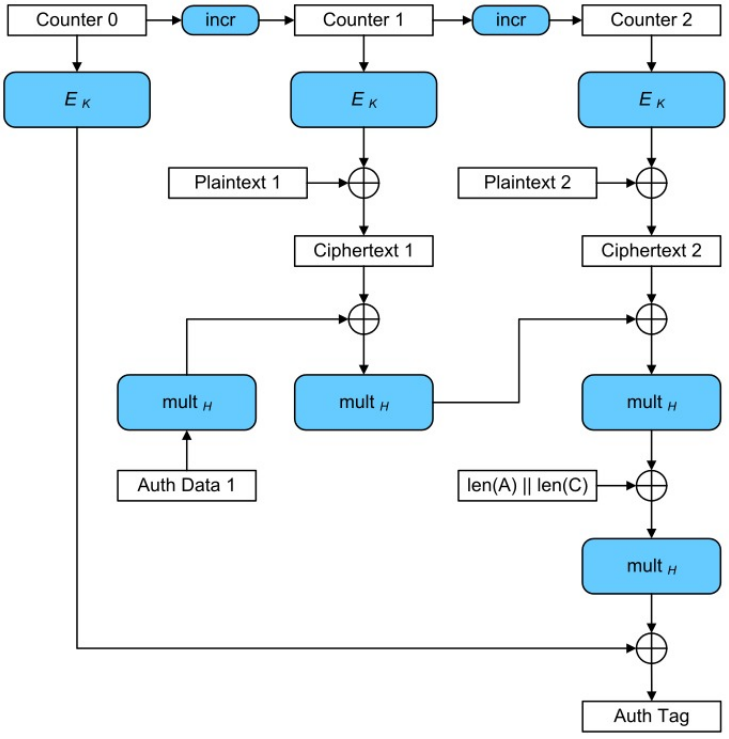
Counter (CTR) mode decryption

Courtesy of Wikimedia Commons

Figure 3.3: CTR Encryption and Decryption

Galois/Counter Mode (GCM)

Galois/Counter Mode builds upon the counter mode. The actual encryption and decryption is functionally identical, but it uses the first of the outputs for authentication. GCM does however append a construct on the standard counter mode, as seen in figure 3.4. This construct utilises Galois field multiplication on the ciphertexts to construct an authentication tag, which can be used to verify the authenticity of the data. This mode of operation is the preferred mode of several standards bodies, and has been proven to be secure, provided a suitable initialisation vector and encryption algorithm is chosen. Additionally due to being based on counter mode each block can be encrypted separately but the authentication tag has to be calculated serially, resulting in almost linear scaling.



Courtesy of United States Department of Commerce

Figure 3.4: GCM Encryption

3.5 Key Management

As mentioned in the section on best practices, key management is considered the hardest part of cryptography in practice, and it is often the one universal weak spot across all cryptographic algorithms and methods. While the concept of cryptographic key management includes various topics, such as generating and replacing keys, exchanging them between systems that need them, and storing and using them, the main focus of this report will be storage and secure usage of keys. Storing cryptographic keys on any generic computer is considered bad practice, as both RAM and permanent storage can easily be accessed by any malicious application once it has gained access to the computer in question, and hence the only solutions that are considered secure are based on implementations on secured hardware.

3.5.1 Hardware Solutions

As mentioned in the section on best practices the only solution to key management that is universally accepted is Hardware Security Modules (HSM). There are a lot of different modules available, all of them providing both full key management (creation, storage, deletion) as well as full in-device cryptoprocessing for large amounts of keys. Most HSM' are designed to be tamper proof, both in terms of digital and physical attacks, and are capable of logging, alerting, and taking actions based on detected or suspected attacks such as deleting all key material on the device. HSM' come as either plug in cards for servers or clients, or as separate units, the latter often being preferred due to being a completely separate unit capable of handling multiple simultaneous users without having to reside within a dedicated server.

3.5.2 Certifications

Certification in this context means a formal acknowledgement that a device adheres to a certain standard. There are two certifications that are common for Hardware Security Modules: The US FIPS 140-2[1] and the more general Common Criteria (defined by ISO/IEC 15408). Sometimes the FIPS 140-2 standard is used instead of the Common Criteria when it comes to cryptographic modules by organisations and institutions that rely on the Common Criteria for other computer security certifications. This could indicate that FIPS 140-2 is a more relevant and applicable standard for HSMs. OpenDNSSec recommends certification level 2 or 3 for FIPS 140-2 certified devices, and EAL 4 or up for Common Criteria certified devices[30].

Chapter 4

System Proposal

As the previous chapter pointed out the main challenge of cryptography, once a suitable algorithm and key strength has been chosen, is the problem of key management. There are no acknowledged secure methods of storing the keys on a server or client computer without exposing it to data breaches. The accepted solution to this problem is the use of, often quite expensive, hardware solutions.

In this chapter a solution that exclusively deals with the problem of key storage, and by extension the cryptographic operations that needs the stored keys, will be proposed using off-the-shelf inexpensive microcontroller development kits. Worth noting here is that FIPS 140-2 certification is not possible, as the system is proposed, due to both the extensive nature of the requirements, but also due to the cost and time required for the certification process. The system is designed to replace a very small subset of the intended use cases of the HSMs, and as such will probably not be usable for any company or organisation where certifications of security is required or where the usecases differ significantly.

The development kit used for this system is an Arduino brand kit, but the idea extends to most other brands of microcontroller development kits. The Arduino family of kits was chosen because it is the most popular and easily available, and because it has both an established development environment and an active community.

As illustrated in figure 4.1 the imagined deployment case would be connecting the Arduino to a server responsible only for communicating with the Arduino and thereby acting as a mediator system. This system of server and Arduino could be used either as a gateway between the storage server and the client(s) (A) or as a system that the client would have to explicitly communicate with for data encryption and decryption (B).

In both architectures the server will do all the communication with the Arduino. The

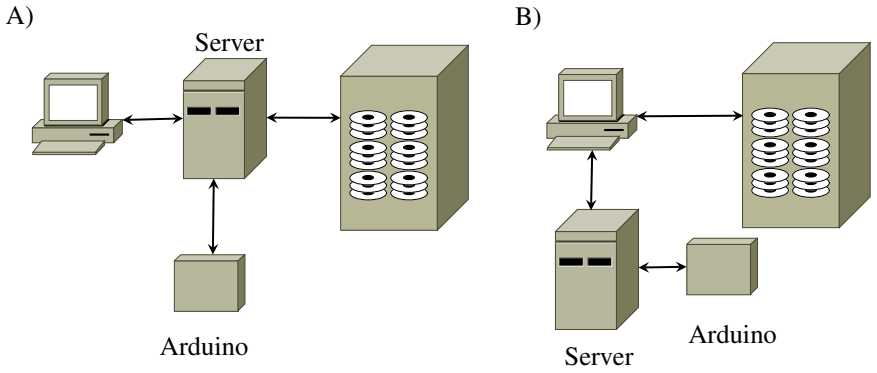


Figure 4.1: Imagined Architecture

difference is in how clients get the data. In A a client will directly query the server for the data, and the server is responsible for getting the requested information from the storage server. While in B the client requests the information from storage, then requests that the server decrypt it, sending the data to the server in the process. The architecture in A is by far the preferable one as it allows for transparency towards the client, the client can treat the server as if it was the storage server, while B requires that the client is aware of the encryption. B will also cause a far higher bandwidth usage on the network.

Using a known AES library the Arduino will have a very limited API, limited to setting the operation (encryption or decryption), as well as sending the data to be encrypted. The encryption keys are stored on the board, and should be transferred to it from a non-networked computer, so that the key is never exposed. While generating the key on the board is a possibility, it is unfortunately difficult as the key might need to reside on several devices, and allowing the key to be extracted from the Arduino could potentially compromise it. As few operations as possible should be run on the Arduino, as its limited processor speed is the single largest limiting factor along with the USB transfer speed to and from the device.

Any operation not using the key is done on the server side, such as the encryption-mode operations. Encryption-mode operations here referring the the operations performed in order to implement the mode of operations, such as the XORs in CBC and CTR. This will both serve to minimise the potential for bugs on the Arduino and to speed up the operations to as large a degree as possible given the limited hardware on the Arduino.

This solution will protect the encryption keys quite effectively, as physical access to the Arduino will be needed to extract the keys from it, and would probably have to involve quite extensive, invasive and expensive alterations to the hardware. It does still allow compromised clients or a compromised server to decrypt data available to it. In case of the server being compromised there is little to be done except trying to detect the intrusion as early as possible. This is why the server, as a weak spot in the proposed system, should run the absolute minimum of software and be kept rigorously up to date. This would limit the amount of attack vectors available to an attacker and keep bugs to a minimum. A

protection mechanism, in case of a client being compromised, would be to limit and restrict the throughput and data available to each client, and by doing so limit the severity of any data breach. This should not serve as a replacement for rigorous logging and detection efforts, but will serve well as an additional protection measure.

Chapter 5

System Feasibility Study

The proposed system is nearly identical to existing systems, such as HSM', when it comes to communication flow. The only major exception being the Arduino and how it communicates with the server. This means a feasibility study on the system can be limited to looking at the server-Arduino part of the system. The software on the Arduino is identical regardless of which of the two proposed architectures from the previous chapter is chosen, so a choice need not be made to proceed with testing. The focus of this implementation will be to test the throughput possible with the Arduino.

5.1 Implementation

As the purpose of this proof of concept (PoC) system is mainly to benchmark the performance of AES on an Arduino, only the software running on the Arduino has been fully developed. The server side software only includes the bare minimum required to transfer data to the Arduino and measure the performance. The software for the Arduino is written in C/C++. Initially the software for the server side was implemented in Python, but after running some tests with Java it was determined that performance was more than doubled, so the server side software was ported over to Java. For the serial communication over USB between the server and Arduino the Java library jSSC is used. Additionally the cryptographic keys are hardcoded into the Arduino software, so as to avoid having to implement a system for transferring and storing them. For testing purposes 128-bit keys were selected because that is considered secure enough for most applications, and it also reduces the number of rounds the AES implementation has to calculate.

5.1.1 Hardware

The main Arduino board selected for the PoC system was the Arduino Due, which uses the AT91SAM3X8E microcontroller, which is an ARM Cortex-M3 32-bit based chip running at 84 MHz. 512 kiB of flash memory is available for the application, and 96 kiB SRAM is available for runtime storage. Due to the Arduino Due not becoming available until late in 2013 an Arduino Leonardo was used for initial feasibility testing. This Arduino is based on an ATmega32u4, which is an 8-bit microprocessor running at 16 MHz and with 32 kiB of flash memory, and 2.5 kiB of RAM. As a server for benchmarking purposes a Lenovo T61 was used.

5.1.2 AES Library

As mentioned original testing of the feasibility of the system was performed on the Arduino Leonardo, using a byte-oriented AES library by Brian Gladman, that was modified for the Arduino by Mark Tillotson (`http://utter.chaos.org.uk/~markt/AES-library.zip`). However upon testing this library on the Due the librarys use of AVR assembly code, which the Due being an ARM based chip does not support, was discovered. And due to the Arduino native libraries being somewhat limited in many respects most AES libraries are simply not suitable to be run on it without major changes. However a very basic, almost dependency free, library written in C by Vincent Rijmen, Antoon Bosselaers, Paulo Barreto, and Philip J. Erdelsky was found and successfully put to use (`http://www.efgh.com/software/rijndael.htm`).

5.1.3 Arduino API

For this PoC system a very simple API is used, it uses ";" as a separator, and either takes a 2 character command or a 32 character data block. While the block size for AES is 128 bits, or 16 bytes, 32 bytes has to be used to transfer it as a string. And since standard hexidecimal using 0-9 and a-f requires two conditionals to decode a more efficient encoding of A to P is used, allowing the encoded string to be converted to its original half-byte value by simply subtracting the value of A. For this report this encoding system will simply be referred to as A-offset-hex. The 2 character command consists of either an "e" or a "d" for encryption or decryption, and a digit 0 to 9 to select the key to use for any subsequent data. This is all that is required for the Arduino to be able to encrypt and decrypt any AES block. While it would have been possible to send data in binary, reducing the transmitted size of a block to 16 bytes, the control and decoding logic required would have been far too extensive to actually gain any performance from the reduced data that would be sent.

As mentioned in the proposal the computation necessary for encryption modes of operations is done entirely on the server, making the above API technically and implementation of electronic code book mode.

However, in order to test whether an implementation of counter mode would run faster than the ECB mode and additional API was used. This API takes a single 30 character command followed by the separator. This command states the initial value of a 12 byte nonce for counter mode, as well as a digit to increment by, four digits for the number of blocks to encrypt, and finally the key to use. The Arduino will then reply with the desired number of new-line separated encrypted blocks. This removes the need to transfer the data to the Arduino, and may increase performance. This is sufficient for testing purposes, but to enable parallelism using counter mode the initial value will have to be extended to the full 16 bytes in order to allow each separate Arduino to process blocks starting at an offset counter value.

5.1.4 Verification

A small set of test vectors for AES were used to test that the library functions properly, and the result of all operations of encrypt and decrypt are verified to result in the same output as input. While this is not formal verification of the validity of the implementation of AES it should suffice to ensure the validity of the benchmarking results.

5.2 Results

In order to test the implemented software a set of 3 pseudorandom keys were generated using an online tool, and an additional key was taken from the aforementioned test vectors. Additionally 10 sets of 10kiB pseudorandom data files were generated using the Unix tool ”dd” with input from \dev\random. For ECB mode each input file was combined with each key. These were then run several times each, for a total of 1023 full 10kiB runs both for encryption and decryption. For the counter mode, only one of the keys were used. For this mode 640 blocks were generated for each run, which is the number of blocks in a 10kiB file. This was then run 1014 times. The runtime of each operation was then measured for each set of 10kiB of data, from the time, in nanoseconds, the first block is sent to the last has been received.

The result of these are summarised in the table below.

	ECB Encryption	ECB Decryption	Counter Mode
Mean	28.719 kiB/s	28.794 kiB/s	241.493 kiB/s
Standard Deviation	0.852 kiB/s	0.726 kiB/s	20.999 kiB/s
Relative Standard Deviation	2.97 %	2.52 %	8.70 %

As the ECB mode was implemented first the performance of the implementation was not as fast as initially expected, but given the limited hardware available it was not very surprising. Once the CTR mode was implemented its performance of came as a surprise, and the source of the increase in throughput has not, as of the writing of this report, been fully understood. While the counter mode does cut the data transfer in half, as well as only require half as many conversion operations between A-offset-hex and unsigned characters, that these factors alone could account for an order of magnitude in performance increase is surprising. Other likely factors, such as compiler optimisation from the decryption part of the library no longer being used, cannot be ruled out.

Grouping the throughput observations together into groups of 0.5 kiB/s for ECB and 10 kiB/s for CTR mode, and then plotting them into a graph based on the number of observations for each grouping produces the following graphs:

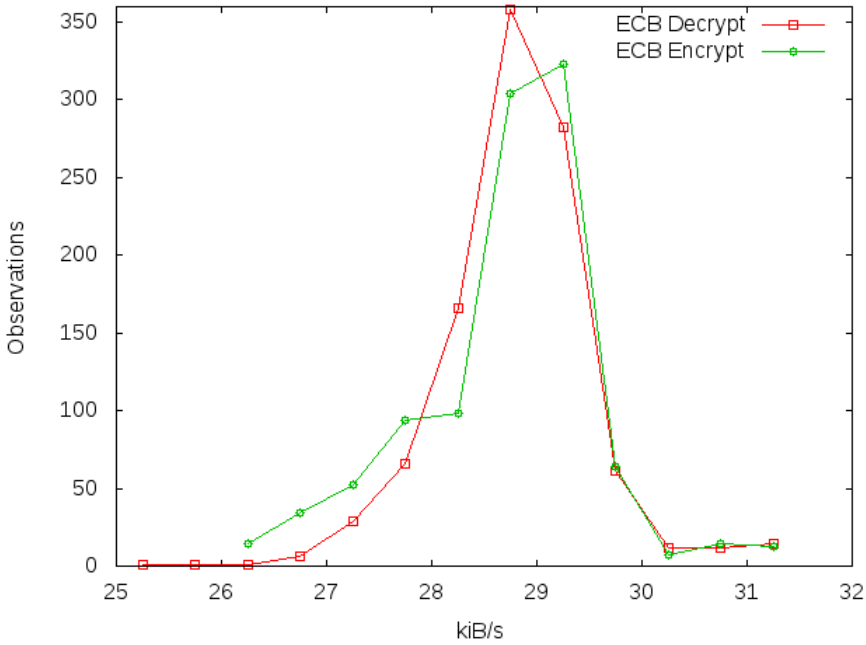


Figure 5.1: ”Observations for ECB mode (range is 0.5 kiB/s)”

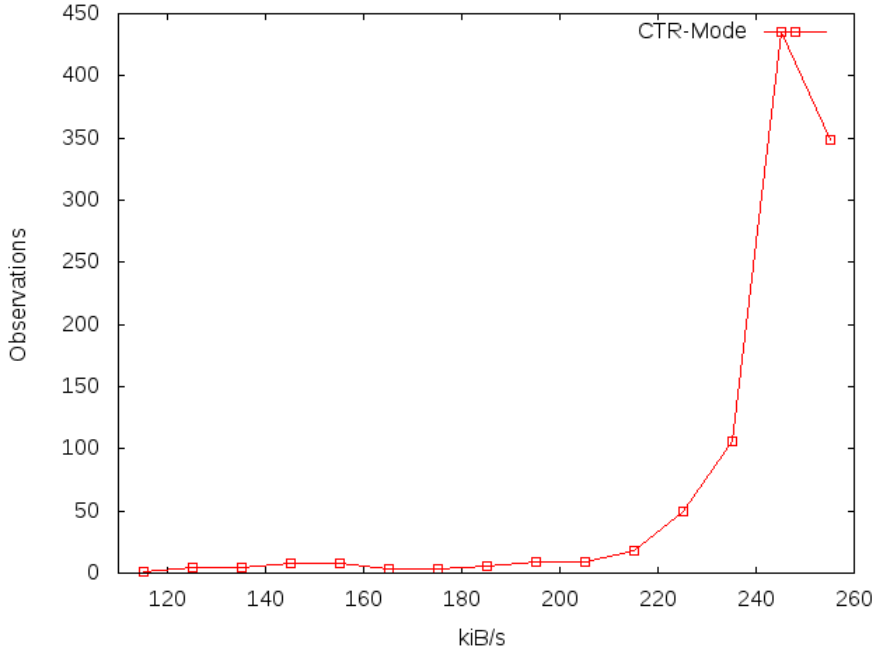


Figure 5.2: ”Observations for CTR mode (range is 10 kiB/s)”

While not extremely useful, these graphs do show that the observations are clustered differently for the two modes. ECB mode has a number of observations of both higher and lowers speeds than the most common grouping. For counter mode there are no observations that are more than one group faster than the most common group, and a large number of them more than one group slower. The most logical reason for this, which would also seem to be supported by the order of magnitude in performance difference, would be that the serial transfer over USB is a large limiting factor in the test. This would also be supported by the difference in throughput by this system compared to both other systems as well as the theoretical maximum performance given the minimum number of cycles needed for AES, in absence of hardware implemented AES.

5.2.1 Other Systems

This section will briefly cover other implementations and performance measurements of relevance for this report. Lee, Lee, And Shin[25] implemented AES on an 8-bit microcontroller running at 20MHz in 2009. While the published data is contradictory, an assumption that their table of performance incorrectly uses ms to denote microseconds correspond with the other data in the table. If this is correct their implementation at 16 byte data inputs manages a throughput of 34.7 kiB/s. Using field-programmable gate arrays [36] manages

to push the throughput up to over 44 MiB/s at 71.5MHz. Based on performance data from [26, 39] wikipedia[43] provides some useful data on the performance of AES on traditional CPUs. At 200MHz an 11 MiB/s throughput should be achievable, and for 1.7GHz this should increase to 60MiB/s. For modern CPUs that include the AES-NI instruction set the throughput can exceed 700 MiB/s. Data on the throughput of AES on HSM' has turned out to be impossible to find, as the focus of performance tests on such systems mainly focus on other cryptographic algorithms and applications, it can nevertheless be assumed to be in the GiB/s.

5.2.2 Suitability

While the implementation in this report compares favorably to the implementation on an 8-bit microprocessor[25], however compared to FPGAs and CPUs the throughput is severely limited.

This makes it unsuitable for any application dealing with large amounts of data, but protecting user data should not be outside the possible scope of this system. Looking at a rather extensive set of attributes for an individual user: full name, address, city, state, postal code, country, email address, password, credit card number, Card Verification Value, and social security number. Storing all of these as strings would, at most, require 500 bytes given the use of UTF-8 or similar. With CTR mode the Arduino would be able to generate the strings necessary for this in around 2 milliseconds. While further operations would be required to decrypt the data these operations would be performed on the server, and therefore would not require any significant additional time. In other words this system may have the potential to support up to 500 users per second per Arduino, given that the issues below can be resolved.

During testing two issues were discovered with the implementation, which is assumed to be implementation problems rather than a hardware issue, that is worth noting. First is a very small number of incidents of one block of data in the 10kiB file being encrypted incorrectly, either by omission of the last character, or an incorrect result. The second is that, in some cases, when sending two queries too fast to the Arduino, it would fail to respond to the second. Due to time constraints and the limited nature of both issues they were not pursued further, but could be eliminated by querying two Arduinos for each set of data, and implementing a simple timeout, if it turns out to be a problem with the hardware.

Chapter 6

Conclusion

This chapter briefly summarises the information from the previous chapters in terms of the four research questions that were posed in the introduction to this report. While the questions are connected, this chapter will look at each of them separately.

6.1 Research Question 1

> What are the current best practices for securing data stored at rest?

Based on the standards studied in this report, in addition to keeping software up to date, the best practices for data at rest includes hashing what can be hashed, using an up to date hashing algorithm, utilising a strong salt, as well as encrypting all sensitive information that cannot be hashed. In terms of actual algorithms there is a strong preference towards SHA-2 and AES, the latter in combination with an approved mode of operation such as CBC, CTR or GCM. When looking at encryption it is also considered essential that the keys used are stored in a secure manner, which exclusively consists of dedicated hardware solutions. This is due to the exposed nature of ram and disk storage on general purpose computers.

6.2 Research Question 2

> Why is it that despite the above best practices data still leaks? Are they too abstract, outdated, weak, or are they simply not being implemented properly?

This has been a difficult question to look into since there are few or no incentives for companies to release any kind of in depth information on exactly how a data leak happened. Even when they are legally required to inform affected parties of the leak they are not required to give any in-depth information. This means that any published research on the topic goes into very little detail on the actual attack vectors used.

What little information has been gained through analysis by third parties, often not peer reviewed and only published on discussion forums or blogs, seems to hint at an endemic problem of security management. This can be seen, for example. in the information that suggests the Sony leaks were caused by outdated software on the back-end servers. And from the leaked passwords from the LinkedIn breach not using a salt, and using an outdated hashing algorithm. Another problematic area was illustrated in the Adobe data breach, where through the theft of source code hackers were able to find weaknesses in the software and perform attacks on third parties that were using it. As is shown by the history of OWASP Top 10 articles is that weaknesses in web applications tend to involve the same classes of attacks, with for example injection attacks being the uncontested leading security risk.

Together the information available has shown that data at rest cannot be considered inaccessible to any malicious party, and thus a basic security best practice is that all data at rest should be stored in a such a way as to make it unreadable to any malicious party, i.e. encrypted, or hashed and salted.

6.3 Research Question 3

> What are the major strengths of the available methods in secure data storage (i.e. hashing, cryptography, key management), and what are their most important weaknesses?

Both AES and SHA-2 are considered fast algorithms, often only requiring only a few hundred cycles per block. AES is also considered secure for all keys, and immune to related text and key attacks. SHA-2 also as of the writing of this report is considered secure, given that a proper salt is implemented.

The only major weakness seen in secure data storage, provided it is implemented properly according to best practices, seems to be in key management. Hardware solutions for key management often require, by law, regulation, or requirements, the implementation of extensive standards, which makes development expensive. Often customers will demand compliance with certification processes, further adding to the cost of development. This is then reflected in the fact that these hardware solutions are extremely costly for small and medium sized organisations. These Hardware Security Modules are more often than not capable of a throughput that is orders of magnitude larger than what is required for small and medium sized customers, and often implement a feature set that includes far more functions than is required for a typical usecase.

This is the reason the system in this report is suggested and evaluated using a proof of concept system, its purpose being to provide functionality to secure the key from theft at a low price point, while still providing sufficient throughput for something like a small or medium sized company.

6.4 Research Question 4

> Could a proof of concept system, utilising affordable off-the-shelf microprocessor kits, alleviate some of the weaknesses of on-server cryptography (most notably key management) without compromising the strengths?

The implementation of the suggested system on the Arduino, while lacking the full set of functionality suggested by the system proposal, did effectively illuminate the question at hand.

The implementation, using a very limited API, can be assumed to provide sufficient security from software based attacks, and as such can be considered to have eliminated the major weakness in cryptography for data at rest.

In terms of maintaining the strengths of AES it was a moderate success. While it did nothing to weaken the AES algorithm, the hardware available on the Arduino board has serious limitations in terms of speed. The system proves to be inadequate for encrypting any large sets of data. It did demonstrate that, given the proper selection of mode of operation, it can be used to protect data on the customer such as login details, personal and financial information. This data is what has been shown to be the main target of most of the data breaches in recent years. And it seems to be the most costly to lose for both the company and other affected parties. Additionally the loss of customer data has often shown to have a negative impact on the public image of the company.

6.5 Future Work

While this report has provided some insight into both security culture, as related to data breaches, as well as having looked into one particular possible way to mitigate the effect of these breaches, time and other limitations has meant that several areas has remained unexplored, or only partially explored. This section will briefly cover how the topics in this report could be expanded upon.

In terms of data breach cases there is little to be done in the time frame allotted for this kind of report, but given sufficient time and access to the right parts of the security industry a more thorough look into the incidents would be possible. Of specific interest would be the actual attack vectors used, be they outdated software of a specific type, or lack of security on certain areas of the network. There is a reason why this information is not available already, namely that this is information a company is very much unwilling to part with, as it can be both considered business critical, and potentially very damaging to the company. This means that any exploration of this topic needs to heavily weigh the value of the information versus the amount of that work would be required to gather the information.

The main topic of interest for further study is the suggested hardware solution. Which consists of three main areas: The Arduino code, the server code, and integration.

The Arduino code as it exists, as of the writing of this report, while optimised to the best of the ability of the developer, could still be optimised further in terms of the use of assembly code, and it could also illuminate how much efficiency there is to be gained through assembly on such platforms. Further the Arduino could also be extended to allow it to handle key creation, key import, as well as key transfer from one Arduino to another.

Looking at the server the most interesting topics of further study would be the encryption modes. Especially Galois/Counter Mode, as this provides authenticated encryption. Parallelism could also be explored on this topic by utilising several Arduinos connected to the same server, thus allowing either several separate sets of data to be encrypted/decrypted at once, or to allow several Arduinos to cooperate on the encryption or decryption.

For integration it would be interesting to explore the possibility of implementing known APIs, such as the Private Key Cryptography Standard #11, which despite its name actually applies to far more than private key cryptography and is often refered to as Cryptoki. Also of interest would be to integrate this system with APIs and other systems, allowing for transparent encryption of for example databases.

In short this report could provide the basis of several further reports on this, and other related topics.

Bibliography

[1] (2001). Fips pub 140-2: Security requirements for cryptographic modules. http://csrc.nist.gov/publications/fips/fips140-2/fips1402.pdf.

[2] (2009). Iso 29772: Information technology - security techniques - authenticated encryption.

[3] (2013). Iso 27002: Information technology - security techniques - code of practice for information security controls.

[4] Armerding, T. (2012). The 15 worst data security breaches of the 21st century. http://www.comnetmarketing.com/documents/15_worst_data_security_breaches_071613.pdf.

[5] BBC (2011a). Sony investigating another hack. http://www.bbc.com/news/business-13636704.

[6] BBC (2011b). Sony reports online security breach on various websites. http://www.bbc.com/news/business-13537128.

[7] Bonneau, J. (2012). The science of guessing: analyzing an anonymized corpus of 70 million passwords. In *Security and Privacy (SP), 2012 IEEE Symposium on*, pages 538–552. IEEE.

[8] Cazier, J. A. and Medlin, B. D. (2006). Password security: An empirical investigation into e-commerce passwords and their crack times. *Information Systems Security*, 15(6):45–55. http://www.tandfonline.com/doi/abs/10.1080/10658980601051318.

[9] CLytle (2011). Possible playstation network attack vectors. http://blog.veracode.com/2011/05/possible-playstation-network-attack-vectors/.

[10] CNet (2012). Millions of linkedin passwords reportedly leaked online. http://www.cnet.com/news/millions-of-linkedin-passwords-reportedly-leaked-online/.

[11] CNN (2012). More than 6 million linkedin passwords stolen. http://money.cnn.com/2012/06/06/technology/linkedin-password-hack/.

[12] Constantin, L. (2013). Linkedin wins dismissal of lawsuit seeking damages for massive password breach. http://www.pcworld.com/article/2030129/linkedin-wins-dismissal-of-lawsuit-seeking-damages-for-massive-password-breach.html.

[13] Curry, C. (2012). 6.4 million passwords reportedly stolen from linkedin website. http://abcnews.go.com/US/linkedin-hacked-64-million-user-passwords-reportedly-leaked/story?id=16508728.

[14] DataLossDB, T. O. S. F. (2014). Data loss statistics. http://datalossdb.org/statistics.

[15] Dignan, L. (2011). Sony's data breach costs likely to scream higher. http://www.zdnet.com/blog/btl/sonys-data-breach-costs-likely-to-scream-higher/49161.

[16] Ducklin, P. (2013). Anatomy of a password disaster - adobe's giant-sized cryptographic blunder. http://nakedsecurity.sophos.com/2013/11/04/anatomy-of-a-password-disaster-adobes-giant-sized-cryptographic-blunder/.

[17] ENISA (2013a). Algorithms, key sizes and parameters report. https://www.enisa.europa.eu/activities/identity-and-trust/library/deliverables/algorithms-key-sizes-and-parameters-report.

[18] ENISA (2013b). Recommended cryptographic measures - securing personal data. https://www.enisa.europa.eu/activities/identity-and-trust/library/deliverables/recommended-cryptographic-measures-securing-personal-data.

[19] Goodin, D. (2013). Anatomy of a hack: How crackers ransack passwords like "qeadzcwrsfxv1331". http://arstechnica.com/security/2013/05/how-crackers-make-minced-meat-out-of-your-passwords/.

[20] Guerses, F., Berendt, B., and Santen, T. (2006). Multilateral security requirements analysis for preserving privacy in ubiquitous environments. *Proceedings of the UKDU Workshop*, pages 51–64. https://lirias.kuleuven.be/bitstream/123456789/164018/1/SedaBettinaThomasUkdu2006.pdf.

[21] Halfacree, G. (2013a). Adobe breach leaks source, millions of customers' details. http://www.bit-tech.net/news/bits/2013/10/04/adobe-breach/1.

[22] Halfacree, G. (2013b). Adobe data breach far worse than first claimed. http://www.bit-tech.net/news/bits/2013/11/05/adobe-breach-2/1.

[23] Hassan Alnoon, S. A. A. (2009). Executing parallelized dictionary attacks on cpus and gpus. *moais. imag. fr*. http://moais.imag.fr/membres/jean-louis.roch/perso_html/transfert/2009-06-19-IntensiveProjects-M1-SCCI-Reports/HassanShayma.pdf.

[24] Krebs, B. (2013). Breach at pr newswire tied to adobe hack. http://krebsonsecurity.com/2013/10/breach-at-pr-newswire-tied-to-adobe-hack/.

[25] Lee, H., Lee, K., and Shin, Y. (2009). Aes implementation and performance evaluation on 8-bit microcontrollers. *arXiv preprint arXiv:0911.0482*.

[26] McWilliams, G. (2011). Hardware aes showdown - via padlock vs intel aes-ni vs amd hexacore. http://grantmcwilliams.com/tech/technology/item/532-hardware-aes-showdown-via-padlock-vs-intel-aes-ni-vs-amd-hexacore.

[27] NIST (2012). Special publication 800-57: Recommendation for key management: Part 1: General. http://csrc.nist.gov/publications/nistpubs/800-57/sp800-57_part1_rev3_general.pdf.

[28] NIST (2014). Block ciphers. http://csrc.nist.gov/groups/ST/toolkit/block_ciphers.html.

[29] NSA (2014). Suite b cryptography. http://www.nsa.gov/ia/programs/suiteb_cryptography/.

[30] OpenDNSSec (2012). Hsm buyersǵuide. https://wiki.opendnssec.org/display/DOCREF/HSM+Buyers%27+Guide.

[31] Plunkett, L. (2014). Sony agrees to give away games, money after 2011 psn hack. http://kotaku.com/sony-agrees-to-give-away-games-money-after-2011-psn-ha-1609938674.

[32] Ponemon Institute (2013). 2013 cost of data breach study: Global analysis. http://www.ponemon.org/library/2013-cost-of-data-breach-global-analysis.

[33] Ransom-Wiley, J. (2011). Psn breach and restoration to cost $171m, sony estimates. http://www.joystiq.com/2011/05/23/psn-breach-and-restoration-to-cost-171m-sony-estimates/.

[34] Risk Based Security (2013). ”data breach quickview 2012”. https://www.riskbasedsecurity.com/reports/2012-DataBreachQuickView.pdf.

[35] Risk Based Security (2014). ”data breach quickview 2013”. https://www.riskbasedsecurity.com/reports/2013-DataBreachQuickView.pdf.

[36] Rouvroy, G., Standaert, F.-X., Quisquater, J.-J., and Legat, J. (2004). Compact and efficient encryption/decryption module for fpga implementation of the aes rijndael very well suited for small embedded applications. In *Information Technology: Coding and Computing, 2004. Proceedings. ITCC 2004. International Conference on*, volume 2, pages 583–587. IEEE.

[37] RSA Laboratories (2009a). Pkcs#11 v2.30 core specification. ftp://ftp.rsasecurity.com/pub/pkcs/pkcs-11/v2-30/pkcs-11v2-30b-d6.pdf.

[38] RSA Laboratories (2009b). Pkcs#11 v2.30 mechanisms. ftp://ftp.rsasecurity.com/pub/pkcs/pkcs-11/v2-30/pkcs-11v2-30m1-d7.pdf ftp://ftp.rsasecurity.com/pub/pkcs/pkcs-11/v2-30/pkcs-11v2-30m2-d3.pdf.

[39] Schneier, B., Kelsey, J., Whiting, D., Wagner, D., Hall, C., and Ferguson, N. (1999). Performance comparison of the aes submissions. https://www.schneier.com/paper-aes-performance.pdf.

[40] Stallings, W. (2011). *Cryptography and Network Security*. Pearson, 5th edition.

[41] The Open Web Application Security Project (2013). Owasp top 10 2013.

[42] Verizon (2014). 2014 data breach investigations report. http://www.verizonenterprise.com/DBIR/2014/.

[43] Wikipedia (2014a). Advanced encryption standard. https://en.wikipedia.org/wiki/Advanced_Encryption_Standard.

[44] Wikipedia (2014b). Key management. https://en.wikipedia.org/wiki/Key_management.